AmiBroker 6.90 User's Guide

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Copyright

AmiBroker 6.90 User's Guide

Contents

Introduction

Thank you for choosing AmiBroker. This guide will help you get up and running.

AmiBroker is a comprehensive technical analysis program, with an advanced charting, back-testing and scanning capabilities. It gives everything you need to trade successfully. Just check out our quick features tour to find out what is included in this powerful software package.

If you are a **first time user** and just installed the software please check out **Tutorial section** that will guide you through most important aspects of using AmiBroker.

The next chapter - Reference guide - provides detailed description of every window and more technical documentation covering ASCII importer and automation interface.

In the Technical analysis guide you will find material that will introduce you to the world of charting and technical indicators.

The next part of the guide describes AmiBroker Formula Language - a powerful tool that allows you to create your own trading systems, scans, custom indicators and commentaries. You will find the description of the language and its syntax, a complete reference of all functions and more.

The last part is provided for the user's of previous versions - this chapter will help them finding out what new features were added without the need to re-read all documention.

About AmiBroker Editions

AmiBroker software is currently available in 2 editions: Standard and Professional.

The following table summarizes differences between these two editions:

Feature	Standard Edition	Professional Edition
End-of-day charting/backtesting/scanning	Yes	Yes
1-, 5-, 15- minute, hourly Intraday charting/backtesting/scanning	Yes	Yes
Custom minute bars	Yes	Yes
Tick charts/backtesting/scanning	No	**Yes***
1-second, 5-second, 15-second bar charts/backtesting/scanning	No	**Yes**
Streaming real time quote display	10 symbols	**UNLIMITED symbols**
Time and Sales window	1 symbol	**UNLIMITED symbols**
GetRTData / GetRTDataForeign AFL function	No	**Yes**
Wait for backfill in Automatic Analysis	No	**Yes**
Automatically updating real time charts	Yes	**Yes**
Maximum Adverse/Favourable Excursion Distribution charts in Portfolio backtest reports	No	**Yes**
64-bit version	No	**Yes**
Multi-threading Charts	Yes	Yes
Multi-threading Analysis window	Yes, upto 2 threads	Yes, **upto 32 threads ****
Bitness	32-bit only	64 bit and 32-bit
Requires RT data subscription	No	Not required, but nice to have (Professional Edition works with EOD data perfectly fine, but real-time features (like real-time quote) of course are require real-time data source)

* - this feature is available only using eSignal RT, Interactive Brokers, DDE feed
In the future the Professional Edition may have additional extra features not available in Standard Edition. For pricing and ordering information check out How to order section.

** - the number of threads depends on number of logical processors on your computer and number of symbols under test. For details see: Efficient use of Multithreading.

Quick Tour

Basic features

Powerful charting

- **object-oriented drawing tools (trend lines, rays, parallel lines, regression channels, fibonacci retracement, expansion, Fibonacci time extensions, Fibonacci timezone, arc, gann square, gann square, cycles, circles, rectangles, text on the chart, and more)**
- **drag-and-drop indicator creation -** allows you to create complex indicators without writing single line of code
- **modern, fully customizable user interface**
- instant viewing of intraday/daily/weekly/montly charts in line, bar or candlestick styles overlaid with configurable moving averages, Bollinger bands, Volume chart, SAR, etc.
- ability to display most common 1-, 5-, 15-, 60- minute intraday charts as well as **fully customizable N-minute charts (where N is 1..1380)**
- **5-second and 15-second bar charts (RT version)**
- **tick charts, custom N-tick charts (RT version)**
- **multiple time frame charts**
- on-the-fly time compression - no need to wait when switching between various chart periodicities
- **relative performance charts**
- tens of most popular indicators built-in including ROC, RSI, MACD, OBV, CCI, MFI, NVI, Stochastics, Ultimate oscillator, DMI, ADX, Parabolic SAR, TRIN, Advance/Decline line, Accumulation/Distribution, TRIX, Chaikin oscillator, unique risk-to-yield map and more
- study drawing tools including trend lines, horizontal/vertical lines, Fibonacci retracements and timezones, text boxes
- multiple chart panes, windows, different views and time scales are possible all at the same time
- extermely fast zooming and live scrolling

Multiple data feeds

AmiBroker is capable of handling virtually ANY exchange in the world.

- **Real-time streaming quotes via eSignal's TurboFeed featuring access to all US exchanges and major European exchanges.**
- **Real-time streaming quotes via myTRACK feed, IQFeed**, **QCharts/Quote.com, QuoteTracker, Interactive Brokers, any DDE-enabled data feed**
- **Direct** feed from Quotes Plus, TC2000, FastTrack and Metastock (including intraday) databases. Read more...
- User-configurable ASCII import wizard - allows you to read quotes in the format you can define (including intraday)!
- Built-in Metastock(R) database importer - reads directly all symbols from your Metastock database (works with both EOD and intraday modes) in a matter of seconds!
- AmiQuote downloader program provides quick way of obtaining free end-of-day from major world exchanges (all US markets, LSE, ASX, Paris, Milan, Frankfurt)
- Free FOREX data downloadable via AmiQuote
- Free historical intraday delayed quotes from US exchanges downloadable via AmiQuote
- Script-driven, one-click automatic downloaders available for NYSE, Amex, Nasdaq, Australian Stock Exchange, Johannesburg Stock Exchange, Warsaw Stock Exchange

AmiBroker is successfully used in the following countries: USA, Canada, United Kingdom, Australia, Germany, Italy, Southern Africa, Poland, Holand, Norway, France, ...

For more information on data sources for AmiBroker click here.

Symbol & quotes database

AmiBroker features advanced database system that offers the following:

- **build-up and store historical tick or 5- or 15-second bar data for backtesting purposes (certain RT data sources only)**
- **build-up and store intraday minute-bar or end-of-day data for backtesting purposes**
- unlimited number of symbols and unlimited number of quotes
- multiple database support
- stores quotes, company information, financial results, categories, industry/sector information
- powerful filtering by sector, industry, group and market
- innovative symbol tree browser showing symbols grouped by sectors, industries, indexes
- automatic handling for composities (number and volumes of advancing, declining and unchanged symbols)
- automation support allowing you to control your database from external programs written in any language including Java Script, VBScript

AmiBroker Formula Language

The language

The AFL is an advanced formula language that allows you to create your own indicators, trading systems and commentaries. It is specialy designed for traders so writing analysis formulas is easier and quicker than in general-purpose languages.

AFL features more than 200 built-in AFL functions to use as a building blocks for your formulas. AFL includes trigonometric, averaging, statistical, data manipulation, conditional, pattern-detection and predefined indicator functions.

AFL supports unlimited variables, unlimited parentheses nesting, unlimited nested function calls and multiple logical operators. Version 4.40 brings completely rewritten engine with native flow-control and looping (if-else, while), user-defined functions and procedures with local and global variable scope.

New version 4.50 provides native multiple time-frame support, so you can mix different bar intervals in single formula.

Formula Editor / Drag-drop charting

Formula Editor allows you to quickly re-create any indicator/study found in the literature. Drag and drop charting allows to create complex overlays, indicators-on-indicators and more. Among other things it is possible to:

- any number of graphs that can be overlaid in the same chart pane
- modify built-in indicators
- custom or automatic scaling
- flexible grids
- access to composite data (number/volume of advancing, declining, unchanged issues)

Formula - based alerts

- Ability to write complex formula-based alerts that can be displayed on the screen, sent to you via e-mail, plus play a user-defined WAV file.
- Ability to run external applications via alerts - this allows automated trade execution

PORTFOLIO-LEVEL system back-testing, optimization, explorations and screening

Screening: Automatic analysis window enables you to scan your database for symbols matching your defined buy/sell rules. AmiBroker automaticaly produces the report telling you if buy/sell signals occurred on given symbol in the specified period of time.

Exploration: search your database for symbols matching your criteria and create the report showing the data you want to see: indicator values, past performance, etc. Then sort the results by any value listed.

Back-testing: AmiBroker can also perform full-featured back-testing of your trading strategy, giving you an idea about performance of your system.

The back-testing engine highlights:

- **PORTFOLIO LEVEL BACKTESTING/OPTIMIZATION**
- **Three-dimensional (3D), fully animated charts of optimization results**
- **Advanced custom backtester interface**
- **User-definable backtest metrics**
- **Different position sizing / money management techniques based on Portfolio Equity**
- **Hyper-fast execution - AmiBroker can backtest 10000 symbols (3000 data bars each) = 30 million data points in FIVE minutes!**
- **Integrated support for MULTIPLE time-frames in single formula**
- **NEW Report Explorer provides great way to organize/compare/view all backtest results**
- **Scanning/Exploration/Backtest/Optimization on Real Time data (tick and up) (RT version only)**
- **Scanning/Exploration/Backtest/Optimization on intraday data (1-min bars and up)**
- Back testing whole exchange or only limited, user-definable set matching your market, group, industry, sector selection
- **Equity curve plotting, Equity rainbows, composite equities curves**
- Test long, short or both long and short trades
- Maximum-loss stop, profit-target stop, trailing-stop, N-bar (time) stop
- Realistic back-testing
- Ability to control position size from your formula (Read more...)
- Create your own composites and scan/backtest them
- Detailed reporting giving you imporant statistics of your system.

Optimization: AmiBroker allows you to optimize your trading system with up to 10 optimization variables on single or **MULTIPLE securities** at once!

Automatic Chart Commentaries and Interpretation

- Full, textual descriptions of actual situation on the market
- automatic buy-sell arrows visible on the charts
- automatic textual interpretation of indicators and price chart (Window->Interpretation)

Scripting/COM/DLL support

- AFL engine allows embedding VBScript/JScript code within AFL formulas providing UNLIMITED possibilities

- ability to call external COM (ActiveX) objects from the AFL formula
- free SDK (software development kit) for registered users allowing writing indicator DLLs (plug-ins)
- many already available 3rd party plug-ins

Additional features

Portfolio manager

Built-in portfolio manager helps you track your investments. It allows you to registed buy/sell transactions, calculates brokerage commission, dividend (with setable dividend tax), cash deposits/withdrawals. You get the instant calculation of your equity value, percentage and point yield.

Scripting support

AmiBroker features automation interface that exposes objects and methods that could be accessed from any programming language including scripting dialects such as JScript (JavaScript) and VBScript. The scripting capabilities of AmiBroker allows you to automate time consuming database management tasks. Using scripting you will be able to create automatic downloaders, maintenace tools, exporters customized to your specific needs.

Internet integration

AmiBroker features built-in web browser that allows you to quickly view company profiles. The profile viewer is completely configurable so you can set it up for your particular exchange. The settings are market based so you can access different web sites for each market automatically. No longer will you be forced to waste your time browsing manually to get the latest news and symbol related information.

Configurability

AmiBroker is designed to be configurable and customizable in almost every area. It is not tied to particular exchange or data provider. Thanks to flexible import methods and scripting you will be able to adopt it easily to your favourite market(s). Also technical analysis tools built in into AmiBroker allow you to change every parameter with easy, and if you want even more, you can create your own indicators using flexible formula language.

Getting started

Hardware requirements

To run AmiBroker you need PC-Compatible computer meeting following minimum requirements

- Pentium 450 MHz or higher
- 256 MB RAM
- 20 MB hard disk space
- 256 color graphics card (high color recommended) 800x600 minimum screen resolution

Recommended machine configuration

- CPU: 1GHz or more, multiple cores
- 4 GB RAM or more

Supported operating systems

AmiBroker works on the following operating systems:

- **Windows 11 (any edition) 32-bit**
- **Windows 11 (any edition) 64-bit**
- **Windows 10 (any edition) 32-bit**
- **Windows 10 (any edition) 64-bit**
- **Windows 8 (any edition) 32-bit**
- **Windows 8 (any edition) 64-bit**
- **Windows 7 (any edition) 32-bit**
- **Windows 7 (any edition) 64-bit**
- Windows **Vista** (any edition) 32-bit
- Windows **Vista** (any edition) 64-bit
- Windows Server 2012 (any edition) 32-bit
- Windows Server 2012 (any edition) 64-bit
- Windows Server 2008 R2 (any edition) 64-bit
- Windows Server 2008 (any edition) 32-bit
- Windows Server 2008 (any edition) 32-bit
- **Windows XP** (any edition)
- Windows XP x64 (64-bit)
- Windows **2000** (any edition)

For more information about OS compatibility see: 32-bit/64-bit version compatibility chart

Installation and running

Install AmiBroker using it's setup program - it is available for download from
http://www.amibroker.com/download.html. After downloading double click on the program's icon. This will

launch the setup program - you can safely accept all default values by clicking "Next" on each page and "Install" on the last page. By default AmiBroker is installed to "C:\Program Files\AmiBroker" directory and this location is referred to as "main AmiBroker directory".

If setup program asks you to restart machine please do so to allow to replace system components.

After installation, you can start AmiBroker from Windows' standard Start->Programs->AmiBroker->AmiBroker menu.

Just after starting AmiBroker splash window shows up, then for few seconds AmiBroker loads its quotation database. Next the main AmiBroker screen appears.

AmiBroker main screen with price chart,
MACD and RSI indicators and profile view open. (Windows version)

In default setup you can see the toolbar, workspace window with symbol list on the left side and chart windows on the right side.

The toolbar provides fast access to the most often used program functions. With the symbol list view you can select active symbol. Changing the selection will cause chart redraw and update in some information windows if they are open. The chart windows let you to analyse current price trends and the behaviour of technical indicators.

You can quit AmiBroker using the *File/Exit* menu item.

Getting help

AmiBroker features new **context-senstive help system**, available by pressing **F1** key anywhere in the program.

When you press **F1** key while any window and any menu is shown, AmiBroker opens up a relevant help file page describing the window or menu in question. No more searching through the help file.

In addition to using F1 context-sensitive help it is **highly recommended** to read ALL Tutorial articles first. The answers to most common problems are given there. In case of major problem check Troubleshooting guide. Also there is a "Search" tab on the left of this on-line help window that allows to quickly locate information by keyword(s). Just type word(s) you are looking for and click "Display".

In case of further questions/problems you may check the following resources:

- AmiBroker web page - which is searchable using "Search" box in the top left corner of the page. The page gives you an access to:
    - AmiBroker Tips newsletter containing valuable step-by-step instructions on using various aspects of AmiBroker
    - Support area - featuring additional documentation
    - Frequently Asked Questions - the list of most commonly asked questions with the answers
    - AFL Library - featuring ready-to-use AFL formulas for custom indicators, commentaries and trading systems
    - Members area - featuring material accessible by registered users only
- AmiBroker Forum the place where you can meet other AmiBroker users, ask questions and share with ideas (with searchable archive).

Checking these places first will help me focusing on developing new features in AmiBroker. In case of problems not covered in above resources please please use customer support page at http://www.amibroker.com/support.html

AmiBroker 32-bit vs 64-bit Compatibility Chart

SUMMARY

The following table clearly shows that AmiBroker Professional 32 bit **runs on EVERY Windows version (BOTH 32 and 64-bit)** and with every data plugin. 64-bit version of AmiBroker runs solely on 64-bit versions of Windows and only with limited number of data sources due to lack of data vendors' API support for 64-bit technology.

	AmiBroker Standard AmiBroker Professional 32-bit	**AmiBroker Professional 64-bit**
Operating systems		
Windows 11 32-bit	**Yes**	No
Windows 11 64-bit	**Yes**	**Yes**
Windows 10 32-bit	**Yes**	No
Windows 10 64-bit	**Yes**	**Yes**
Windows 8 32-bit	**Yes**	No
Windows 8 64-bit	**Yes**	**Yes**
Windows 7 32-bit	**Yes**	No
Windows 7 64-bit	**Yes**	**Yes**
Windows **Vista**	**Yes**	No
Windows **Vista** x64	**Yes**	**Yes**
Windows **Server** 2008-2022 (32-bit)	**Yes**	No
Windows **Server** 2008-2022 x64 (64-bit)	**Yes**	**Yes**
Windows XP	**Yes**	No
Windows XP x64 (64-bit)	**Yes**	**Yes**
Windows 2000	**Yes**	No
Windows NT 4	**No**[4)]	No
Windows Millenium Edition (ME)	**No**[4)]	No
Windows 98	**No**[4)]	No
Windows 95	**No**[4)]	No

Addressable memory space		
32-bit Operating System	**2 GB or 3 GB**[3)]	**N/A**
64-bit Operating System	**4 GB**	**1000 GB**

Data sources		
AmiQuote (Yahoo, MSN, Google Finance)	**Yes**	**Yes**
Metastock import	**Yes**	**Yes**
Metastock plugin	**Yes**	**Yes**
eSignal RT	**Yes**	**Yes**
IQFeed RT	**Yes**	**Yes**
Interactive Brokers	**Yes**	**Yes**
Premium Data (via Metastock plugin)	**Yes**	**Yes**
TC2000/TCNet	**Yes**	No[2)]
FastTrack	**Yes**	No[2)]
DDE	**Yes**	**Yes**
ODBC database	**Yes**	**Yes**
any ASCII file (via import)	**Yes**	**Yes**

Remarks:

[1)] 64-bit native version of this plugin possible and is under development
[2)] 64-bit native version of this plugin is not technically possible because of lack of 64-bit API from data vendor at the moment
[3)] 3GB addressable memory is only possible with /3G switch in 32-bit Windows BOOT.INI file

[4)] Windows 95, 98, Me, NT4 are supported only by old AmiBroker versions 5.xx

What's new in the latest version?

Highlights of version 6.90

- **AFL editor improvements**
    - Multi-cursor Navigation: Enhanced editing with multi-cursor support, multiple selection editing, and split window editing.
    - Information Tooltips showing description and parameters of each function as you type them
    - Inline "profiler/checker" Messages: Added to show which lines/functions cause looking into the future.
    - better HiDPI scaling
    - Synthetic Data for Debugging: When a symbol has no quotes, synthetic data is created to allow code verification and debugging.
    - when errors are detected the "Explain" button is added to the caption bar that displays help page with error explanation
    - ChatGPT Code Detection: Added a notice to detect and inform users about potentially invalid ChatGPT-generated code.
    - UI Improvements: Fixed typo in auto-complete dropdown, fold margin width scales with font/DPI, and watch window displays small arrays entirely.
- **New Web Research browser**
    - brand new, standard compliant Chromium / Edge (WebView2) browser is now available as an alternative to old browser

      To use new web browser engine within AmiBroker you have to use 64-bit AmiBroker and do the following:

      1. Download x64 WebView2 component from Microsoft
      https://developer.microsoft.com/en-us/microsoft-edge/webview2/

      2. In AmiBroker go to Tools->Preferences, "Miscellaneous" tab and turn ON the option "Use Chromium/Edge (WebView2) for Web Research"

      3. Restart AmiBroker
- **AFL new features and improvements**
    - added support for maps / dictionaries (key-value pairs) - MapCreate function
    - added support for C++ raw strings and Python-style raw strings
    - added AddToComposite automatic normalization via atcFlagNormalize
    - added TrimResultRows function
    - added SparseInterpolate function
    - added GetOption("BHSymbol")
    - enhanced Nz, SafeDivide, Matrix, StaticVarInfo, SumSince
    - added static variable declarations with custom initial value (instead of Null)
    - stricter checking of printf/StrFormat formatting string
    - one-shot auto save of persistent static variables via SetOption("StaticVarAutoSave", -1)
    - Matrix Initialization: New parameter increment allows creation of matrices with monotonically increasing elements.
    - Memory and Performance Enhancements: Increased micro-allocator block size to 64 bytes and improved handling of static variables.
    - Error Handling: New error codes, stricter checking for format strings, and enhanced warnings for invalid characters.
    - Gui Control Functions*: Added extra 'style' parameter for all Gui* functions.

- **Charting improvements / new features**
    - Text tool allows to specify font style / size
    - increased limit of minimum X grid spacing for HiDPI displays
    - lots of QuickGFX fixes, improvements and speed-ups
- **Analysis window improvements / new features**
    - added information when equity gained from interest is greater than from trades alone
    - added information on how many times liquidity limit was hit
    - added support for "Show current trade arrows" after EXPLORATION run
    - New Analysis Info: Summary of symbols with filtered-out quotes due to padding displayed in the INFO tab.
    - Individual Optimization: Fixed crashes and sharing violations when GenerateReports was turned on.
    - Timestamp Resolution: Increased resolution for report folder timestamps using a unique counter.
    - Category Management: Filter dialog shows instructions if no symbols meet the selected criteria and shows categories with empty names.
- **Report Explorer**
    - Performance Improvements: Re-reads/refreshes are 10x faster, and the display performance is improved by 20x using a hyper-fast list view implementation.
    - Error Handling: Sharing violation messages are quietly displayed in the status bar instead of a modal blocking dialog box.

- **Database maintenance / data feed improvements**
    - Edit->Delete range supports multiple symbol selection, so you can delete range of quotes from multiple symbols at once
    - IQFeed plugin now has option to consolidate unbundled ticks and symbol lookup in the context menu
    - ASCII Import: Improved performance by early removal of duplicate lines in large files.
- **UI improvements**
    - modal dialog boxes flash and move to where mouse position is to prevent user confusion when modal dialog box opened on other monitor blocks the program from continuing
    - High DPI Support: Various UI elements, such as flyout buttons, modal dialog boxes, property grid expand buttons, and XAML markup units, now properly scale with High DPI displays.
    - added the Start page to improve new-user orientation
    - added pause button to Batch scheduler
    - Dark Mode Improvements: Enhanced readability of read-only items in the Account Summary page in dark mode.
    - Symbol Bar: Added context menu key support (VK_APPS) in addition to right-click.
    - Category Dialog: Prevents setting category names to empty, reverting to default name if attempted.
- **Other fixes**
    - Parameter file does not get corrupted when parameters mistakenly used new lines
    - Indicator Maintenance Wizard: Automatically creates timestamped backup files for easy restoration.
    - Easy Alerts: Now display comment field content in Alert Output window.

Highlights of version 6.40

Version 6.40 brings lots of new functionality especially with regards to the formula language and performance . There are hundreds of new features and changes to existing functionality as compared to version 6.30, listed in detail in "Release Notes" document in AmiBroker directory. Below is just a short list of few of them:

- **QuickGFX - brand new chart rendering engine** delivering upto 100x performance boost as compared to GDI rendering. GDI rendering speed also improved upto 2x.

  In-house developed QuickGFX direct rendering technology completely bypasses Windows GDI and offers amazing 10x-100x performance gain as compared to GDI rendering

  To enable QuickGFX experimental tech go to Tools->Preferences, "Miscellaneous" tab, click "Experimental: Use QuickGFX render" and press "Apply" or "OK"
  You can see when it is enabled, chart timing footnote (when enabled) will say "QuickGFX render ...ms" instead of "GDI render ...ms"

  This is PROFESSIONAL edition feature only. Users of Standard version would need to purchase upgrade in order to use this feature.
- **Automatic Analysis improvements**
  - Easy Sequencing of multiple Analysis actions via #pragma sequence(scan,exploration) and new "Run Sequence" button without need to write batches. Allows for example single click two-step analysis runs creating composites or static variables in first step and using them in second step
  - Backtester - added "Max. position value" option in the settings allowing to specify maximum dollar value of position. Zero (0) means no maximum. Positions larger will be shinked (if shrinking is enabled) or won't be entered at all (if shrinking is disabled)
  - huge speedups in running large explorations, dark theme and 5x speed up for rendering huge list views like (as compared to 6.31)
- **AmiBroker Formula Language new functions** / features
  - FindIndex(array, value, start_from = 0, dir = 1) - find index of array item matching specified value
  - BarsSinceCompare(past, comparison, current) - for every bar, compare past array values with current bar value and return the number of bars since last time comparision was true
  - InternetSetHeaders("headers") - set custom HTTP headers for subsequent web requests
  - InternetSetOption(option, value) - set HTTP option for internet session
  - InternetPostRequest("http://url_to_your_web_resource", data, flags = 0) - send HTTP Post request to Internet web resource (URL)
  - MxCopy() - in-place copy of rectangular blocks from one matrix to the other (copy portions of one matrix to the other matrix)
  - _exit() - gracefully ends AFL execution at the point of the call
  - InternetGetStatusCode() - returns HTTP status code of last InternetOpenURL or InternetPostRequest call
  - Chr(code) returns string representing single character of given ascii code
  - GetObject(path, class) providing functionality equivalent to JScript GetObject and VBScript GetObject
  - GuiSendKeyEvents("ED") - register given characters to be sent as Gui event when keyboard key is pressed. GuiGetEvent will return code == notifyKeyDown and id of given character, for example id == 'D' for D letter
  - inverf(x) - inverse of erf function
  - erf(x) - computes Error function https://en.wikipedia.org/wiki/Error_function
  - SafeDivide(x, y, valueifzerodiv)- safe division that handles division by zero using special handling (replace result with user-defined value)
  - new Javascript engine (Chakra) featuring native JSON support (EnableScript("chakra"))
- **AmiBroker Formula Language improvements, enhancements and fixes**
  - exponentation operator is made 50-100x faster in cases of small integer exponents of 2, 3, 4 and 5
  - GuiSetText() function avoids sending change notifications to prevent update loops

- ♦ ADX function vectorized (2x faster than before)
- ♦ added extra parameter for Error() function to stop execution
- ♦ internal function signatures changed to allow flexible ordering of arguments regardless of their type and varargs with less overhead
- ♦ GfxDrawImage with PNG images exclusive file lock removed
- ♦ parser warns if empty body is used in 'for' or 'while' statements
- ♦ support for gzip and deflate compression in Internet functions
- ♦ plugin interface backward compatibility with changing function signatures
- ♦ more runtime checks to prevent user errors and other fixes

- **Batch window new features / improvements**
  - ♦ Clipboard Cut/Copy/Paste implemented in batch editor
  - ♦ Edit->Delete (from main menu) and Edit-Delete All implemented for completeness
  - ♦ list view uses virtual mode now (owner data)
  - ♦ added "add results to watchlist" action / WatchlistAddResults
  - ♦ added "clear watchlist" action / WatchlistClear
  - ♦ added "comment" action
  - ♦ added optional parameter to Data Import ASCII command to allow specify format definition file
  - ♦ added optional parameter to Execute and Wait command to specify current working directory for command
  - ♦ added optional parameter to Export to File / Export walk-forward to file to specify column separator in CSV files
    -optional parameter defines the column SEPARATOR used for export. Only single character is used. When it is not supplied, comma is used.
- **AFL editor enhancements**
  - ♦ call tips (parameter information tooltips) now provide extra description about the function and its parameters (as of v6.39 only 20 functions have this extra info)
  - ♦ debugging session is automatically terminated with appropriate notice when user attempts to edit the code during debugging (this saves mouse clicks that were needed to stop debug session in order to edit)
  - ♦ previously when no text was selected and Prettify Code was choosen, the message box caused main frame to get focus instead of focus staying in AFL frame. Fixed
  - ♦ when an runtime error is detected debugging session the message bar shows now number of detected errors (as it did previously during normal "verify"
  - ♦ when dark mode list views are used watch window default text color was black making it hardly visible, changed to white

- **User Interface new features and improvements:**
  - ♦ Dark theme for all owner-draw list views (NOT in dialogs) now feature customizable theme (currently available "system (light) theme" and "black theme") - go to Tools->Customize, "Appearance" tab, "Dark mode for listviews" checkbox
  - ♦ 5x speed up for rendering huge list views (bypassing Windows rendering)
  - ♦ Preferences: added Text Tool font setting independent from Axis font (Preferences->Miscellaneous page)
  - ♦ Charts: improved text tool with per-study selectable font size
  - ♦ Parameter names are not truncated in HighDPI screens
- **Database improvements**
  - ♦ new 8-digit ICB structure implemented: https://www.ftserussell.com/data/industry-classification-benchmark-icb
  - ♦ Database Purify is 10x faster
  - ♦ OLE: added Stock.Quotations.Adjust function to perform adjustments programmatically OLE: added Stock.Quotations.Adjust() function. long Adjust(BSTR pszFieldList, float fMultiplier,

float fOffset, DATE dDateTime, boolean bBefore)

Highlights of version 6.30

Version 6.30 brings lots of new functionality especially with regards to the formula language and performance . There are hundreds of new features and changes to existing functionality as compared to version 6.20, listed in detail in "Release Notes" document in AmiBroker directory. Below is just a short list of few of them:

- **HIGHLIGHT: User-definable on-chart GUI controls**

  18 new AFL functions were added to allow creation of user-definable on-chart graphical user interfaces.

  ◊ **GuiButton** - create on-chart button control (AFL 4.30)
  ◊ **GuiCheckBox** - creates on-chart checkbox control (AFL 4.30)
  ◊ **GuiDateTime** - creates on-chart date-time picker control (AFL 4.30)
  ◊ **GuiEdit** - create on-chart edit control (AFL 4.30)
  ◊ **GuiEnable** - enables or disables on-chart control (AFL 4.30)
  ◊ **GuiGetCheck** - get checked state of control (AFL 4.30)
  ◊ **GuiGetEvent** - get GUI event (AFL 4.30)
  ◊ **GuiGetText** - get text from on-chart control (AFL 4.30)
  ◊ **GuiGetValue** - get numeric value of on-chart control (AFL 4.30)
  ◊ **GuiRadio** - creates on-chart radio button control (AFL 4.30)
  ◊ **GuiSetCheck** - set checked state of on-chart control (AFL 4.30)
  ◊ **GuiSetFont** - set the font for on-chart control (AFL 4.30)
  ◊ **GuiSetRange** - set slider control range (AFL 4.30)
  ◊ **GuiSetText** - set text value of on-chart control (AFL 4.30)
  ◊ **GuiSetValue** - set numeric value of on-chart control (AFL 4.30)
  ◊ **GuiSetVisible** - shows or hides on-chart control (AFL 4.30)
  ◊ **GuiSlider** - creates on-chart slider control (AFL 4.30)
  ◊ **GuiToggle** - create on-chart toggle button control (AFL 4.30)

- **New features in AmiBroker Formula Language**

  - static variable declaration
  - passing variables by reference
  - new voice functions
    ◊ **VoiceSetRate** - sets voice speech rate (AFL 4.30)
    ◊ **VoiceSetVolume** - set the volume of speech (AFL 4.30)
    ◊ **VoiceWaitUntilDone** - waits until TTS voice has finished speaking (AFL 4.30)
  - system error handling via GetLastOSError
  - support for geometric pens in GfxSelectPen
- **Huge performance improvements in 64-bit version with migration to new VC++2017 compiler**

  Why do we migrate to new compiler with 64-bit version?
  - New compiler supports new CPU instructions (SSE3/AVX) that we can use to offer better performance
  - According to our tests new compiler support produces faster code by itself (better optimizations, auto-vectorization, etc)

  - ♦ New compiler is better with error checking (less bugs to slip through)
  - ♦ We don't need to care about compatibility with pre-Vista systems in 64-bits version and all 64-bit capable CPUs are "modern" enough.

  Why do we stay with old compiler in 32-bit version?

  - ♦ New compiler does not produce code compatible with older operating systems (XP or earlier). Old compiler offers 100% compatibility with all Windows versions
  - ♦ New compiler requires modern CPUs

  Exact performance improvement is function dependent and hardware dependent. Many functions are faster by 30-50% but in some cases such as Min()/Max() functions as large as 8x speed up can be observed in 64-bit version.

- **Other key improvements**
  - ♦ Auto-optimization framework
  - ♦ HTML5 compatibility in Web Research window
  - ♦ comment folding in the AFL editor
  - ♦ clickable links in Analysis result list

Highlights of version 6.20

Version 6.20 brings lots of new functionality especially with regards to system testing. There are hundreds of new features and changes to existing functionality as compared to version 6.10, listed in detail in "Release Notes" document in AmiBroker directory. Below is just a short list of few of them:

- **HIGHLIGHT: Integrated Batch Processor** - allowing you to automate complex, sequential tasks, featuring
  - ♦ easy-to-use interface
  - ♦ built-in time scheduler
  - ♦ command line interface
  - ♦ AFL triggering
  - ♦ logging
- **New features in AmiBroker Formula Language**:
  - ♦ direct internet access functions (IntenetOpenURL/InternetReadString/InternetClose) allow interfacing with web / REST services from AFL
  - ♦ general purpose distribution function: PriceVolDisribution
  - ♦ more statistical functions: NormDist/Kurtosis/Skewness
  - ♦ variable period PercentRank and StDev
  - ♦ cumulative product functions CumProd/Prod/ProdSince
  - ♦ bitmap image drawing function GfxDrawImage
  - ♦ ability to select Speech API voice via VoiceSelect/VoiceCount
  - ♦ multi-text exploration columns via AddMultiTextColumn
- **Performance improvements**
  - ♦ faster startup
  - ♦ more responsive Analysis window while it is busy processing (plus new Pause function)
- **UI and usability improvements**
  - ♦ many adjustments in UI scaling to better support HiDPI/4K displays
  - ♦ walk-forward settings are now per-project, not global
  - ♦ debugger's watch window supports display of static variables and expressions with static variables

Highlights of version 6.10

Version 6.10 brings lots of new functionality especially with regards to system testing. There are hundreds of new features and changes to existing functionality as compared to version 6.00, listed in detail in "Release Notes" document in AmiBroker directory. Below is just a short list of few of them:

- **HIGHLIGHT: Integrated Visual Debugger** - featuring
    - single-step execution
    - breakpoints
    - user-definable watches with expression evaluator
    - array view with per-item change highlighting
    - value inspection tooltips
    - output window (for printf output)
- **HIGHLIGHT: Enhanced Matrix support**
    - matrix inversion (via Gauss-Jordan elimination)
    - solving linear equation systems (example polynomial fit code included)
    - determinant
    - fast block access
    - matrix column/row sorting
    - converting matrix from/to string
    - persistency (saving/loading matrices from/to disk via static variable mechanism)
- **New features in the AFL Editor:**
    - Bookmarks
    - Find in Files
    - Block comment/uncomment
    - Clickable links to external documentation/files in comments
- **New features in AmiBroker Formula Language**
    - faster composites by means of StaticVarAdd
    - compression of persistent static variables (can save as much as 90% of disk space and memory)
    - stricter error checking (printf/StrFormat format string checking)
    - static and dynamic variable functions now support matrices
    - new functions GfxFillSolidRect, SumSince, MxInverse, MxSolve, MxDet, MxSetBlock, MxGetBlock, MxSort, MxSortRows, MxToString, MxFromString, StaticVarAdd, fgetcwd
    - new parameters in functions: DateTimeToStr, StaticVarSet
    - performance improvements in printf(), variable period Sum
- **Charting, User interface and under-the-hood improvements**
    - enhanced category window (allowing to rearrange move up/down markets/groups/sectors/industries and watch lists)
    - user-definable candlestick wick thickness
    - user-definable decimal places in horizontal line price level
    - cycle tool handles years past 2038
    - large icons are used on HighDPI displays for tree views and list view checkboxes for much better accessability
    - auto-size Analysis columns to contents
    - crash recovery improvements
    - tons of other improvements (see Release Notes for details)

Highlights of version 6.00

Version 6.00 brings lots of new functionality especially with regards to system testing. There are hundreds of new features and changes to existing functionality as compared to version 5.90, listed in detail in "Release Notes" document in AmiBroker directory. Below is just a short list of few of them:

- Integrated high-performance **Monte Carlo simulator** - with cumulative distribution charts of equity, max. drawdowns, support for custom user-definable metrics and ability to peform MC simulator driven optimizations.
- Full **Matrix support** (two dimensional arrays) in AFL with direct native matrix arithmetic (matrix operations like addition, subtraction, multiplication, division, transpose, etc), see Matrix, MxIdentity, MxTranspose, MxGetSize
- Detailed **Buy-and-hold (benchmark) statistics** automatically added to the backtest reports
- User **definable stop precedence** (SetStopPrecedence function) and **stop validity** (ValidFrom/ValidTo parameters in ApplyStop function)
- **Sparse array** support: SparseCompress, SparseExpand
- **Infinite Impulse Response** filter function (IIR) for efficient implementation of higher order smoothing algorithms
- **Raw text output in explorations** via AddRow function
- New styles supported by Exploration XYCharts
- **Variable period Percentile** function
- **Unicode (UCN) support** in PlotText, PlotTextSetFont, GfxDrawText, GfxTextOut, chart titles, interpretations and commentary windows (allows various graphic annotations / windings)
- **New Low level graphic functions**: GfxSelectHatchBrush, GfxSelectStockObject
- wildcard matching function StrMatch
- enhanced Assignment Organizer
- **Word-wrap** functionality in AFL editor and **enhanced "Code Prettify"** function

Highlights of version 5.90

In addition to completely **new functionality** this version focuses on **speed improvements** and enhancements of existing functionality. There are hundreds of new features and changes to existing functionality as compared to version 5.80, listed in detail in "Release Notes" document in AmiBroker directory. Below is just a short list of few of them:

- **Performance improvements**
  - ♦ **AFL Engine**: custom memory allocator does not use Microsoft runtime lib for reference tracking anymore. Result - **complex formulas** with lots of loops and OLE (especially low-level custom backtests) **run upto 3 times faster in 32 bit and 4 times faster in 64-bit**
  - ♦ execution **speed improved by factor > 2x** for AFL functions: MACD, Signal, CCI, Sum (variable period)
- **Brand new Code Snippets window and keyboard triggers**
  - ♦ added **Code Snippets** window - allows inserting/deleting/saving selected parts of the formula as snippets. Also implemented is convenient drag-drop of snippet to the formula edit window
  - ♦ Code snippets are available in auto complete list (type @ plus first letter of snippet key trigger), and even without auto complete activated @keytrigger is replaced by snippet text
- **Re-designed Report Explorer and improved Report Viewer (HTMLView)**
  - ♦ Column layout (order and sizes) is now saved and restored between runs
  - ♦ Loading and refresh performance significantly improved (5x) using owner draw/ virtual mode
  - ♦ Multi-column sorting implemented
  - ♦ Numeric columns are now right aligned for better readability
  - ♦ visuals significantly improved (list uses modern style, grid lines, immediate column resizing, double buffering for no flicker, thousand separators, negative values are displayed in dark red, HighDPI aware, changed toolbar)
  - ♦ HTMLView - Backtest report viewer - added Edit/Copy, Edit/Select All and Edit/Copy TABLE. The last command transforms HTML tables into CSV format and copies it into clipboard so tables can be pasted easily to Excel. Also it divides Entry/Exit columns into separate Entry/exit date/price columns

- new **Bid/Ask trend indicator** in **Real-time quote window** - a graphical indicator showing the direction of 10 most recent changes in real-time bid/ask prices The right-most box is most recent and as new bid/ask quotes arrive they are shifted to the left side.
- **User-definable HTML backtest reports**
    - now it is possible to output HTML instead of graphics in report chart formulas using AFL: EnableTextOutput(3) - HTML output to backtest report
    - rewritten *3. Profit Table.afl* using HTML embedding features auto-scalable layout (so it enlarges when numbers are bigger), bold summary columns, negative values in red, boundary date changed to last day of year/month
- **Charting improvements**
    - Left/right extended Trend lines and Rays now use user-definable Extension Factor (new field in Study properties) instead of always infinite extent. Ext. Factor equal to ZERO means INFINITE, other values 0.1 ... 26 define how far to the left/right line is extended
    - Max zoom achievable via View->Zoom Out is increased to 5 million bars, also Pref/Charting/Default zoom limit set to 5 million.
    - Line drawings now have user definable line width in pixels (new "Line width" field in Study Properties dialog). In addition to that "Thick line" box makes line twice as wide (so actual width of thick line is 2 * lineWidth instead of adding 1 pixel to width)
    - added ability to control number of decimals in chart value labes via GraphLabelDecimals variable (example, adding GraphLabelDecimals = 2; to the formula would give you value lables with 2 decimal places)
- **User Interface improvements**
    - Parameter window look and feel improved. Item height is increased and slider thumb made wider for easier use on small size/high DPI screen
    - New Analysis UI refreshes faster
    - Colors, bold and italic styles are now added to Interpretation and Commentary windows
    - Filter dialog now shows number of matching symbols in real-time
    - Column setup dialog has new Mark All / Toggle All buttons

- **AFL new features / improvements:**
    - **new** AFL functions: GetFormulaPath, NullCount, Sort, Reverse, StrSort, StrTrim, SendEmail
    - extended functionality of AFL functions: StrExtract, StrMid, RestorePriceArrays, PlotGrid, EnableTextOutput, GetOption
    - single-characters literals added to AFL
- **Stability & debug improvements**
    - added lots of parameter checks
    - 64-bit version has now call stack trace in the bug report for better debugging
    - added more memory checks, early warnings and error messages when running out of memory
    - added checks for unusual, yet potentially 'troublemaker' scenarios

Highlights of version 5.80

In addition to completely new functionality this version focuses on incremental improvements and enhancements of existing functionality. There are hundreds of new features and changes to existing functionality as compared to version 5.70, listed in detail in "Release Notes" document in AmiBroker directory. Below is just a short list of few of them:

- **Brand-new completely rewritten AFL Formula Editor** that supports the following features:
    - Improved Syntax highlighting
    - Automatic brace matching/highlighting (NEW)
    - Auto indentation (NEW)

- Indentation markers (NEW)
- Enhanced auto-complete in two modes (immediate (NEW) and on-demand)
- Parameter information
- Line numbering margin and selection margin (NEW)
- Code folding (NEW)
- In-line Error Reporting (NEW)
- New tabbed user interface with ability to work in both MDI and separate floating frame mode, can be moved behind main AmiBroker screen and brought back (Window->Toggle Frame) (NEW) or kept on top (Window->Keep on top)
- Rectangular block copy/paste/delete (Use mouse and hold down left **Alt** key to mark rectangular block)
- Enhanced printing (with syntax highlighting and header/footer)

- **Code snippets** - these are small pieces of re-usable AFL code. They can be inserted by right-clicking in the AFL editor window and choosing "Insert Snippet" menu. Code snippets are user-definable.
- **New features in Low-level graphics**
    - multiple Z-order layers GfxSetZOrder
    - co-ordinates can now be given in both pixels and bar-price mode GfxSetCoordsMode
    - speed improvements (upto 3x)
- **Persistent Static variables -** StaticVarSet/StaticVarSetText (added 'persistent' parameter)
- **Analysis (Backtest/Optimize) enhancements**
    - new "Trade using FX cash conversion" setting
    - 64-bit SPSO/Tribes engine fixes
- **New/enhanced AFL functions**
    - PlotTextSetFont (NEW)
    - GfxSetCoordsMode (NEW)
    - GfxSetZOrder (NEW)
    - GfxGetTextWidth (NEW)
    - fopen (added 'shared' parameter)
    - StaticVarSet/StaticVarSetText (added 'persistent' parameter)
    - SetOption (new option "StaticVarAutoSave")
    - SetChartOptions (new flags chartDisableYAxisCursor, chartDisableTooltips)
    - PlotText (new parameter yoffset)

Highlights of version 5.70

In addition to completely new functionality this version focuses on incremental improvements and enhancements of existing functionality. There are 116 new features and changes to existing functionality as compared to version 5.60, listed in detail in "Release Notes" document in AmiBroker directory. Below is just a short list of few of them:

- **Analysis improvements:**
    - **New Multi-threaded Individual Optimization**
    - **New general-purpose ranking functions** StaticVarGenerateRanks/StaticVarGetRankedSymbols
    - User-definable ranking columns (via AddRankColumn function)
    - Lots of internal speedups in backtesting/optimization engine
    - SPSO, Tribes optimization engines now available also in 64-bit

- **Time&Sales improvements**: user-definable filtering, user-definable colors, 2 user-selectable display modes
- **Database improvements**:
    - 64-bit version supports files larger than 2GB per symbol

- ♦ in-memory cache can hold upto 100 000 symbols (up from 20K)
- ♦ new 64-bit DDE and ODBC plugins

- **Charting improvements:**
  - ♦ Greatly improved performance QuickData technology implemented lowering CPU usage for charts
  - ♦ Edit->'Paste Special' allows to copy-paste entire chart pane with various options
  - ♦ Distance measuring when drawing trendlines (X,Y distance in the status bar)
  - ♦ X/Y constrains for drawing tools (press X and/or Y key to constrain movement in either X or Y direction when drawing)
  - ♦ ASCII importer adds support for millisecond timestamps
- **AFL improvements**:
  - ♦ new functions:
    - ◊ StaticVarGenerateRanks, StaticVarGetRankedSymbols - general-purpose user-definable ranking
    - ◊ Error - display user-definable error messages (also useful for plugin developers)
    - ◊ fdir - directory listing
    - ◊ CategoryCreate - programmatic creation of watch-lists
    - ◊ AddRankColumn - ranking columns in exploration
  - ♦ performance improved for Percentile() (order(s) of magnitude)
  - ♦ new fields supported in GetFnData
  - ♦ XShift support added to PlotShapes
  - ♦ speeded up transcendental mathfunctions (sqrt, sin,asin, cos, acos, tan, atan, ln, log10, etc)
  - ♦ improved SetSortColumns

Highlights of version 5.60

- **Multithreaded GDI (graphics) rendering** - now all drawing (graphic rendering) is done in separate worker threads so the user interface is way more responsive and charts are updated faster and completely independently from each other.
- **Automatic Walk-Forward out-of-sample summary report -** each out-of-sample step produces individual report now, plus there is a new **summary report** that covers all out-of-sample steps. It is visible in the Report Explorer as last one and has "PS" type.
- **Enhanced color-coded backtest report**

- **XY (scatter) charts in explorations**
- **Chart themes** and improved chart look (esp. the grid)
- **One-click automatic setup and update of stocks listing, sector and industry assignments** for all major US exchanges
- **Unlimited ad-hoc chart intervals** by means of new Interval combo box that accepts any interval typed manually
- **support for ICB** (Industry Classification Benchmark) categories in AFL, UI, ASCII importer and OLE interface
- native **Gradient area charts**
- **super-thick lines** in Plot, PlotOHLC, PlotForeign
- **new AFL functions**: GetAsyncKeyState, InIcb, IcbID, StaticVarInfo, SetGradientFill, XYChartAddPoint, XYChartSetAxis
- **updated AFL functions** with new functionality: Status, CategoryGetSymbols, CategoryGetName, CategorySetName, CategoryAddSymbol, CategoryRemoveSymbol, CategoryFind, Plot, PlotOHLC, PlotForeign
- new 64-bit eSignal plugin

- updated UI in many places
- many other improvements (see Release Notes for details)

Highlights of version 5.50

- **New Analysis window** introduced in version 5.50 brings the following improvements over old Automatic Analysis
    - **multi-threaded operation = speed -** new Analysis window uses all available CPUs/cores to execute formulas in many threads in parallel providing significant speed ups. For example on 4 core Intel i7 that can run upto 8 threads, it can run upto 8 times faster than old Analysis window. Exact speed up depends on complexity of the formula (the more complex it is, the more speedup is possible), amount of data processed (RAM access may be not as fast as CPU thus limiting possible speed gains).
    - **non-blocking operation** - you can now view, scroll and sort results of analysis while they are still generated, also as user interface thread is not used for processing for most part, charts and other GUI-driven program parts are way more responsive than with old automatic analysis
    - **multiple instances** - you can run more than one instance of New Analysis at a time, so you can run many scans/backtest/explorations/optimizations in parallel without waiting for one to complete
    - **slicker user interface -** New Analysis window can act as tabbed document, can be floated, buttons can be re-arranged for better workflow. There is way more space for the result list, extra information about execution is provided on the new "Info" tab. Also walk-forward results are now displayed within New Analysis window for less clutter.
- Mini **High-Low rank chart** in Real Time quote window
- User-definable **mini bar charts in Explorations** (see AddColumn function)
- **Add Rank Column** feature - right-click Analysis result list and choose "Add Rank column" - it adds a column with ordinal rankings based on current sort or just row number column when list is not sorted
- IRA account backtesting via SettlementDelay feature (see SetOption function)
- **Range bars** algorithm improved significantly
- **new AFL functions**: ThreadSleep, StaticVarCompareExchange
- **updated AFL functions** with new functionality: AddColumn, SetOption, GetOption, CategoryGetSymbols, PopupWindow, GetFnData, ClipboardSet
- updated OLE interface to support new Analysis window
- updated custom backtester interface to support access to local, per-analysis EquityArray property
- updated UI in many places
- Owner-draw list views for 10x speed improvement when displaying millions of rows
- many other improvements (see Release Notes for details)

Highlights of version 5.40

- **Fully Multi-threaded charting. Massively parallel AFL execution** (each chart pane runs in separate thread) allows to maximize speed and utilisation of modern multi-core / multi-CPU computers. For example on 8-core Intel i7 CPU your charts will run upto 8 times faster than in version 5.30. **The AFL engine has been completely rewritten** from ground up to allow multiple instances of the engine running simultaneously. This enables not only multithreading but also enhances responsiveness of entire application, as even badly-written user formula used in a chart is not able to lock or slow the rest of the program. **Multi-threading is ON by default.** It can be turned off by unchecking "Multi-threaded charts" box in Tools->Preferences, "AFL" tab but it is strongly discouraged. Multi-threading should be ON if you want AmiBroker to operate at full speed.
- **12 new AFL functions**
    - DateTimeAdd - adds specified number of seconds/minutes/hours/days to datetime

    - HMA - Hull Moving average
    - FIR - Finite Impulse Response filter
    - PercentRank - calculate percent rank
    - Lookup - search the array for bar with specified date/time
    - FirstVisibleValue - get first visible value of the array
    - LastVisibleValue - get last visible value of the array
    - InGICS - check if given symbol belongs to specified GICS category
    - GicsID - get information about GICS category
    - PlaySound - play .WAV sound file
    - ShellExecute - execute external program / file
    - _DT - synonym of StrToDateTime
- Quote Editor improvements and fixes: allows user to turn on/off time shift and editing timestamps down to milliseconds, fixed handling of 12 hour (AM/PM) regional setting
- Charting improvements: better looking value labels, low-level gfx functions speeded up 4 times.
- Charting-related changes
    - Data Window and data tooltip readout is immediate and does not require extra AFL execution (values required to display them are stored in RAM during normal chart refresh and available without need to re-run the formula). Tooltip variable is now obsolete. To display custom values in tooltips without plotting a line you can use Plot() with styleHidden flag.
    - Interpreation display does not require AFL execution
    - Inserting indicator and resetting parameters are orders of magnitude faster
    - chart zoom setting is now saved in a layout file and restored when layout is loaded
- OLE interface improvements (new IsBusy method of Analysis object and Import method refreshes UI automatically)
- AFL engine improvements:
    - added warnings that detect potential user mistakes - such as assignment within conditional expression or redundant calls to Plot() function
    - added extra checks for invalid parameter values for many functions and array subscript == Null - appropriate error message is displayed
    - PlotText optimized to conserve memory and reduce execution time by skipping invisible parts
- Account manager fixes
- improved compatibility with Windows 7 (high-DPI aware manifest, compatibility with Internet Explorer 9)
- 64bit-specific fixes (including fixing problems with 3rd party DLLs)
- new appearance themes and many other improvements and fixes

Highlights of version 5.30

- changed database format to support time stamp granularity down to one microsecond (0.000001s) and more data fields
- static **array** variables
- **user-definable backtest report charts** (see examples in Charts window, "Report Charts" folder)
- new Data Window (Window->Data Window)
- new Performance Monitor tool
- tick statistics added to Time&Sales window
- chart blank area extension using **END** key (on the keyboard), to restore original setting press **HOME** key
- added option to require variable declarations (SetOption("RequireDeclarations", True));
- persistent column state (widths/order/visibility) in the Automatic Analysis and all other list-views.
- gradient area charts capability (see Charts - Basic Chart - Gradient Price chart)
- **new typeof() AFL operator**
- **new AFL functions**

    - ColorBlend
    - DateTimeDiff
    - HighestVisibleValue
    - LowestVisibleValue
    - StaticVarCount
- **User-definable Z-order** of drawings and indicator plots
- optional data padding for non-trading days
- Rectangle and ellipse drawing tools are now solid by default
- **X-Y co-ordinate labels** added (use View->X-Y Labels menu to display/hide).
- Support for SSL (secure connection) and TCP/IP port selection for e-mail alerts added
- new Symbols window with ultra quick full-text search and sorting
- support for GICS 4-level category system
- placing orders directly from chart (Interactive Brokers)
- many other improvements and fixes (see Release Notes document for details)

Highlights of version 5.20

- **Smart (non-exhaustive) trading system optimization**

  AmiBroker now ships with 3 non-exhaustive, evolutionary optimization algorithms:
  SPSO (Standard Particle Swarm Optimizer)
  TRIBES (Advances Particle Swarm)
  CMA-ES (Covariance Matrix Adaptation Evolutionary Strategy
- **Support for market-neutral, long-short balanced strategies** via MaxOpenLong/MaxOpenShort control and separate long/short rankings in the backtester
- **Performance optimizations in chart drawing engine -** charts are orders of magnitude faster when number of bars displayed is much greater than number of pixels.
- **Log window** implemented - allow tracing and run-time error reporting
- **QuickAFL implemented in the Automatic Analysis -** speeds up backtests, optimization and explorations by factor of 2 or more (if range is less than all quotations). (Note: in order to enable it you need to check "Use QuickAFL" box in the Automatic Analysis setttings).
- **Multiple-segment Volume-At-Price charts** (via PlotVAPOverlayA function)
- 32-bit AmiBroker is now LARGEADDRESSAWARE, i.e. can now use upto 4GB of RAM
- **Built-in Quarterly and Yearly intervals**

- **Automatic summary rows in the explorations** (via AddSummaryRows AFL function)
- **Charting enhancements and improvements**
    - better handling of drawing tools
    - better magnet mode
    - Fibonacci timezones now include lines 144 and 233
    - zooming via scroll bar improved
- Range bars now use per-symbol TickSize as a unit
- **new AFL functions**:
  GetChartBkColor
  CategorySetName
  PlotVAPOverlayA
  AddSummaryRows
  DaysSince1900
  OptimizerSetEngine
  OptimizerSetOption
  StrCount

- AFL performance improvements in LinearReg, LinRegSlope, LinRegIntercept, TSF and StdErr, Day(), Month(), Year(), DaysSince1900(), DayOfWeek(), DayOfYear() functions (order of magnitude faster)
- Improved AFL functions: queued Say() command (text-to-speech), improved StrExtract() - can now refer to items counting from the end
- real-time data plugins updated (IB version 1.2.4, eSignal version 1.9.0), IBController updated to support latest changes in data sources

Highlights of version 5.10

- Automatic **Walk-Forward testing** (trading system optimization and validation technique)

- **Floating windows** (TRUE multi-monitor charting capability)
  ability to "undock" (or "float") the chart window and move it to separate monitor.
  All layout code is also updated to correctly save and restore multi-monitor chart setups
  http://www.amibroker.com/video/FloatAndLink.html

- **Symbol and Interval linking**
  multiple charts can now be linked by symbol and/or by interval using easy-to-use color-coded links

- **AFL Code Profiler** - shows code analysis with detailed per-function timing report (AFL Editor: **Tools->Code Check & Profile** menu)

- **Real-time quote window** improvements
  - re-ordering of symbols in the RT quote using drag-and-drop
  - direct type-in symbols into RT quote window
  - ability to separate groups of symbols by inserting empty line
  - faster refresh and multi-stage background color fading on quote change

- **new/improved AFL functions**
  - (new) SetBarFillColor
  - (improved) GetCursorXPosition
  - (improved) GetCursorYPosition
  - (improved) GetCursorMouseButtons
  - (improved) SetChartOptions
  - (improved) SetOption
  - (improved) Status

- Improved speed of backtesting/optimization (up to 2x in some cases as compared to v5.00)

- improved chart crosshairs - no flicker, work faster and can be switched on/off globally

- track more foreign markets: now you can define rates for up to 20 currencies (different than base currency) for multiple currency backtesting in the preferences window.

- new backtester modes: backtestRegularRaw2 and backtestRegularRaw2Multi

- new FindSignal method of backtester object

- 3D optimization chart animation is now smoother (100fps)
- unlimited nesting of #include and #include_once statements
- Improved scaling of semi-log charts, MDI tab order saved in the layout, improved bug reporting, high resoltion Vista icon added, other fixes and improvements

Highlights of version 5.00

- New Watchlist system featuring:
    - unlimited number of watch lists
    - lists keep original order in which symbols were added (still can be sorted alphabetically on-demand)
    - new AFL function to refer to watch lists by name
- Support for **AFL Code Wizard -** brand new automatic formula creation program for people without any programming experience. For more information about AFL Code wizard see this introductory video: http://www.amibroker.com/video/amiwiz/AFLWiz1.html
- AFL engine enhancements
    - new flow control statements: switch /case / break / continue
    - new compound assignment operators: +=, -=, *=, /=, %=, &=, |=
    - new functions: GetPlaybackDateTime(), PopupWindow(), Mersene Twister Random Number Generator mtRandom(), and others
- New dedicated memory heap allocators for quotes and trading system signals resulting in ability to run much longer optimizations than ever without getting out-of-memory messages
- Two new backtester modes (available using SetBacktestMode function) allowing handling of unfiltered (raw) entry signals
- User-definable 5-tier commission schedule in the backtest (Automatic Analysis / Settings)
- Chart template sharing
  now you can save the chart as "Chart Template, Complete (*.chart)" that stores all layout AND referenced formulas in SINGLE file that can be sent to your friend and entire chart will be restored on any computer with ease, without need to copy individual formulas.
- New-Look charts - divider lines between panes are now single pixel and no borders around charts giving cleaner, larger and more readable chart display and printout
- Custom Range Bars (supported in the charts and via TimeFrameSet())
- New Low-level graphics interface (23 new AFL functions)
- HTML Import in Automatic Analysis
- Full screen Anti-Aliasing in 3D optimization chart viewer (beautifully smooth 3D charts and improved readability)
- Enhanced Real-Time Quote window display (faster updates, dual-color change marks)
- Control of Time Shift in the ASCII importer

Detailed Change Log

CHANGES FOR VERSION 6.10.0 (as compared to 6.09.0)

- SetOption new fields for Monte Carlo
  MCUseEquityChanges - use equity changes instead of trade list

MCChartEquityScale - 1 for log scale, 0 for linear scale
MCLogScaleFinalEquity - 1 for log scale, 0 for linear scale
MCLogScaleDrawdown - 1 for log scale, 0 for linear scale
MCNegativeDrawdown - 1 - use negative numbers for drawdown (reverse drawdown CDF)

- QuickData cache was not flushed when double clicking on Analysis result list to display chart. Fixed.
- MonteCarlo: added an option to use negative number for drawdowns (on by default). This reverses the ordering of "drawdown" column in the MC table and reverses the meaning (i.e. 10% percentile value means that there is 10% chance of drawdowns being equal or worse (more negative) than presented value). Wwith this option turned off (as in old versions), drawdowns are reported as numbers greater than zero and 10% percentile value means 10% chance of drawdowns being equal or better (smaller) than presented amount.
- AFL: Gfx: low-level graphic recorded on layer 128 was played prematurely. Fixed.
- Added error message when user attempts to write value to BarCount (read-only symbol)
- More documentation updates

CHANGES FOR VERSION 6.09.0 (as compared to 6.08.0)

- AFL: new function fgetcwd - get current working directory
- AFL Editor: an error message box displayed when @link file was not found caused main frame to get focus. Fixed.
- AFL Editor: auto-complete and parameter tooltips poped up when adding/editing Doxygen/JavaDoc comment sections. Fixed.
- Updated documentation

CHANGES FOR VERSION 6.08.0 (as compared to 6.07.0)

- AFL Editor: Implemented clickable JavaDoc/Doxygen-style links in doc comments
  To use links in comments you have to put @link command followed by path to file or URL inside JavaDoc/Doxygen style comment:
  a) multiline comment that begins with
  ```
  /**
  @link readme.html
  @link http://www.amibroker.com
  */
  ```
  (note double asterisk after initial slash)
  b) single line comment that begins with triple slash
  ```
  /// @link readme.html
  /// @link c:\program files\amibroker\readme.html
  /// @link http://www.amibroker.com
  ```

  Now when you hover the mouse over @link command you will see the underline that indicates it is clickable.
  It reacts to DOUBLE CLICK (not single click). When you double click it linked document will be open

  @link command can open web pages, local files (both relative and absolute paths are supported) with Windows-registered program
  to open given file type. So if you use
  ```
  /// @link something.doc
  ```

  then MS word would be used.

  ```
  /// @link test.xls
  ```

would open test.xls in Excel.

Relative paths refer to AmiBroker working directory.
Html files are open with default browser, txt files are usually open with Notepad (or whatever application you use).

If file does not exist then you will get an error message.

- AFL: Unary minus and NOT operator could cause crash when applied to result of undefined function or variable of non-numeric type. Fixed.
- 64-bit stack walker does not use Microsoft symbol server anymore so it can init a lot faster. Also initialization of stack walker is moved to startup sequence because doing it after exception is risky and unreliable
- 64-bit: access violations during Analysis run do not result in 'application not responding' freeze. Instead bug report is displayed with proper call stack.
- After change in 6.02 exception dialog displayed first error in the formula even if exception was really caused not by first but last error. Fixed.
- Exception info could get mixed up when multiple threads generated them at the same time. Fixed.
- New Analysis: Auto-size column functionality re-displayed hidden columns. Fixed.
- UI: AmiBroker now prevents piling on "Bug recovery" dialogs when multiple exceptions are thrown from multiple threads. First dialog is displayed, other exceptions are logged into DebugView as long as "Bug recovery" is not dismissed.
- "Rename" and backup functions in several places could fail if new file name already existed. Fixed.
- Support for Windows 95 is dropped

CHANGES FOR VERSION 6.07.0 (as compared to 6.06.0)

- AFL: compression for array Static variables implemented to save memory and file size: StaticVarSet has additional parameter that controls compression
  compressMode parameter decides whenever given variable will be compressed or not.
  By default only persistent static variables will be compressed (cmDefault).
  You can turn it off completely compressionMode = cmNever, or turn it on for persitent and non-persistent variables using
  compressionMode = cmAlways

  Compression is done by removing repeated values from the sequence as repeated values are restored when doing StaticVarGet.
  Compression is NOT compatible with non-aligned mode of StaticVarGet.
  If compressed array is retrieved by StaticVarGet with align=False, then repeated values found in original array would not be retrieved.

  Turning compression on slows down StaticVarSet (as it needs to do some extra processing), but does not affect performance of other functions,
  so StaticVarGet is equally fast with or without compression.
- AFL: printf/StrFormat check for %s, %c, %d, %x, %i, %u, %x, %p sequences that are not supported now and prints Error 62 when they are found.
- AFL: printf/StrFormat now implement a check for correct formatting string as sometimes users passed strings with % that is special marker for formatting string instead of %% to print actual percent sign

  When check failes, "Error 61. The number of % formatting specifier(s) does not match the number of arguments passed." is displayed
- AFL: Some variable period functions such as Ref/Sum/MA incorrectly accepted matrix as a 'period'. Fixed (proper error message is displayed now).

- AFL: StaticVarGet/StaticVarSet functions now support matrices including persistency (so matrices can be written to disk and loaded back between AmiBroker runs)
- AFL: VarSet/VarGet officially accept matrices (allow dynamic variables of matrix type). (FWIW they were accepted in previous versions but implementation was incomplete.)
- Charting: Candlestick wicks thickness is now user definable as % of candle width with a max. pixel width (Tools->Preferences, "Bars and Candles" tab). Also by default wick thickness is now 25%/5px max instead of constant 1 pixel.
- Charting: Cycles tool in weekly chart produced incorrect lines when it went past 2038. Fixed.
- Charting: Horizontal line tool level is now displayed with number of decimals as defined in Preferences (Misc tab, "decimal places in chart titles/tools"). Note that this setting is used when you draw a new line or modify old.
- Debugger: Added F5 key as keyboard shortcut for "Debug / Go"

CHANGES FOR VERSION 6.06.0 (as compared to 6.05.0)

- AFL Editor: Bookmarks implemented (Ctrl+F2 - toggle bookmark, F2 - go to next bookmark, Shift+F2 - go to previous bookmark
- AFL Editor: Find in Files - when file name contained braces () double click on it would not open it. Fixed.
- AFL Editor: Line comment puts double // comments on lines that appear after empty line. Fixed. Now empty lines are left untouched.
- AFL: extra parameter for DateTimeToStr function mode (mode = 0 - convert both date and time portion, 1 - only date, 2 - only time) Note that mode 2 would give you empty string when applied on chart with daily or longer interval
- AFL: new function GfxFillSolidRect(x1, y1, x2, y2, color) - it is fastest method to fill solid rectangle with single color (faster than GfxRectangle)
  pw = Status("pxwidth");
  ph = Status("pxheight");

  GfxFillSolidRect(0, 0, pw, ph, colorBlack);
- AFL: new function SumSince - a fast sum of array elements since condition was true, works like Cum(array) - ValueWhen(condition, Cum(array)) or Sum(array, BarsSince(condition), but much faster
  Syntax: SumSince(condition, array)
- AFL: Variable period Sum() performance improved significantly when period changes +/-1 on bar by bar basis
- Categories dialog: added ability to re-arrange the order of markets/groups/sectors/industries and watch lists using "Move Up" / "Move down" buttons
  This is non-trivial task as all symbols must be synchronized with the change as when the ordering of categories change then all symbols data must be re-indexed to reflect new order as symbols refer to oridinal position of category.
- New Analysis: Added an option "Auto-size columns to fit content" - OFF by default. When turned on, column widths are adjusted to fit the content after analysis is complete. Note that this means that any widths specified in width parameter in AddColumn width will be ignored when this is turned ON.
- New Analysis: when "Wait for backfill" was turned ON and some symbols in the "Apply to" list were wrong (non-existing) then Analysis attempted to run code for symbol without quotes causing trouble. Fixed.

CHANGES FOR VERSION 6.05.0 (as compared to 6.04.0)

- AFL Editor Find in Files implemented - on the output list you can double click on line to open the file in the editor

- AFL Editor/Debugger: Code Check and Profile stopped working when it was run after debugging session. Fixed.
- AFL: Null is accepted by MxFromString now
- AFL: performance of printf() for large volume output (>60000 characters) improved ~60 times.
- Charts: reduced tearing when resizing the chart window
- Debugger now can use either base time interval or current chart interval (Tools->Preferences, Debugger tab, "Bar interval" setting)
- Debugger: implemented Output window (for printf() output during debugging)
- Removed old-style legacy AFL editor from the code, removed "use new editor" checkbox from the preferences. A new editor is used always now.
- UI: Unfold triangles in tree views and checkboxes in Layers list were hard to click on HighDPI screens because icons were too small. Fixed. Now larger icons (32x32) are used instead of small (16x16) on small tablets like 8'' Windows with righ resolution (such as 1920x1200) that have very high DPI (approx 200DPI).
- Watch window: now expressions with 2D subscripts to access elements of matrix matrix[x][y] are supported

CHANGES FOR VERSION 6.04.0 (as compared to 6.03.0)

- AFL Editor & Watch Window: the contents of matrices is now displayed in value tooltips and watch window
- Debugger state (watch variables/expressions and breakpoint locations) is now saved between AFL editor sessions (formula-wise so each formula has its own "debug state"). The data are saved in XML file with .dbg extension
- Debugger: MDI mode implemented. In 6.0.3 AFL Editor did not support 'MDI mode' (even crashed), now this functionality is back. Also Watch window is available in MDI mode and debugger is working in MDI mode too.
- Preferences: added new page with Debugger settings.
  The settings are as follows:
  + Limit BarCount to - defines maximum number of bars in arrays (BarCount) used during debugging
  + Auto-scroll to first changed item - when this is ON, the "Arrays" list in the Watch window is scrolled automatically to first array item that has changed (so you don't need to locate elementsm that changed manually)
  + Keep debugging state - when this is ON, AFL editor saves the debug state (breakpoints and watches) in the .dbg file along with the formula when closing the editor and restores the state when formula is reopened.
- Preferences: 'Editor' tab moved so 'AFL', 'Editor', and 'Debugger' tabs are next to each other, also 'Alerts' and 'Currencies' tabs moved before 'Miscellaneous'
- Watch window: a new tab "Arrays" shows exploration-like array output for detailed examination of array contents (first 20 arrays from watch window are reported)
- Watch window: implemented array item change highlighting and auto-scrolling so first changed item is always visible in the 'Arrays' tab
- Watch window: implemented expression evaluator, so you can not only display variables, but also expressions involving variables such as (high+low)/2 or individual array elements such as myvariable[i] where i is dynamic loop counter
- Watch window: implemented value change highlighting: changed values are displayed with light yellow background, additionally numeric (scalar) values are displayed in green when they increased or red if they are decreased
- Watch window: pressing DELETE key while editing variable name caused deletion of variable from watch. Fixed.
- Watch window: variables can now be drag-dropped from AFL editor window

CHANGES FOR VERSION 6.03.0 (as compared to 6.02.0)

- AFL Editor: Implemented brand new fully integrated Visual AFL Debugger
  NOTES: this the work-in-progress more features will be added in later betas
- Debugger: Implemented breakpoints (including those set before compilation as well as adding/removing breakpoints during debugging)

  To add/remove breakpoint use red circle toolbar button or press F9
  Breakpoints can be added and removed at any time (during editing, when debugger is in 'running' state or stopped at breakpoint)
  Breakpoints are implemented statement-wise (as single-stepping). Keep in mind that there can be only one breakpoint in any single line so if line has more than one statements
  like this:
  x=1;y=1;
  the breakpoint will trigger before first statement in this line.

  Breakpoints currently work with:
  a) regular statements (that end with semicolon). For multi-line statements place breakpoint at the beginning line of the statement
  b) for loops
  c) while loops
  d) do-while loops (you need to place breakpoint where 'while' clause is located, it won't break at the 'do' line as it essentially is no-op, if you want to break at the beginning
  of do, just place breakpoint on first statement inside { block }
  d) if statements
  e) return statements
  f) switch/case statements
  g) break statements

  Breakpoints that you place on other lines, won't trigger. The AFL editor won't allow to place breakpoint on empty line, or line that beginnins with // comment or sole brace
- Debugger: Implemented single-stepping Step Into, Step Over

  Single-stepping is done statement-wise. So it works on single statement at a time. So for example, empty lines are skipped and statements spanning multiple rows like below are treated as one step.

  ```
  x = "test" +
  " second row" +
  " third row";
  ```

  But if you put two statements in single line like this:
  x = 1; y = 2;

  Then this line would be treated as two steps (each expression x =1; and y=2; separately).

  Keyboard shortcuts:
  Step Over F10
  Step Into F11
- Debugger: Implemented variable value inspection tooltips (hover mouse over variable to see its value)
- Debugger: Implemented Watch window with variable value display automatically updating when debugger single-steps or reaches breakpoint

To add variable to watch window simply double click on the list (at the end to add new variable). You can also change variable name by double clicking on existing item
- Renaming of PersistVars.temp to PersistVars.bin failed on some versions of OS. Fixed.
- RT Quote window: drag-drop list view shows small arrow marker for the drop point now.
- RT Quote window: when moving items using drag-drop, moved items image was flickering. Fixed
- UI: Chart zoom via Ctrl+mouse wheel now works so it attempts to current mouse position as a "center" point of zoom when possible. (Previously it always worked so right border was the 'center').
- AFL: Sum(array,N) outputs NULL values for indices starting from 0 upto index N-1 instead of N
- When intraday data were used timestamp of 00:00:00 was not displayed in data tooltip (empty field was shown). Fixed.

CHANGES FOR VERSION 6.02.0 (as compared to 6.01.0)

- AFL Editor: Implemented Line comment/uncomment feature (Edit->Line Comment, Ctrl+Q) to automatically add/remove // line comments from single or multiple lines. If multiple lines are selected, the content of the first selected line is deciding whenever block of lines is to be commented or uncommented
- AFL: If any matrix cell has Null value then matrix product operator @ produces Null respective row/column of the result matrix.
- AFL: if user called (a) GetPerfomanceCounter(1) then (b) GetPerformanceCounter(0) then subsequent call to GetPerformanceCounter(0) returned cumulated time not from call (a) but from system boot. Fixed.
- AFL: matrix identifier can now be used in if-else statement. Such condition checks whenever very first element of matrix [0][0] is NOT NULL. This is useful for checking output of functions like MxInverse/MxSolve that would return NULL in all cells if matrix is singular
  So

```
m = Matrix( 10, 10, 0 );

// do something with matrix
if( m )
{
// some code
}

is equivalent to

if( NOT IsNull( m[ 0 ][ 0 ] ) )
{
// some code
}
```

- AFL: MxSolve/MxInverse now return a matrix filled with Nulls as a result if source matrix can not be inverted and produce warning level 2 instead of an error.
- AFL: On Windows Vista and higher static variables use slim read-write (SRW) lock instead of critical section. This gives 5% performance increase in multithreading scenarios.
- AFL: StaticVarAdd("name", value, keepAll = True, persistent = False) - an atomic addition (interlocked read-add-write) operation for static variables

  It is multithreading safe addition for static variables that are shared by multiple threads. This function is atomic with respect to calls to other static variable functions.

KeepAll flag when it is set to true emulates the behavior of AddToComposite. It keeps all values that are already present, so if data holes exists in current symbol,
the bars that are present in static variable but not present in current symbol remain untouched.
When KeepAll is set to false then only bars that are present in current symbol are kept. Any other bars that were present in static variable but not present in currently
processed symbols are removed. That is what normally happens with StaticVarSet().

In fact when KeepAll is set to False, StaticVarAdd can be seen as the following pseudo code:

```
EnterCriticalSection
x = Nz( StaticVarGet( "name" ) ); // read exisiting value (and convert
Nulls to zero)
x += Nz( value ); // add value to existing
StaticVarSet( "name", x ); // store updated value
LeaveCriticalSection
```

The function can be used to create composites like this:

```
if( status("stocknum") == 0 )
{
    // remove any earier composite values
   StaticVarRemove("~Composite");
}

StaticVarAdd( "~Composite", MACD() > Signal() );
Buy = 0;
```

NOTES:
1. StaticVarAdd automatically converts all Nulls to zeros (as AddToComposite does).
2. If you want to replace AddToComposite with StaticVarAdd, keep in mind that by default AddToComposite skips symbols in group 253. This is done so composite symbols
are not added to themselves. If you have composite symbols in your database and want to skip symbols in group 253 you can use
if(GroupID() != 253) StaticVarAdd("~Composite", values);
3. Thanks to extensive code tuning, StaticVarAdd generally offers better performance than AddToComposite which was already blazing fast. Single threaded StaticVarAdd may be twice as fast as ATC. With 8 threads running StaticVarAdd may be 4x as fast (it does not scale as much as naive person may think, because critical section limits performance due to lock contention). To illustrate the amount of fine tuning applied it can be said that first 'straightforward' version of StaticVarAdd was actually 20 times slower than ATC.
4. Be careful when using "quickafl" as StaticVarAdd would not increase 'required bars' (as ATC does), so if you want to actually add all bars and quick afl is turned on in analysis, it is better to add SetBarsRequired(sbrAll, sbrAll)

- AFL: Study() returned NULL array when line's start date was greater than end date. Fixed (now it works for lines drawn from right to left too).
- Docs: Example polynomial fit formula shows how to gracefully handle singular matrix and overflow/not-a-numbers in polynomial calcs

```
order = Param( "n-th Order", 10, 1, 16, 1 );
length = 60;

lvb = BarCount - 1;
fvb = lvb - length;
```

```
yy = Matrix( length + 1, 1, 0 );
xx = Matrix( length + 1, order + 1, 1 );

yy = MxSetBlock( yy, 0, length, 0, 0, Ref( C, fvb ) );

x = BarIndex() - length/2;

for( j = 1; j <= order; j++ )
{
     xx = MxSetBlock( xx, 0, length, j, j, x ^ j );
}

xxt = MxTranspose( xx );
aa = MxSolve( xxt @ xx, xxt ) @ yy;
//aa = MxInverse( xxt @ xx ) @ xxt @ yy; // alternative way

if( aa ) // check if matrix is not null (so solution exists)
{
    rr = Null; // store the fit in rr
   for( i = fvb; i <= lvb; i++ )
    {
       rr[i] = aa[0][0];

       for( j = 1; j <= order; j++ )
       {
          rr[i] += aa[j][0] * x[ i - fvb ] ^ j;
       }
    }

     if( IsNan( rr[ fvb ] ) )
     {
       // our polynomial yields infinite or not-a-number result due to
overflow/underflow
      Title = "Polyfit failed. The order of polynomial is too High";
     }
     else
     {
       SetChartOptions( 0, chartShowDates );
       SetBarFillColor( IIf( C > O, ColorRGB( 0, 75, 0 ), IIf( C <= O,
ColorRGB( 75, 0, 0 ), colorLightGrey ) ) );
       Plot( rr, "rr", colorWhite, styleLine | styleThick);
    }
}
else
{
   Title = "Matrix is singular. The order of polynomial is too high";
}

Plot( C, "", IIf( C > O, ColorRGB( 0, 255, 0 ), IIf( C <= O, ColorRGB( 255,
0, 0 ), colorLightGrey ) ), styleDots | styleNoLine );
```

- Persistent static variables are now saved to PersistVars.temp and once write is successful the file is renamed to PersistVars.bin. This is to prevent data loss when writing very large sets of persistent variables.
- When more than one error is detected in single line of the formula then the first error message is displayed instead of last one in chart/commentary/analysis.

CHANGES FOR VERSION 6.01.0 (as compared to 6.00.0)

- AFL: MxDet(mx, method = 0) - calculates determinant of the matrix
  method = 0 - auto (use slow method for matrices of upto and including 5x5, fast for larger matrices)
  method = 1 - slow (slow, more accurate)
  method = 2 - fast (LU decomposition, less accurate)

  "slow" method uses Laplace expansion
  " fast" method uses LU decomposition
  " Slow" method for small matrices (1x1, 2x2, 3x3, 4x4) is actually faster than "fast", equally fast for matrix 5x5 and
  slower than "fast" method for matrices larger than 5x5

  For this reason "auto" method uses "fast" LU method only for matrices larger than 5x5

  LU decomposition is fast but subject to higher numerical errors. "Slow" method is slower yet produces much more reliable results.
  For example Octave/MatLab that use LU decomposition would say that determinant of singular matrix like this
  { {16, 2, 3, 13}, { 5, 11, 10, 8}, {9, 7, 6, 12}, {4, 14, 15, 1 } }
  is -1.4495e-012 due to roundoff errors of LU method.

  If you want to calculate determinant using fast (LU decomposition) method, call MxDet with fast parameter set to 2.

  CAVEAT: Laplace method has complexity of O(N!) and for this reason, even if you use method = 1, the maximum dimension for this method is limited to 10x10.
  Matrices larger than that are always calculated using LU method

- AFL: MxFromString() - creates a new matrix out of string in Mathematica/Wolfram list-style: "{ { 1, 2, 3 }, { 4, 5, 6 } }" or Matlab/Maple style "[[1, 2, 3], [4, 5, 6]]" or GNU Octave comma-semicolon style [1, 2, 3; 4, 5, 6]
- AFL: MxGetBlock(matrix, startrow, endrow, startcol, endcol, asArray = False)

  Retrieves items from rectangular submatrix (block) and returns either smaller matrix (when asArray is set to False)
  or "normal" AFL array (when asArray is set to True). If array has different number of bars, unused elements are filled with Null.

  ```
  z = Matrix( 2, 20, 0 );
  // first row
  z = MxSetBlock( z, 0, 0, 0, 19, Close );
  // second row
  z = MxSetBlock( z, 1, 1, 0, 19, RSI( 5 ) );
  printf("Matrix z\n");
  ```

```
printf( MxToString( z ) );

x = MxGetBlock( z, 0, 1, 0, 19, True );

printf("Items are now in regular array (data series):\n" );
for( i = 0; i < 20; i++ )
printf( NumToStr( x[ i ] ) + "\n" );

z = MxGetBlock( z, 0, 1, 0, 1 ); // retrieve upper 2x2 submatrix
printf("Upper submatrix z\n");
printf( MxToString( z ) );
```

- AFL: MxInverse(mx) - calculates inverse of the matrix (see comments to MxSolve for more info)

- AFL: MxSetBlock(matrix, startrow, endrow, startcol, endcol, values = 0)
  Sets values in the rectangular block of cells (rows in the range startrow..endrow and columns in the range startcol..endcol inclusive).
  This allows to fill entire or partial rows, columns and all other kind of rectangular areas in the matrix with user specified data
  Row and column numbers are zero based.
  If values parameter is scalar, all cells in specified block are filled with that value.
  If values parameter is an array, cells in the block are filled from left to right and from top to bottom with consecutive values taken from that array.
  If there are more cells in the block than values in the array, the array item counter wraps around to zero and starts taking values from the beginning

  Note: the function creates new matrix as a result (so source matrix is unaffected unless you do the assignment of the result back to the original variable)

  Example 1:
```
// Create a matrix 6x6
// and fill 4x4 interior (except edges with consecutively increasing numbers)

y = Matrix( 6, 6, 0 );
y = MxSetBlock( y, 1, 4, 1, 4, Cum(1));
printf("Matrix y\n");
printf( MxToString( y ) );
```

  Example 2:
```
// Create a matrix 2 rows x 20 columns and fill rows 0, 1 with first 20 values of Close and RSI(5)
arrays respectively

z = Matrix( 2, 20, 0 );
// first row
z = MxSetBlock( z, 0, 0, 0, 19, Close );
// second row
z = MxSetBlock( z, 1, 1, 0, 19, RSI( 5 ) );
printf("Matrix z\n");
printf( MxToString( z ) );
```

- AFL: MxSolve(A, B) - solves linear equation system A@X = B

  A needs to be square matrix NxN
  B has to have N rows and at least one column (vertical vector).

Then calling
X = MxSolve(A, B) would give vertical vector holding solution of the system of equations A @ X = B

B can also be a matrix,with each of its column representing different vector B. This way single call to MxSolve can solve several systems with same matrix A but different right hand vectors.
If B is a matrix NxM then MxSolve will produce result also having NxM cells with each column representing single solution.

Example 1:

```
A = MxFromString("[ 1, 1, 1, 1; 0, 2, 5, -1; 2, 5, -1, 1; 2, 2, 2, 1 ]");
B = MxFromString("[ 7; -5; 28; 13 ]" ); // single vertical vector B

printf( "Solving A * X = B\n" );
printf("Matrix A\n");
printf( MxToString( A ) );
printf("\nMatrix B\n");
printf( MxToString( B ) );

X = MxSolve( A, B );

printf("\nSolution X\n");
```

Example 2:

```
A = MxFromString("[ 1, 1, 1, 1; 0, 2, 5, -1; 2, 5, -1, 1; 2, 2, 2, 1 ]");
B = MxFromString("[ 7, 14 ; -5, -10; 28, 56; 13, 26 ]" ); // 2 right-hand side vertical vectors

printf( "Solving A * X = B\n" );
printf("Matrix A\n");
printf( MxToString( A ) );
printf("\nMatrix B\n");
printf( MxToString( B ) );

X = MxSolve( A, B );

printf("\nSolutions X\n");

printf( MxToString( X ) ); // two solutions
```

(Highly) Technical note about numerical precision:

Despite the fact that both MxSolve and MxInverse use double precision arithmetic solving/inverting matrices is subject to numerical precision of double IEEE
and for example zero result may come up as something like 1.4355e-16 (0.0000000000000001) due to the fact that double precision is still limited in accuracy (16 digits).

The result of
X = MxInverse(A) @ B;
although mathematically the same as solving the system of equations, would yield slightly different result because if you do the inverse the returned matrix is converted back
to single precision and matrix product is performed with single precision. When you use MxSolve you

are performing all calcs using 64-bit (double) precision and
only end result is converted back to single precision. So for example polynomial fit code works better with MxSolve than MxInverse

```
// Least Squares Polynomial Fit test

order = Param( "n-th Order", 15, 1, 25, 1 );
length = 60;

lvb = BarCount - 1;
fvb = lvb - length;

yy = Matrix( length + 1, 1, 0 );
xx = Matrix( length + 1, order + 1, 1 );

yy = MxSetBlock( yy, 0, length, 0, 0, Ref( C, fvb ) );

x = BarIndex() - length/2;

for( j = 1; j <= order; j++ )
{
     xx = MxSetBlock( xx, 0, length, j, j, x ^ j );
}

xxt = MxTranspose( xx );
aa = MxSolve( xxt @ xx, xxt ) @ yy;
//aa = MxInverse( xxt @ xx ) @ xxt @ yy; // alternative way

rr = Null; // store the fit in rr
for( i = fvb; i <= lvb; i++ )
{
     rr[i] = aa[0][0];

     for( j = 1; j <= order; j++ )
     {
         rr[i] += aa[j][0] * x[ i - fvb ] ^ j;
     }
}

SetChartOptions( 0, chartShowDates );
SetBarFillColor( IIf( C > O, ColorRGB( 0, 75, 0 ), IIf( C <= O, ColorRGB(
75, 0, 0 ), colorLightGrey ) ) );
Plot( rr, "rr", colorWhite, styleLine | styleThick);
Plot( C, "", IIf( C > O, ColorRGB( 0, 255, 0 ), IIf( C <= O, ColorRGB( 255,
0, 0 ), colorLightGrey ) ), styleDots | styleNoLine );
```

- AFL: MxSort(mx, dim = -1, ascening = True) - sorts the matrix

  Sorts all items in a matrix
  When dim == -1 (the default) it would sort:
  a) a row if there is only one row (vector is horizontal)
  b) a column if there is only one column (vector is vertical)

c) each column separately if there are more rows and columns than one (so we have actual 2D matrix).

When dim == 0 the function sorts the items in each row separately
When dim == 1 the function sorts the items in each column separately

```
// example
m = MxFromString("[ 9, 5, 6; 8, 7, 3 ]");

printf( MxToString( m ) + "\n\n" );

printf("%g, %g\n\n", MxGetSize( m, 0 ), MxGetSize( m, 1 ) );

m2 = MxSort( m, 0 ) ;

printf( MxToString( m2 ) + "\n\n" );

m3 = MxSort( m, 1 ) ;

printf( MxToString( m3 ) + "\n\n" );
```

- AFL: MxSortRows(mx, ascending = True, col1 = 0, col2 = -1, col3 = -1)

  Sorts the rows of the matrix in ascending/descending order of the col1 column. When the col1 column has equal values, SortRows sorts according to the col2 and col3 columns in succession (if col2 and col3 are specified and >= 0).
  Column numbers are zero based.

  Hint: if you want to sort columns instead you can Transpose/Sort rows/Transpose back.

```
m = MxFromString("[ 9, 1, 6; 40, 30, 20; 8, 7, 3; 3, 5, 1 ]");

printf("Input matrix\n");

printf( MxToString( m ) + "\n\n" );

printf("Rows %g, Cols %g\n\n", MxGetSize( m, 0 ), MxGetSize( m, 1 ) );

printf("Sorting every row separately\n");
m2 = MxSort( m, 0 ) ;

printf( MxToString( m2 ) + "\n\n" );

printf("Sorting every column separately\n");
m3 = MxSort( m, 1 ) ;

printf( MxToString( m3 )+ "\n\n");

printf("Sorting rows by contents of first column\n");
m4 = MxSortRows( m, True, 0 ) ;

printf(MxToString( m4 )+ "\n\n");
printf("Sorting rows by contents of second column\n");
m5 = MxSortRows( m, True, 1 ) ;

printf(MxToString( m5 )+ "\n\n");
```

- AFL: MxToString - creates string out of matrix variable in the Wolfram list style like this (for 3x3 matrix): { { x00, x01, x02 }, { x10, x11, x12 }, { x20, x21, x22 } }

CHANGES FOR EARLIER VERSIONS ARE DOCUMENTED IN RELEASE NOTES DOCUMENT THAT YOU CAN FIND IN AMIBROKER INSTALLATION FOLDER.

Tutorial

This chapter will guide you through the most important parts of AmiBroker.

Basic tasks:

- Basic operations

User interface topics:

- Beginners' charting guide
- How to use drag-and-drop charting interface
- Chart themes
- User interface customization
- Working with chart sheets and window layouts
- Working with layers
- Using Web Research
- Using Account Manager
- Using Fundamental data
- Using New Analysis window
- Using Batch window

Updating quotes:

- How to get quotes from various exchanges
- Setting up eSignal RT feed (RT version only)
- Setting up myTrack RT feed (RT version only)
- Setting up Quote Tracker as a RT data source
- Setting up IQFeed RT feed (RT version only)
- How to use AmiBroker in Real Time mode (RT version only)
- Using AmiBroker with other external data source (Quotes Plus, TC2000 / TCNet, Metastock, FastTrack)
- Automatic update of EOD quotes for US & Canada markets from Yahoo
- Using manual mode of AmiQuote downloader (Yahoo, MSN Money Central, Quote.com Livecharts)
- Using Metastock importer

Database management:

- Understanding database concepts
- Understanding categories
- Working with watch lists

AmiBroker Formula Language topics:

- Understanding how AFL language works
- Creating your own indicators
- Using graph styles and colors in the indicators
- How to create your own exploration
- How to write your own chart commentary
- Using studies in your AFL formulas
- Backtesting your trading ideas

- Portfolio backtesting
- Reading backtest report
- How to optimize a trading system (advanced)
- Walk-Forward testing (advanced)
- Backtesting futures (advanced)
- Pyramiding/scaling and multiple currencies in the portfolio backtester (advanced)
- Monte Carlo simulation of trading systems (advanced)
- Using formula-based alerts (advanced)
- Using interpretation window (advanced)
- Multiple time frame support (advanced)
- Efficient use of multithreading (advanced)
- Ranking functionality (advanced)
- How to use code snippets
- Using AFL Debugger
- Using on-chart GUI controls (advanced)

More information:

- Video Tutorials On-Line

Basic operations

Adding a new symbol

In order to add a new symbol into database you can use *Symbol->New* menu item or Add symbol toolbar button.

After selecting this function you will be prompted for new ticker symbol. The maximum ticker symbol length is 48 characters. For proper import functioning you should enter the symbol with CAPITALS.

Removing a symbol

In order to remove existing symbol from the database you can use Symbol->Remove menu item or Remove symbol toolbar button. After choosing this function you will be asked for confirmation of symbol removing. Note well that this operation can not be undone !!!

Removing multiple symbols at once is possible using Assignment organizer.

Splitting a stock

To perform stock split use *Symbol->Split* menu item or Split toolbar button.

AmiBroker provides easy way of handling stock splits. Program will try to guess split date and ratio by analyzing quotations. If there is just a single quotation after split this should work, if not you will be asked for split date and ratio. Note well that this operation can not be undone!!!

From version 2.0 and up the split function offers more functionality: you can use old-style ratio or you can specify a split using following expression:

x->y

which means that x shares before split become y after it. For example 2->3 means that 2 shares become 3 after the split. So ordinary split into five pieces will be 1->5.

As you have probably guessed it is possible now to perform reverse-split, for example 2->1, which means that 2 shares are joined together into 1 share.

Deleting quotation

To delete a quotation simply select the quote you want to delete by clicking on the chart (a vertical line will appear showing selected date and quote). Then choose *Edit->Delete quotation* menu option.

To delete quotations of all stocks from given day you should use *Edit->Delete session*.

You can also use Quote Editor to delete quotes.

Adding/removing symbol from favourites

To add the symbol to the favourites you should check favourite box in the Information window. To remove it from favourites simply uncheck that box. Alternatively you can click on the tree with the right mouse button and select "Add to favourites" and "Remove from favourites" options from the context menu.

Merging quotations of two symbols

It happens sometimes that the ticker for the symbol is changed then you may get two tickers in your database - one holding historical quotes and the second one holding newest quotes (after name change). In order to put all quotes to the single ticker you should use *Symbol->Merge* feature. You should just select the new ticker (after name change) and choose *Symbol->Merge*. Then from the combo you should choose original ticker ("merge with") and optionally check the following fields:

- overwrite duplicate quotes - checking this option will overwrite the quotes already existing in "new" ticker with those present in "old" ticker (this should really not be the case, but may happen).
- delete "merge with" afterwards - checking this option will delete the "old" ticker after merging
- assign alias name - checking this option will copy the "old" ticker to the alias field of the "new" ticker

Beginners' charting guide

Introduction

AmiBroker charting engine allows object-oriented manipulation of all drawings. Now you can simply move, resize, cut, copy, paste and delete all drawing objects with ease. This chapter will guide you though most important aspects of using charting tools.

Let's now take a look at the user interface:

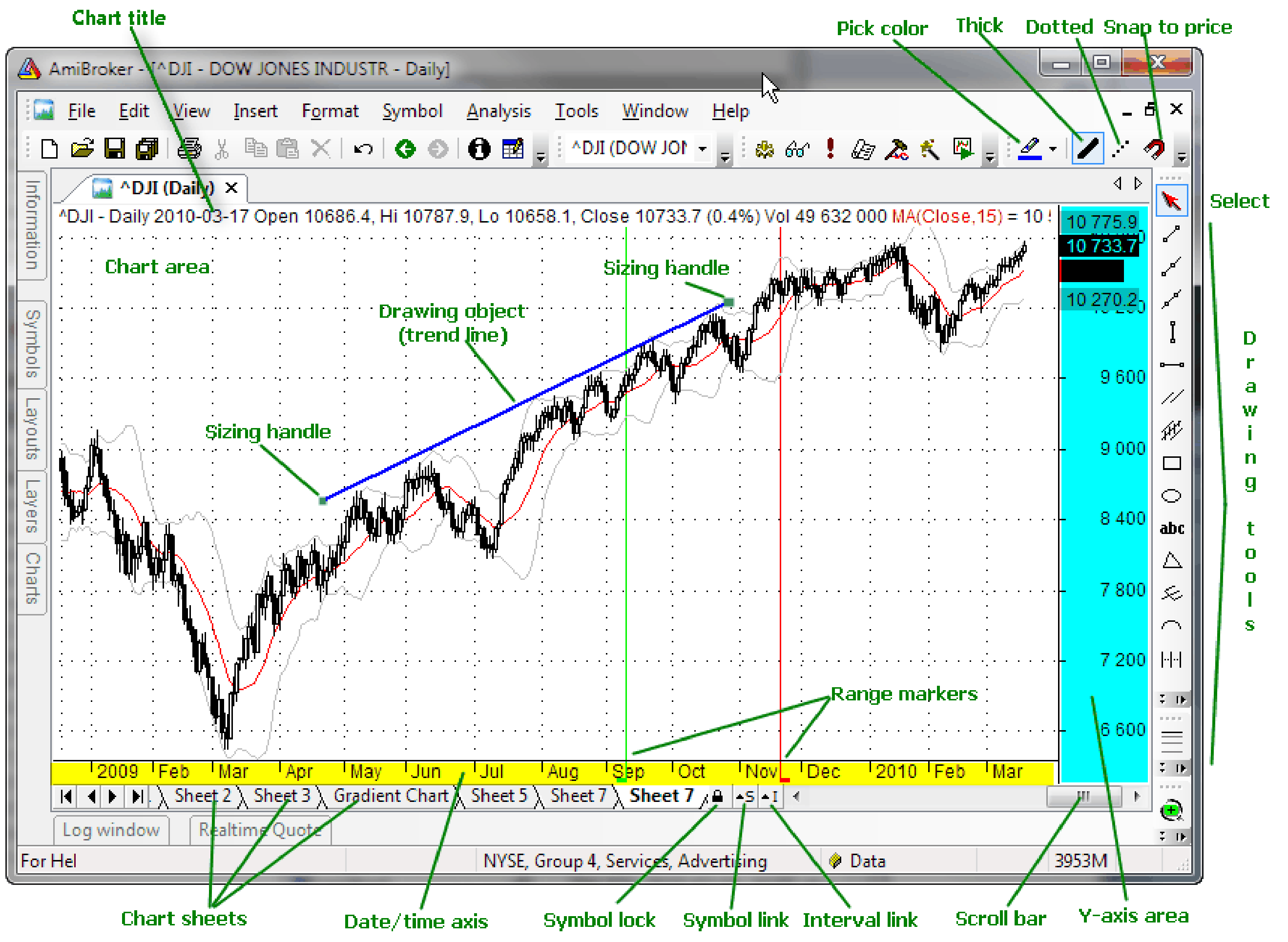


Chart title
Pick color
Thick
Dotted
Snap to price
Select
Drawing tools
Chart area
Sizing handle
Drawing object (trend line)
Sizing handle
Range markers
Chart sheets
Date/time axis
Symbol lock
Symbol link
Interval link
Scroll bar
Y-axis area
AmiBroker - [^DJI - DOW JONES INDUSTR - Daily]
File Edit View Insert Format Symbol Analysis Tools Window Help
^DJI (Daily)
^DJI - Daily 2010-03-17 Open 10686.4, Hi 10787.9, Lo 10658.1, Close 10733.7 (0.4%) Vol 49 632 000 MA(Close,15) = 10
Sheet 2 Sheet 3 Gradient Chart Sheet 5 Sheet 7 Sheet 7
Log window Realtime Quote
NYSE, Group 4, Services, Advertising
Data
3953M

As you can see the in the center we have chart area in which price chart with moving average and Bollinger bands is plotted (**you can control the apperance of built-in charts** from **Tools->Preferences** window).

In the bottom of the chart you can see date axis (marked with red color), and below scroll bar and chart sheets tab control. Scroll bar can be used to display past quotes, while sheet tab allows to view different chart pages/sheets (click here to learn more about chart sheets).

To the right you can see Y-axis area (marked with blue color) that shows Y-scale and value labels. Value labels are color fields that display precisely the "last value" of plots. "Last value" is the value of the indicator (or price) for the last currently displayed (rightmost) bar. Y-axis area is used also to move/size chart vertically.

Next to the right is a drawing objects toolbar that allows you to choose from available drawing types (note that only most popular tools are shown here, complete set is available from **Insert** menu). A special tool called

"Select" (red arrow) is used to select/move/resize already drawn objects and to select quotes from the chart.

In the upper part you can see formatting toolbar that allows you to quickly modify color, style (thick/dotted) and mode (snap to price) of currently selected drawing object.

In the picture you can also see the trend line drawn with sizing handles marked. These handles are used to drag/size the object as will be explained below.

Basic operations

Scrolling

To **scroll** the chart forward/backward just drag scroll bar thumb or use < and > arrows on the left and right sides of the scroll bar. Note that using < > scroll bar arrows allows you to move chart by one bar. To scroll the chart you can also use the mouse equipped with a wheel. Just roll the wheel up and down to scroll back and forward.

Zooming

To **zoom** the chart (increase or decrease number of data points (bars) displayed) you can use either **View->Zoom** menu, zoom toolbar or mouse wheel.
You can also zoom by **dragging the left or right edge of scroll ba**r. There are following options available: zoom-in - decreases the number of data points displayed, zoom-out - increases the number of data points displayed, zoom-all - displays all available bars, zoom-normal resets number of bars displayed to the value defined in **Tools->Preferences->Charting.** Zoom-in and zoom-out options are accessible directly from the View toolbar. (see picture below). To zoom using mouse wheel just press and hold down CTRL key and roll the wheel. You can also zoom to any from-to range selected on the chart (see 'Marking range' later in this tutorial)

Shrinking, expanding and moving Y-axis scale

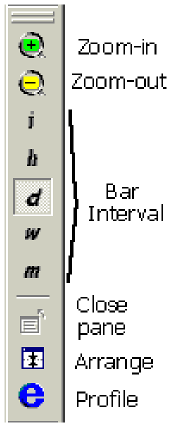
To **move** Y-axis scale hover the mouse to Y-axis area (marked with blue color in the picture above) and you will see that cursor changes to up/down arrow. Now you click and drag up/down Y axis and release button when the axis is in the correct position.

To **shrink/expand** Y-axis scale: press down SHIFT key and click in the Y-axis area, now shrink/expand Y axis scale by moving your mouse up and down. Release the button to finish.

To **reset** Y-axis scale and position simply double click in the Y-axis area.

Changing bar interval (periodicity)

You can easily switch between daily/weekly/monthly and intraday intervals by choosing it from **View** menu and pressing the toolbar button (see below).



The toolbar uses following notation for intervals **- i** -intraday, **h** - hourly, **d** - daily, **w** - weekly, **m** - monthly. The **i** represents "base" intraday interval as defined in **File->Database Settings**. Remaining intraday intervals are available from **View->Intraday** menu.

The interval setting affects active window only, so each window can have different interval.

Please note that intraday intervals are disabled if your database is in end-of-day mode. Intraday modes are available only for databases that have "Base time interval" in **File->Database Settings** set to anything less than end-of-day. If you for example set "Base time interval" in **File->Database Settings** to 5-minute, all chart periodicities from 5-minutes up will be enabled.

The following intervals are built-in:

- daily
- weekly
- monthly
- hourly (intraday)
- 15-minute (intraday)
- 5-minute (intraday)
- 1-minute (intraday)
- 15-second (intraday RT only)
- 5-second (intraday RT only)
- tick (intraday RT only)

In addition to that you can define 5 custom n-minute bar intervals and 5 custom n-tick intervals in **Tools->Preferences->Intraday**. Custom intervals are available from **View->Intraday** menu only.

Selecting a quote

You can very easily see the past quote and values of indicators by using "select" mode. To **select** past quote first switch to "Select" mode (red arrow in the toolbar) then click in the chart area (but not on the drawing object). A vertical line will show up marking the quote under the cursor. The chart title will display this bar quote. Indicator panes will show indicator value for given bar. Once quote is selected you can move to previous/next quote using keyboard left and right arrow (cursor) <- and -> keys.

To **switch off** quote selection either click again on the line or click in the date axis area (marked with red color in the picture above) or click in the right margin (blank quotes) area. When selection is off chart title displays the values for last visible bar.

Marking range

To show range marker just double click the chart at the beginning of the range and double click again at the end of the range. You can also use F12 key in conjunction with "select" mode (described above). Just select quote and press F12 for begin and SHIFT+F12 for the range end. You can switch off the range marker by pressing CTRL+F12 key or double clicking in the same place twice.

Range markers can be used to select zoom-in range (View->Zoom->Range) and to perform calculations on selected values via BeginValue and EndValue AFL functions.

Adding / closing chart panes

Each window can consist of several panes displaying various charts / indicators. To display a new indicator in a separate chart pane just find the indicator in the Charts list (use **Window -> Charts** menu) and **double-click** on the indicator name. For more information see Drag&drop charting tutorial.

To close any chart pane: click on the pane, then use either **View->Pane->Close** from main menu or **click on the pane with right mouse button** and choose **Close** from context menu.

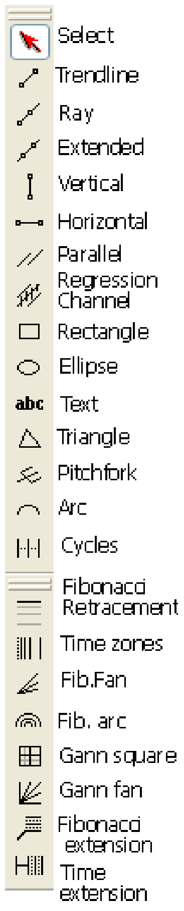
Linking and locking chart

Multiple chart windows (that were open usign **File->New->Default Chart** or **File->New->Blank chart**) can be interval-linked, symbol-linked using appropriate small "S" and "I" buttons that appear on the left side of the scroll bar. When you click on button the menu showing colors will pop up, select one color from 2 or more charts and symbols linked using same color will be linked using symbol and/or interval. Linking means that change of the symbol and/or interval in one of the linked windows automatically changes symbol and/or interval in all linked windows using same link color.

You can also prevent symbol from being changed for given chart. It can be done by turning on little pad lock button ("Symbol Lock") on the right side of the scroll bar. When chart is symbol-locked, it will not allow to change the symbol selected unless the lock is released (by pressing pad lock button again).

Using drawing tools

AmiBroker features extensive set of drawing tools:



The following tools are available:

- trend line
- ray (new in 4.20)
- extended line (new in 4.20)
- vertical line
- horizontal line
- parallel lines (new in 4.20)
- Regression channels: Raff, standard deviation, standard error (all new in 4.20)
- Fibonacci Retracement study (enhanced in 4.20)
- Fibonacci Time zones study
- Fibonacci Fan
- Fibonacci arc
- Gann Square (new in 4.20)
- Gann Fan (new in 4.20)
- Ellipse tool
- Arc tool
- Rectangle
- text box tool

They are available from **Insert** menu and **Draw** toolbar. Each drawing object can be moved, resized, copied, deleted and modified after it is drawn.

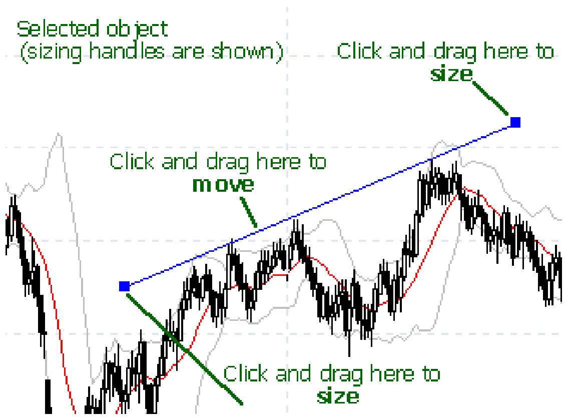
To **draw** an object on the chart switch on appropriate tool button (see picture below) and start drawing on the chart by pointing the mouse and pressing left mouse button where you want to start the drawing. Then move the mouse. Study tracking line will appear. Release left mouse button when you want to finish drawing. You can also cancel study drawing by pressing ESC (escape) key.

If you hover your mouse over the object you will see that cursor shape changes in the proximity of the object. This means that

If cursor is near either end of the object it will change its shape to **sizing** pointer:
⤡

If the cursor is near remaining parts of the object it will change its shape to **moving** pointer:
✥

Once object is drawn it can be selected, moved, resized, deleted, copied.



To **select** the object simply move the mouse over the object so "moving pointer" appears and click once - the object will be marked so the sizing handles (see first picture) will appear.

To **de-select** click in the blank chart space.

To **size** the object click on the sizing handle and drag to the desired location as shown in the picture.

To **move** the object click on any other part of the object and move to the desired location.

To **delete** object - select it first and press **DEL** (**DELETE**) key on the keyboard or use **Edit->Delete** menu or use Delete toolbar button.

To **copy** the object to the clipboard - select it first and press **Ctrl+C** or use **Edit->Copy** menu or use Copy toolbar button.

To **cut** the object - select it first and press **Ctrl+X** or use **Edit->Cut** menu or use Cut toolbar button.

To **paste** the object from the clipboard press **Ctrl+V** or use **Edit->Paste** or use Paste toolbar button. Pasted object will drawn in the exactly same location as copied one and will be selected automatically so you can move it to a new location.

To **apply color or style** to the object select it and use Format menu or Format tool bar buttons to change color, thick, dotted and snap to price styles. Note that you can also select color and style of the object <u>before</u> drawing new object: simply deselect previous object (if any), change color / style selections and draw new object.

To **modify properties** of the object - either double click it or use **Edit->Properties** menu or **Alt+ENTER** key

To **delete all** objects use **Edit->Delete All** menu

Further information

To learn more about drawing tools please read Drawing tools reference chapter.

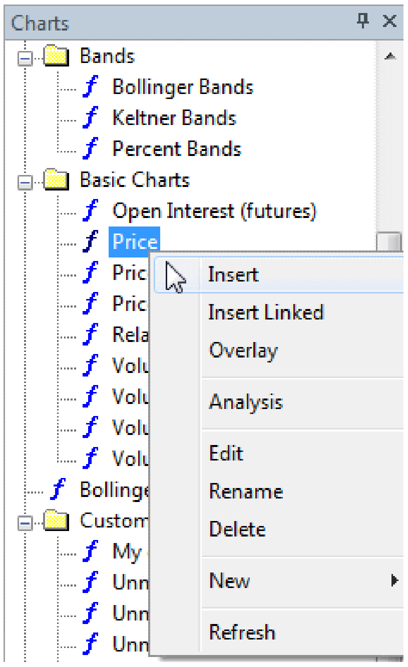
How to use drag-and-drop charting interface

Introduction
AmiBroker allows you to easily create and modify your indicators with few moves of a mouse. From now on you can build sophisticated indicators without any programming knowledge at all. The available (ready-to-use) indicators are listed in **Charts** tab of the **Workspace** window.

There is a video tutorial at: http://www.amibroker.net/video/dragdrop1.html that shows basic usage of new drag and drop functionality.

How to insert a new indicator.

To display a new indicator in a separate chart pane just find the indicator in the Charts list (use **Window -> Charts** menu) and **double-click** on the indicator name.



Alternatively you can choose **Insert** from the context menu. As a result new indicator pane will be created and **Parameters** dialog will be displayed. Here you can change the properties of the indicator (like color or periods). To accept the settings press **OK** button. (you will find the detailed description of parameters window below).

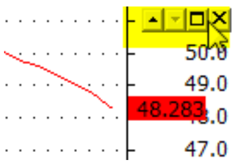
Example:
To insert RSI pane - find RSI indicator in the list, double-click on the name, select the number of periods and color, then press OK.

How to overlay one indicator on another indicator.

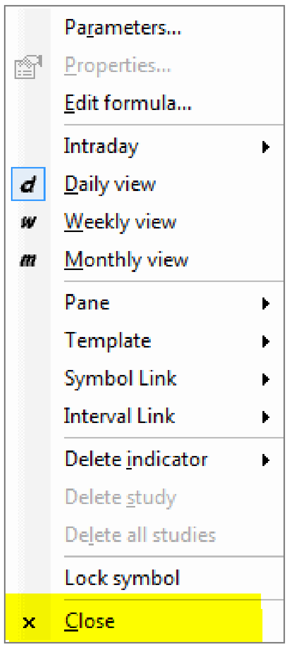
To overlay one indicator on another one, press LEFT mouse button on the indicator name, drag (with mouse button held) the chosen indicator into the destination pane and release the button.
Example:
To insert another RSI (based different periods number) into the same pane - drag RSI into the previously created RSI pane, change the number of periods in the Parameters window and press OK
Alternatively you can choose **Overlay** option from context menu.

How to delete the indicator.

To remove the indicator, press **Close** button from the menu on the top right-hand side of the indicator pane (the menu will be displayed if you place the mouse cursor in the nearby). This menu allows you also to move the indicator pane up/down or maximize the pane.

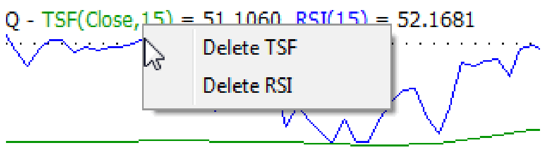


You can also use **Close** command from context menu that shows up when you click on the chart pane with right mouse button.



How to remove the indicator plot from the pane.

To remove one of the indicators displayed in the indicator pane - click with RIGHT mouse button on the chart title (near the top of chart pane) and select the indicator that you want to remove.



You can also remove the indicator plot using **Delete Indicator** option from chart context menu.

How to change parameters/colors/styles of indicators.

The **Parameters** window allows you to change parameters, colors and styles of your indicators. Parameters window is displayed when you insert a new indicator. You can also click RIGHT mouse on the chart pane and choose Parameters from the context menu. Parameters window displays all the parameters defined in AFL code of certain indicators (also user-defined parameters) so it's contents depends on the indicator chosen. However - for most of the indicators you will see:

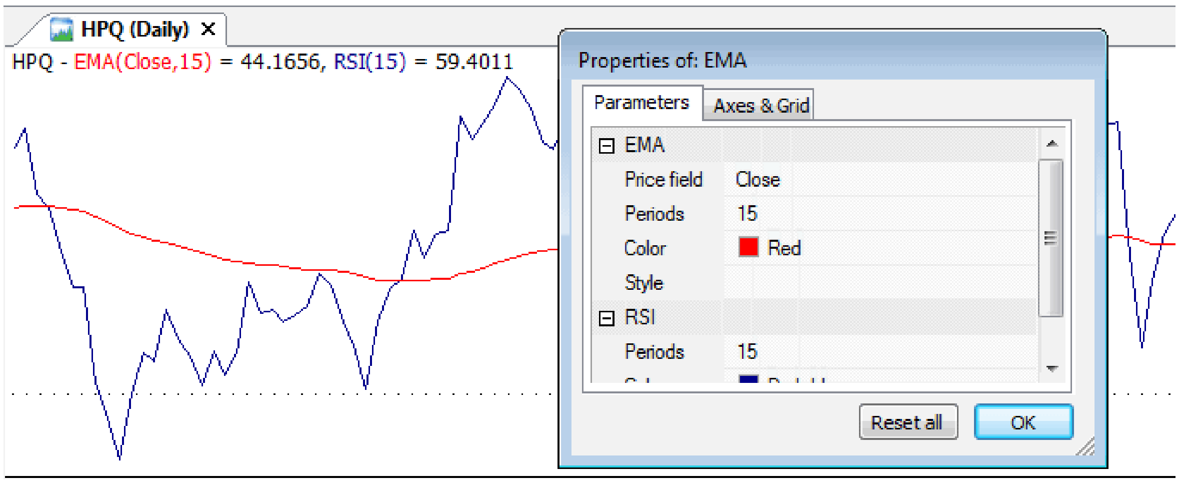
- **Price Field** - the data used to calculate the indicator. If the 'Price Field' contains 'Close', it means that indicator is calculated out of Close prices. Price Field is not available for all indicators, because not all indicators allow you to choose the input (e.g. ADLine).
- **Periods** - defines the number of periods used to calculate the indicator
- **Color** - allows you to change the color of the indicator
- **Style** - allows you to determine the style of the plot (the styles are described in more detail in Using graph styles and colors tutorial section.

How to overlay indicators with different scales.

To have in one pane two (or more) indicators that use different scaling, drag the second indicator onto the first one, in Parameters window click on **Style** field and check **StyleOwnScale** setting.
Example:
Drag OBV (On Balance Volume) into RSI pane. Then define style as styleOwnScale. As a result - both indicators are visible and properly displayed.

How to create an indicator based on another indicator.

AmiBroker allows you also to easily create indicators based on values of another indicator. All you need to do is to press LEFT mouse button on the indicator name, drag (with mouse button held) the chosen indicator into the destination pane and release the button. As a result - the indicator will be placed in the existing chart pane. In the parameters dialog **Price field** parameters indicates what base values are used to calculate the indicator.
Example:
To calculate Simple Moving Average of previously created RSI indicator, drag the MA indicator into RSI pane. The contents of "Price Field" parameter indicates, that Moving Average is calculated out of RSI(15) values. (See the below picture).



HPQ - EMA(Close,15) = 44.1656, RSI(15) = 59.4011

Properties of: EMA

⊟ EMA	
Price field	Close
Periods	15
Color	🟥 Red
Style	
⊟ RSI	
Periods	15

NOTE: The part below contains technical information for advanced users only. Beginners may skip this part.

Using Param(), ParamColor(), ParamToggle(), ParamStyle() functions

These functions, when used in formula, allow you to change indicators' settings directly from **Parameters** window.

Param(*("name", defvalue, min = 0, max = 100, step = 1, sincr = 0*)
Adds a new user-definable parameter, which will be accessible via Parameters dialog.

- "name" - defines parameter name that will be displayed in the parameters dialog
- defvalue - defines default value of the parameter
- min, max - define minimum and maximum values of the parameter
- step - defines minimum increase of the parameter via slider in the Parameters dialog
- sincr - defines the increase of default value when more than one section of the same kind is inserted (dropped) onto the chart. For example if you insert the default Moving Average indicator into the same pane twice, the first moving average will be based on 15 periods, the other one on 25 (defvalue=15 + sincr=10)

ParamColor(*"name", defaultcolor*)
Adds a new user-definable color parameter, accessible via Parameters dialog.

- "name" - defines parameter name that will be displayed in the parameters dialog
- defaultcolor - defines default color value of the parameter

ParamColor function allows you to use **colorCycle** as a default value. When you use colorCycle parameter, default color cycles through red, blue, green, turquoise, gold, violet, bright green, dark yellow, when you insert your indicators into the same pane.

ParamStyle(*"name", defaultval = styleLine, mask = maskDefault*) - allows to select the styles applied to the plot from the Parameters window. Apart from styles available in previous versions of AmiBroker, there are two new style constants:

- styleHidden - a combination of styleNoDraw | styleNoRescale
- styleDashed - dashed line

The list of available styles displayed in the Parameters window depends on the **mask** parameter.

- maskDefault - show thick, dashed, hidden, own scale styles (this is default mask for ParamStyle)
- maskAll - show all style flags
- maskPrice - show thick, hidden, own scale, candle, bar
- maskHistogram - show histogram, thick, hidden, own scale, area

ParamField(*"name", field = 3*) - allows to pick the **Price field** for the indicator (field which is used to calculate values of the indicator). Function returns the array defined by *field* parameter. Default value = 3 returns Close array. The possible values of *field* parameter are:

- **-1** - ParamField returns the values of the indicator that was inserted as a first one into the pane, or Close if no indicator was present
- **0** - returns **Open** array
- **1** - returns **High** array
- **2** - returns **Low** array
- **3** - returns **Close** array (default)
- **4** - returns **Average** array = (H+L+C)/3
- **5** - returns **Volume** array
- **6** - returns **Open Interest** array
- **7,8,9,....** - return values of indicators inserted into the pane.

ParamToggle(*"name","values",defaultval=0*) - function that allows to use boolean (Yes/No) parameters.

- "name" - the name of the parameter
- "values" - parameter values (separated with | character, e.g. "No|Yes" - first string represents false value and second string represents true value)
- defaultval - default value of the parameter

_

Example:

The below indicator allows you to check how the parameters work in the custom code. You can change settings from Parameters dialog.

```
Buy = Cross(MACD(), Signal() );
Sell = Cross(Signal(), MACD() );

pricefield = ParamField("Price Field", 2);
Color = ParamColor("color",colorRed);
style = ParamStyle("style",styleLine,maskAll);
arrows = ParamToggle("Display arrows", "No|Yes",0);
Plot(pricefield,"My Indicator",Color,style);
if(arrows)
{
  PlotShapes(Buy*shapeUpArrow+Sell*shapeDownArrow,IIf(Buy,colorGreen,colorRed) );
}
```

Special functions: SECTION_BEGIN, _SECTION_END, _SECTION_NAME, _DEFAULT_NAME, _PARAM_VALUES explained (for advanced users only)

These are new functions that are used by drag & drop mechanism. The most important pair is _SECTION_BEGIN("name") and _SECTION_END().

When you drop the formula onto chart pane AmiBroker appends the formula you have dragged at the end of existing chart formula and wraps inserted code with _SECTION_BEGIN("name") and _SECTION_END() markers:

So, if original formula looks as follows:

```
P = ParamField("Price field",-1);
Periods = Param("Periods", 15, 2, 200, 1, 10 );
Plot( MA( P, Periods ), _DEFAULT_NAME(), ParamColor( "Color", colorCycle ),
ParamStyle("Style") );
```

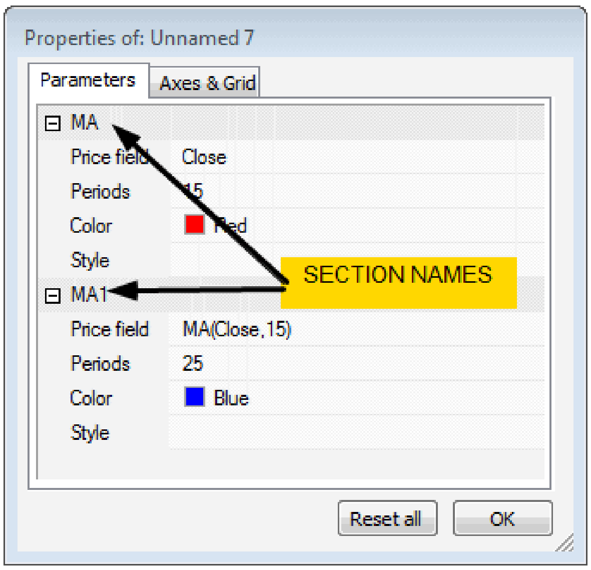
it will be transformed by AmiBroker to:

```
_SECTION_BEGIN("MA");
P = ParamField("Price field",-1);
Periods = Param("Periods", 15, 2, 200, 1, 10 );
Plot( MA( P, Periods ), _DEFAULT_NAME(), ParamColor( "Color", colorCycle ),
ParamStyle("Style") );
_SECTION_END();
```

_SECTION_BEGIN/_SECTION_END markers allow AmiBroker to identify code parts and modify them later (for example remove individual sections). In addition to that sections provide the way to make sure that parameters having the same name in many code parts do not interfere each other. For example if you drop two moving averages the resulting code will look as follows:

```
_SECTION_BEGIN("MA");
P = ParamField("Price field",-1);
Periods = Param("Periods", 15, 2, 200, 1, 10 );
Plot( MA( P, Periods ), _DEFAULT_NAME(), ParamColor( "Color", colorCycle ),
ParamStyle("Style") );
_SECTION_END();

_SECTION_BEGIN("MA1");
P = ParamField("Price field",-1);
Periods = Param("Periods", 15, 2, 200, 1, 10 );
Plot( MA( P, Periods ), _DEFAULT_NAME(), ParamColor( "Color", colorCycle ),
ParamStyle("Style") );
_SECTION_END();
```

Note that code and is parameter names are identical in both parts. Without sections the parameters with the same name will interfere. But thanks to uniquely named sections there is no conflict. This is so because AmiBroker identifies the parameter using section name AND parameter name, so if section names are unique then parameters can be uniquely identified. When dropping indicator AmiBroker automatically checks for already existing section names and auto-numbers similarly named sections to avoid conflicts. Section name also appears in the Parameter dialog:



⊟ MA	
Price field	Close
Periods	15
Color	🟥 Red
Style	
⊟ MA1	
Price field	MA(Close,15)
Periods	25
Color	🟦 Blue
Style	

Last but not least: you should NOT remove _SECTION_BEGIN / _SECTION_END markers from the formula. If you do, AmiBroker will not be able to recognize sections inside given formula any more and parameters with the same name will interfere with each other.

_SECTION_NAME is a function that just gives the name of the function (given in previous _SECTION_BEGIN call).

_DEFAULT_NAME is a function that returns the default name of plot. The default name consists of section name and comma separated list of values of numeric parameters defined in given section. For example in this code:

```
_SECTION_BEGIN("MA1");
P = ParamField("Price field");
Periods = Param("Periods", 15, 2, 200, 1, 10 );
Plot( MA( P, Periods ), _DEFAULT_NAME(), ParamColor( "Color", colorCycle ),
ParamStyle("Style") );
_SECTION_END();
```

_DEFAULT_NAME will evaluate to "MA1(Close,15)" string.

_PARAM_VALUES works the same as _DEFAULT_NAME except that no section name is included (so only the list of parameter values is returned). So in above example _PARAM_VALUES will evaluate to "(Close, 15)" string.

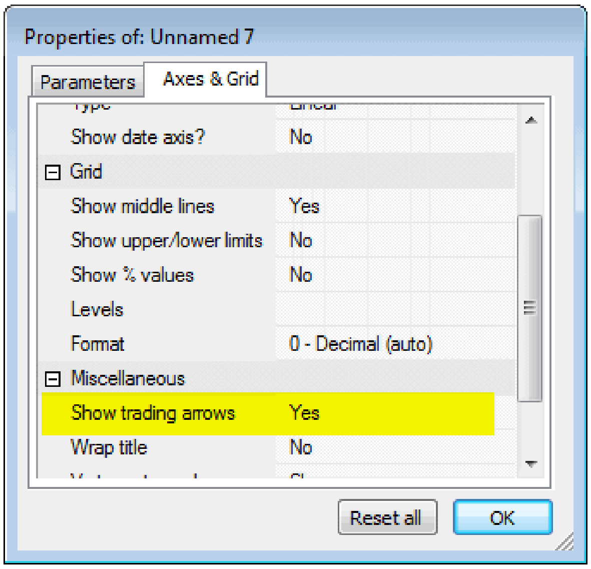
Frequently Asked Questions about drag & drop functionality

Q. What is the difference between Insert and Insert Linked option in chart menu?

A. **Insert** command internally creates a copy of the original formula file and places such copy into hidden drag-drop folder so original formula will not be affected by subsequent editing or overlaying other indicators onto it. Double clicking on formula name in the chart tree is equivalent with choosing **Insert** command from the menu. On the other hand **Insert Linked** command does not create any copy of the formula. Instead it creates new chart pane that directly links to original formula. This way subsequent editing and/or overlaying other indicators will modify the original

Q. I can not see buy/sell arrows from my trading system

A. Trade arrows can be displayed on any chart pane (not only one built-in price chart). However, by default, the arrow display is turned OFF. To turn it ON you have to open Parameter dialog, switch to "Axes and grid" and switch "Show trading arrows" option to "Yes".



Show date axis?	No
⊟ Grid	
Show middle lines	Yes
Show upper/lower limits	No
Show % values	No
Levels	
Format	0 - Decimal (auto)
⊟ Miscellaneous	
Show trading arrows	Yes
Wrap title	No

Q. The read me says: "Automatic Analysis formula window is now drag&drop target too (you can drag formulas and AFL files onto it)". What does it mean?

A. It means that you can drag the formula from either Chart tree or .AFL file from Windows Explorer and drop it onto Automatic Analysis (AA) formula window and it will load the formula into AA window. This is an alternative to loading formula via "Load" button in AA window.

Q. Can I drop a shortcut onto the formula window ?

A: No you can't. You can only drag & drop files with .AFL extension (shortcuts in Windows have .lnk extension).

Q. Can I add my own formulas to the Chart tree ?

A. Yes you can. Simply save your .AFL formula into Formulas subfolder of AmiBroker directory and it will appear under "Charts" tree (View->Refresh All may be needed to re-read the directory if you are using external editor)

Q. I have added new file to the Formulas folder, but it does not show up in the Charts tree unless I restart AmiBroker? Is there a way to refresh Chart tree ?

A. You can refresh Chart tree by choosing **View->Refresh All** menu.

Q. If I modify the formula that ships with AmiBroker will it be overwritten by next upgrade?

A. Yes it will be overwritten. If you wish to make any modifications to the formulas provided with AmiBroker please save your modified versions under new name or (better) in your own custom subfolder.

Q. I can see Reset All button in Parameters dialog but it sets all parameters to default values. Is there a way to reset SINGLE parameter ?

A. No, there is no such option yet, but it will be added in upcoming betas.

Q. I dragged RSI to the price chart pane and got a straight red line at the bottom of the pane. What is wrong?

A. When you drop two indicators / plots that have drastically different values you have to use style OwnScale for one of it. You can turn on OwnScale style using Parameter dialog. This ensures that scales used for each are independent and you can see them properly. Otherwise they use one common scale that fits both value ranges that results in flattened plots.

Q. The light grey color of the new AFL special functions_SECTION_BEGIN etc makes them invisible in my bluegrey background IB color. How could I change the special functions color ?

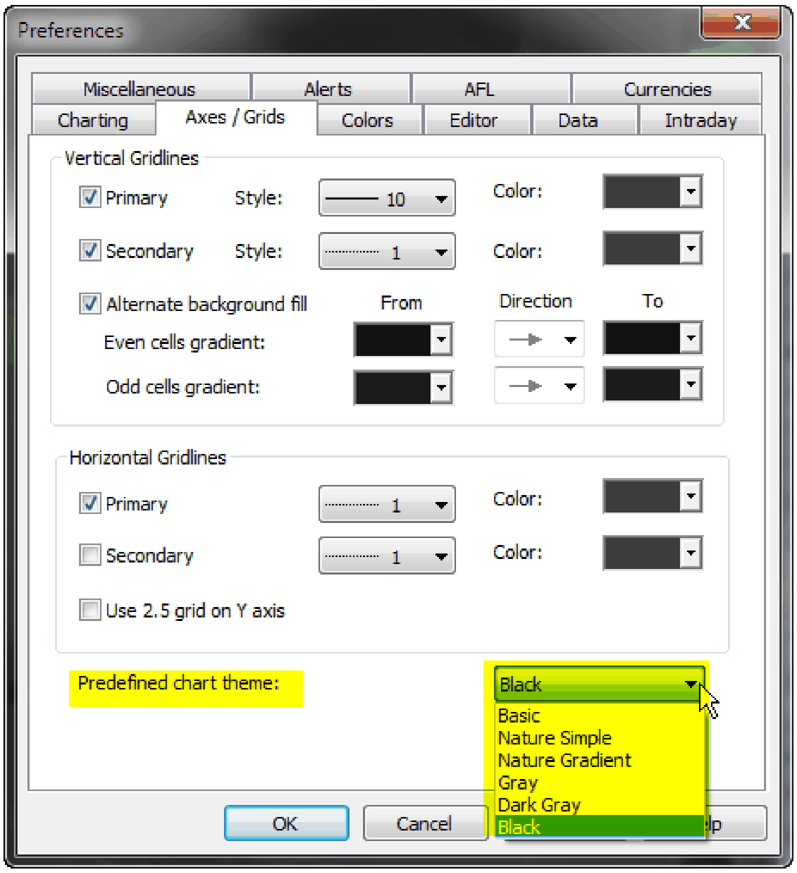
A. Right now, you can't. But there will be a setting for coloring special functions in the next version.

Q. When I drop the indicator the Parameter dialog does not show all parameters. Is this correct ?

A. Yes it works that way. The idea behind it is simple. When you drop new indicator AmiBroker displays a dialog with parameters ONLY for currently dropped indicator. This is to make sure that newly inserted indicator parameters are clearly visible (on top) and new user is not overwhelmed by tens of other parameters referring to previously dropped indicators. On the other hand when you choose "Parameters" item from context menu then ALL parameters will show up - allowing you to modify them all any time later.

Chart themes

AmiBroker 5.52 introduces 6 pre-defined chart themes switchable in Tools->Preferences, "Axes & Grid" tab:



Vertical Gridlines

[✓] Primary Style: 10 Color:

[✓] Secondary Style: 1 Color:

[✓] Alternate background fill From Direction To

Even cells gradient:

Odd cells gradient:

Horizontal Gridlines

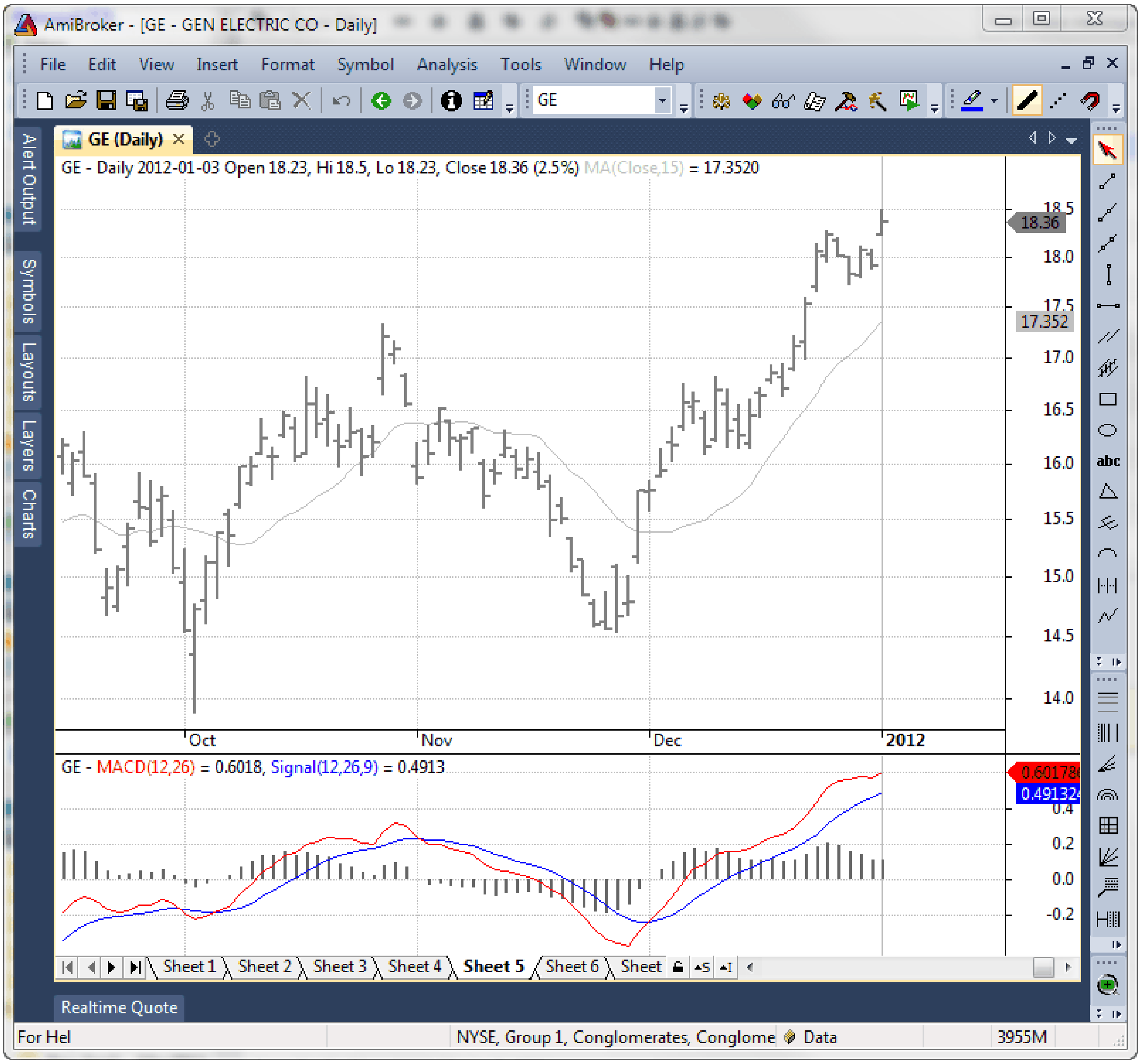
[✓] Primary 1 Color:

[ ] Secondary 1 Color:

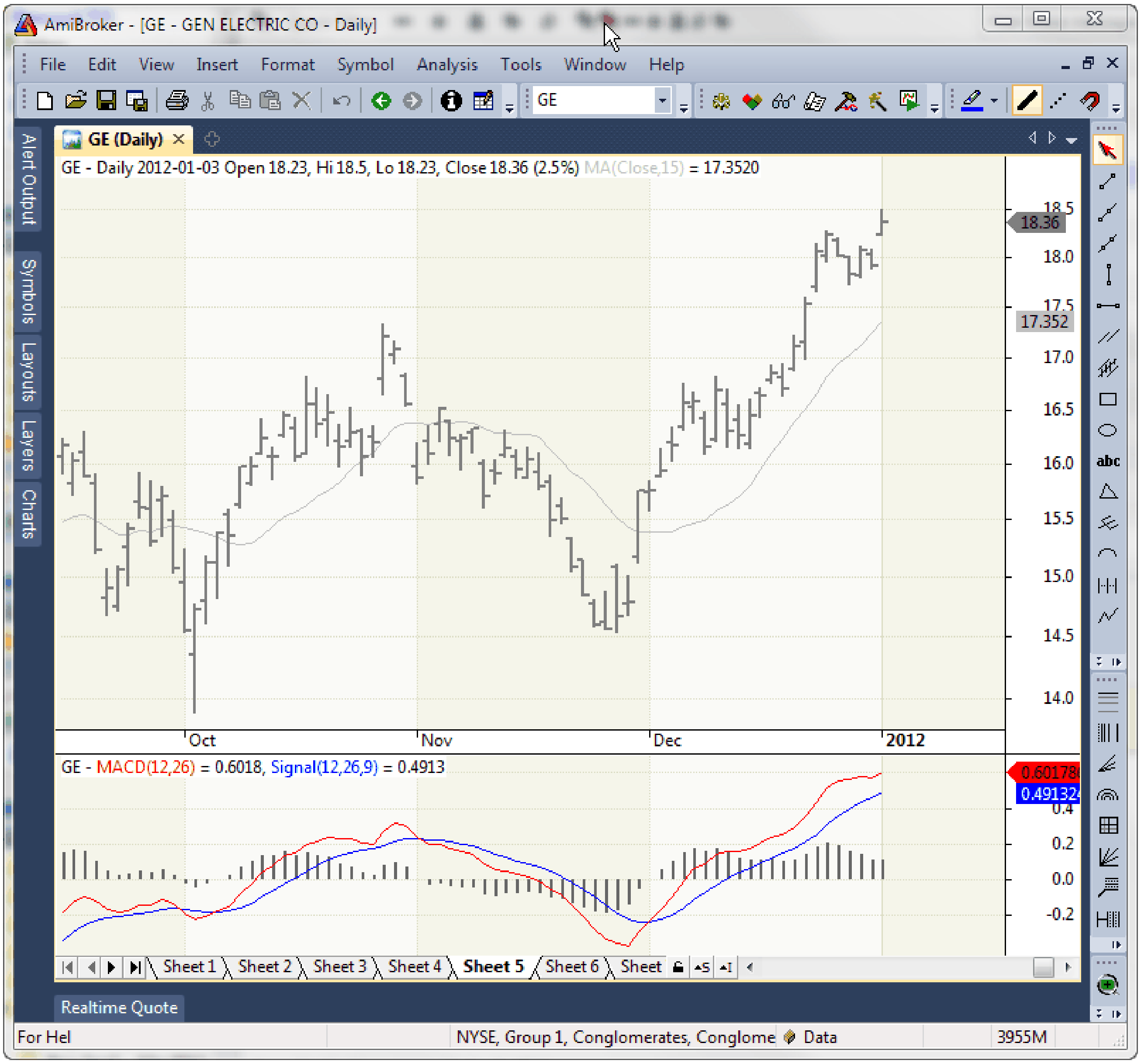
[ ] Use 2.5 grid on Y axis

Predefined chart theme: Black

1. Basic Theme


AmiBroker - [GE - GEN ELECTRIC CO - Daily]
File Edit View Insert Format Symbol Analysis Tools Window Help
GE
GE (Daily)
Alert Output
Symbols
Layouts
Layers
Charts
GE - Daily 2012-01-03 Open 18.23, Hi 18.5, Lo 18.23, Close 18.36 (2.5%) MA(Close,15) = 17.3520
18.36
17.352
18.5
18.0
17.5
17.0
16.5
16.0
15.5
15.0
14.5
14.0
Oct
Nov
Dec
2012
GE - MACD(12,26) = 0.6018, Signal(12,26,9) = 0.4913
0.60178
0.49132
0.4
0.2
0.0
-0.2
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Sheet 5
Sheet 6
Sheet
Realtime Quote
For Hel
NYSE, Group 1, Conglomerates, Conglome
Data
3955M

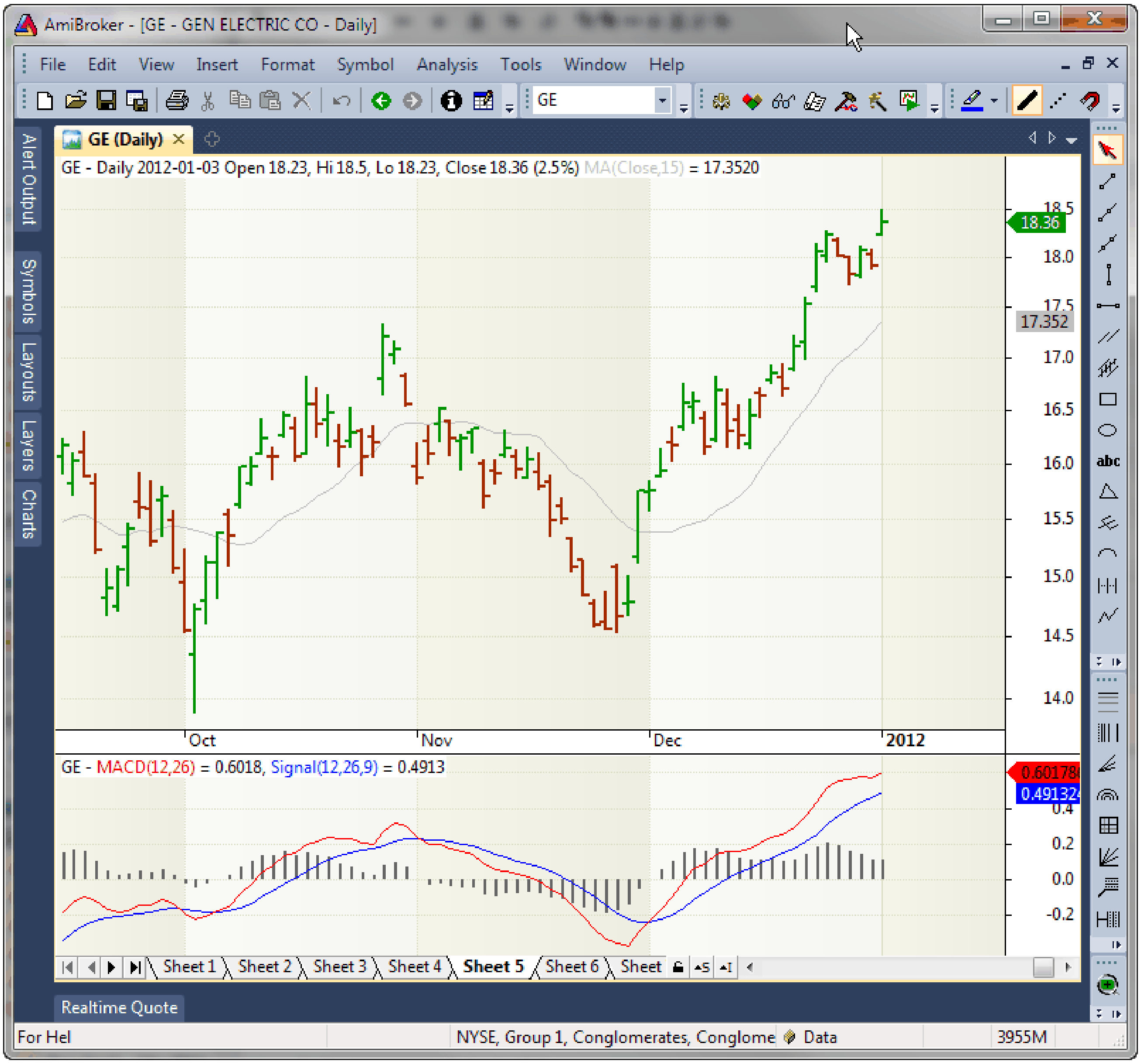
2. Nature Simple Theme


AmiBroker - [GE - GEN ELECTRIC CO - Daily]
File Edit View Insert Format Symbol Analysis Tools Window Help
GE
GE (Daily)
Alert Output
Symbols
Layouts
Layers
Charts
Oct
Nov
Dec
2012
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Sheet 5
Sheet 6
Sheet
Realtime Quote
For Hel
NYSE, Group 1, Conglomerates, Conglome
Data
3955M

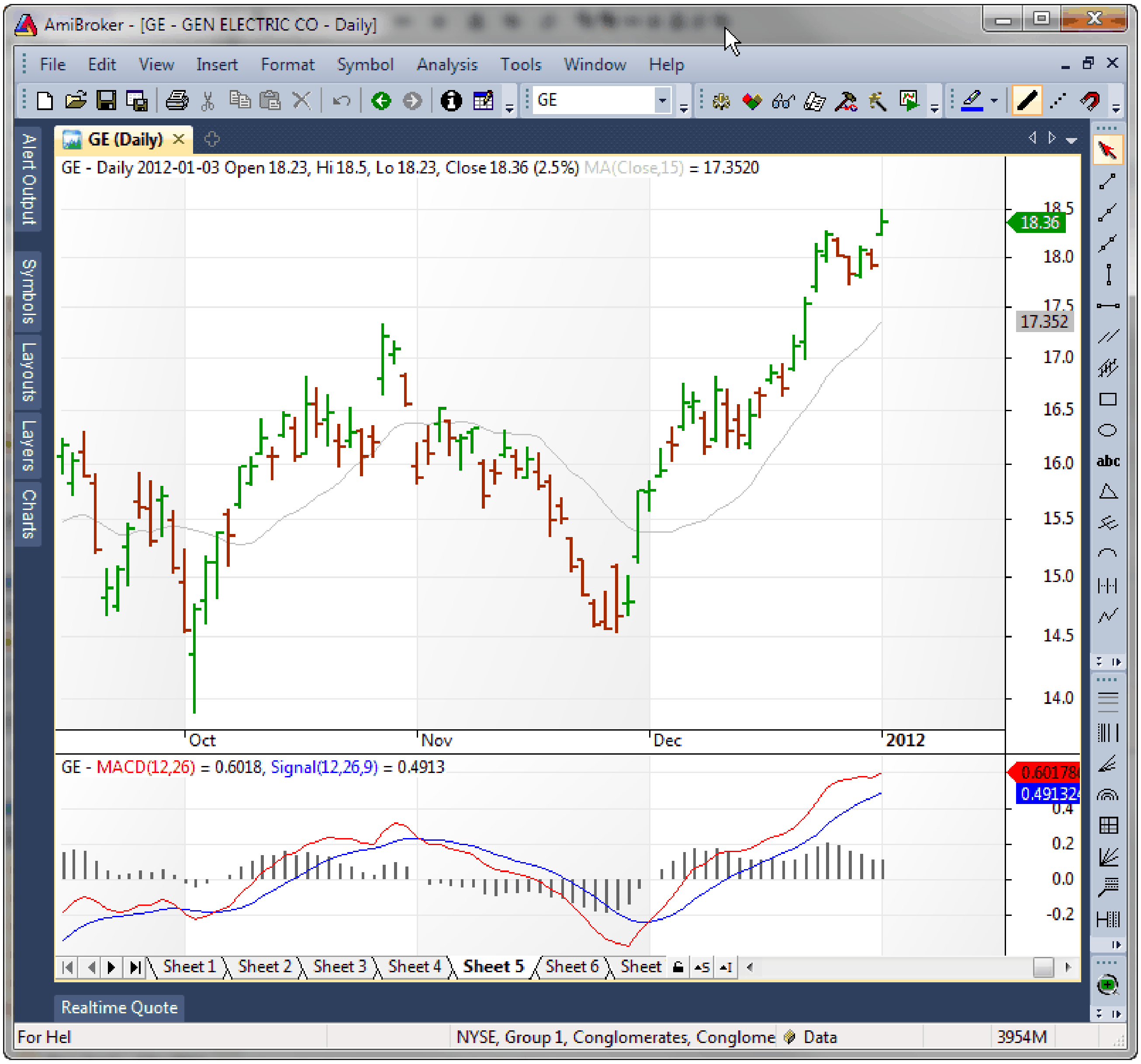
GE - Daily 2012-01-03 Open 18.23, Hi 18.5, Lo 18.23, Close 18.36 (2.5%) MA(Close,15) = 17.3520

GE - MACD(12,26) = 0.6018, Signal(12,26,9) = 0.4913

3. Nature Gradient Theme


AmiBroker - [GE - GEN ELECTRIC CO - Daily]
File Edit View Insert Format Symbol Analysis Tools Window Help
GE (Daily)
GE - Daily 2012-01-03 Open 18.23, Hi 18.5, Lo 18.23, Close 18.36 (2.5%) MA(Close,15) = 17.3520
Oct
Nov
Dec
2012
GE - MACD(12,26) = 0.6018, Signal(12,26,9) = 0.4913
Sheet 1 Sheet 2 Sheet 3 Sheet 4 Sheet 5 Sheet 6 Sheet
Realtime Quote
Alert Output
Symbols
Layouts
Layers
Charts
For Hel
NYSE, Group 1, Conglomerates, Conglome
Data
3955M

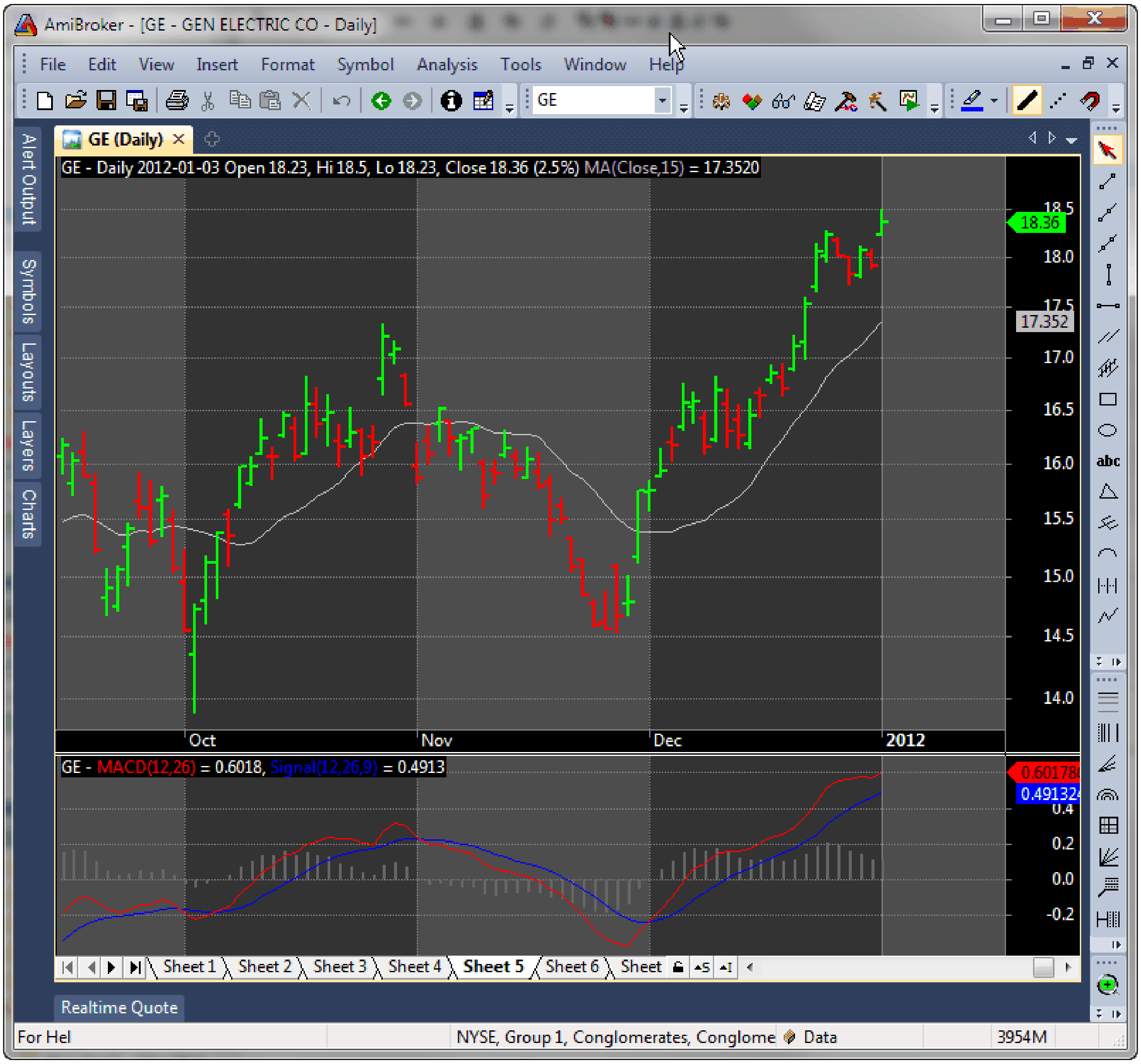
4. Gray Theme



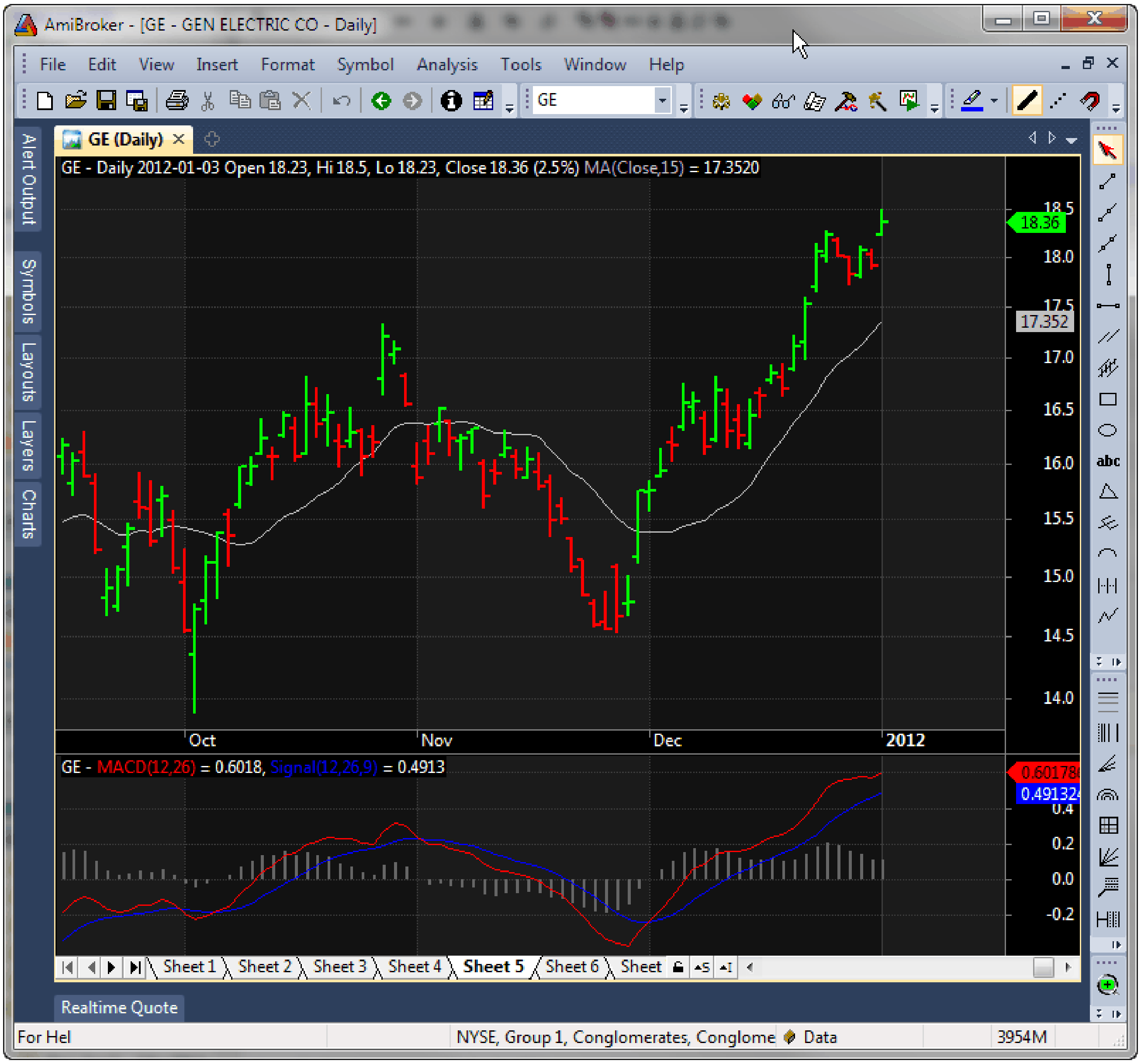
GE - Daily 2012-01-03 Open 18.23, Hi 18.5, Lo 18.23, Close 18.36 (2.5%) MA(Close,15) = 17.3520

GE - MACD(12,26) = 0.6018, Signal(12,26,9) = 0.4913

5. Dark Gray Theme


AmiBroker - [GE - GEN ELECTRIC CO - Daily]
File Edit View Insert Format Symbol Analysis Tools Window Help
GE
GE (Daily)
Alert Output
Symbols
Layouts
Layers
Charts
GE - Daily 2012-01-03 Open 18.23, Hi 18.5, Lo 18.23, Close 18.36 (2.5%) MA(Close,15) = 17.3520
18.36
17.352
18.5
18.0
17.5
17.0
16.5
16.0
15.5
15.0
14.5
14.0
Oct
Nov
Dec
2012
GE - MACD(12,26) = 0.6018, Signal(12,26,9) = 0.4913
0.60178
0.49132
0.4
0.2
0.0
-0.2
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Sheet 5
Sheet 6
Sheet
Realtime Quote
For Hel
NYSE, Group 1, Conglomerates, Conglome
Data
3954M

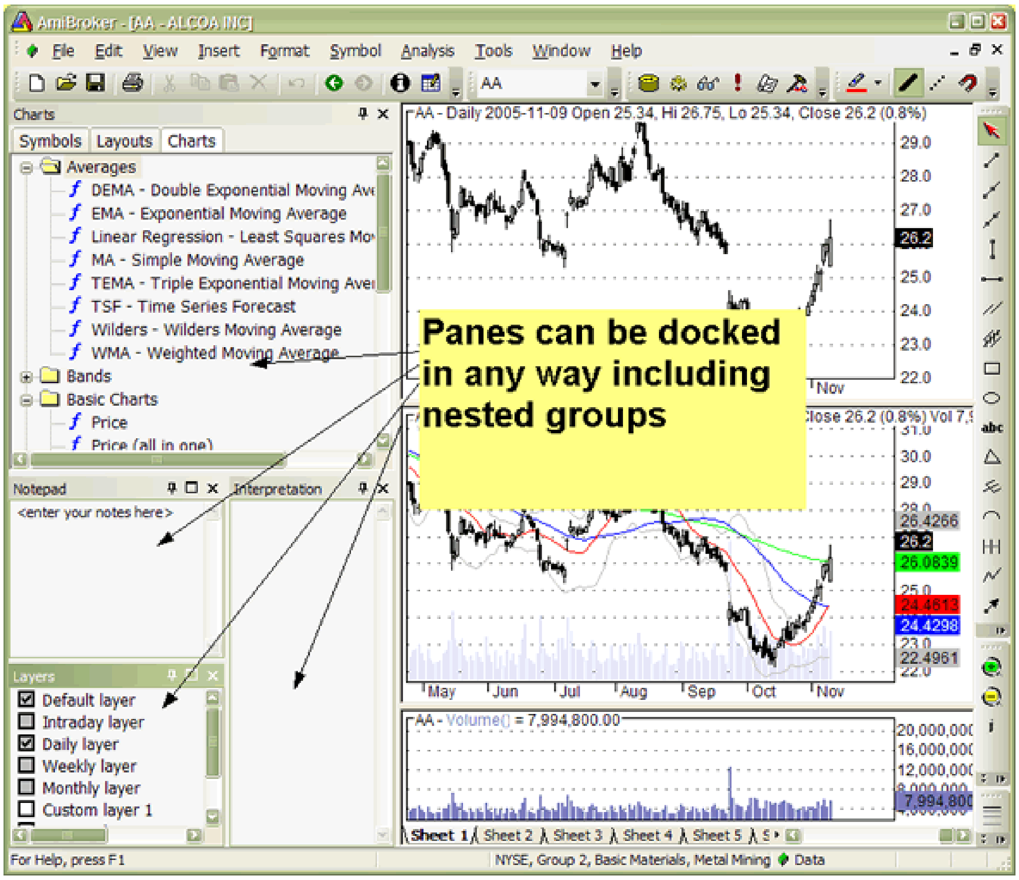
6. Black Theme


AmiBroker - [GE - GEN ELECTRIC CO - Daily]
File Edit View Insert Format Symbol Analysis Tools Window Help
GE
GE (Daily)
Alert Output
Symbols
Layouts
Layers
Charts
GE - Daily 2012-01-03 Open 18.23, Hi 18.5, Lo 18.23, Close 18.36 (2.5%) MA(Close,15) = 17.3520
18.36
17.352
Oct
Nov
Dec
2012
GE - MACD(12,26) = 0.6018, Signal(12,26,9) = 0.4913
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Sheet 5
Sheet 6
Sheet
Realtime Quote
For Hel
NYSE, Group 1, Conglomerates, Conglome
Data
3954M

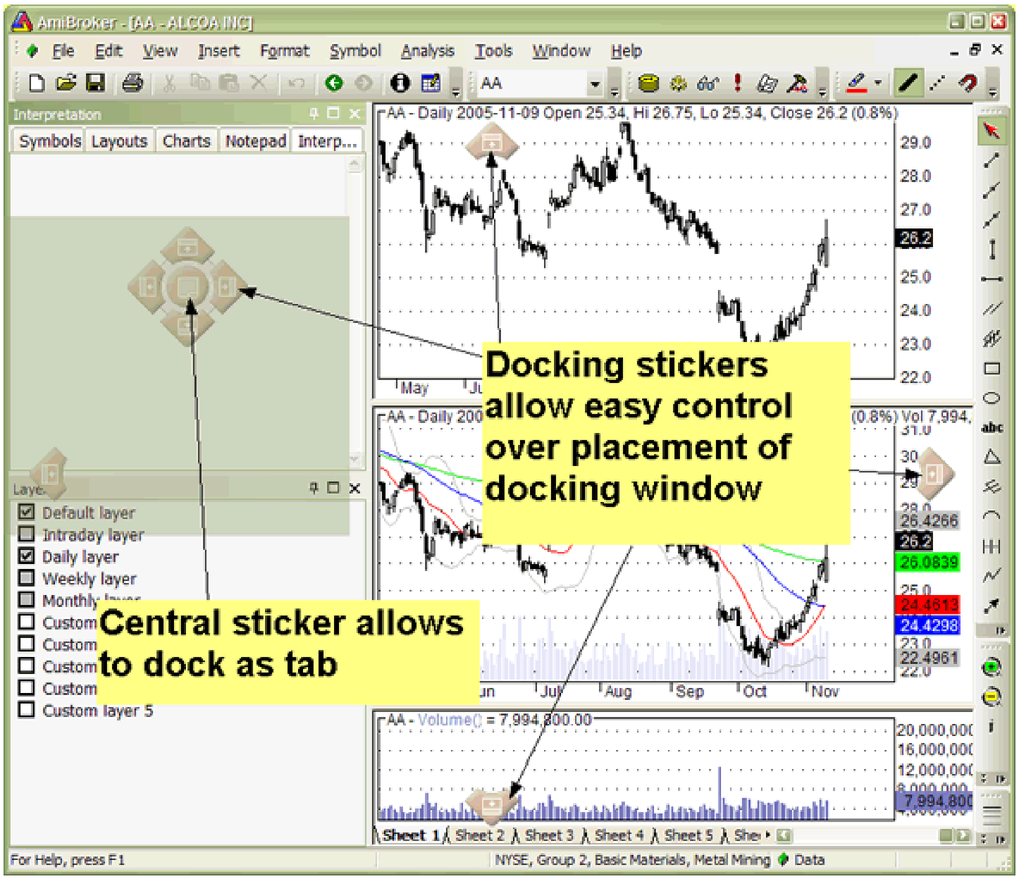
User interface customization

A newly introduced customizable user-interface has several nice features that allow complete control over look and feel of AmiBroker user interface.

Advanced nested docking / tear-off tabs


Panes can be docked in any way including nested groups
Charts
Symbols
Layouts
Charts
Averages
DEMA - Double Exponential Moving Ave
EMA - Exponential Moving Average
Linear Regression - Least Squares Mo
MA - Simple Moving Average
TEMA - Triple Exponential Moving Ave
TSF - Time Series Forecast
Wilders - Wilders Moving Average
WMA - Weighted Moving Average
Bands
Basic Charts
Price
Notepad
Interpretation
Layers
Default layer
Intraday layer
Daily layer
Weekly layer
Monthly layer
Custom layer 1
Sheet 1
Sheet 2
Sheet 3
Sheet 4
Sheet 5

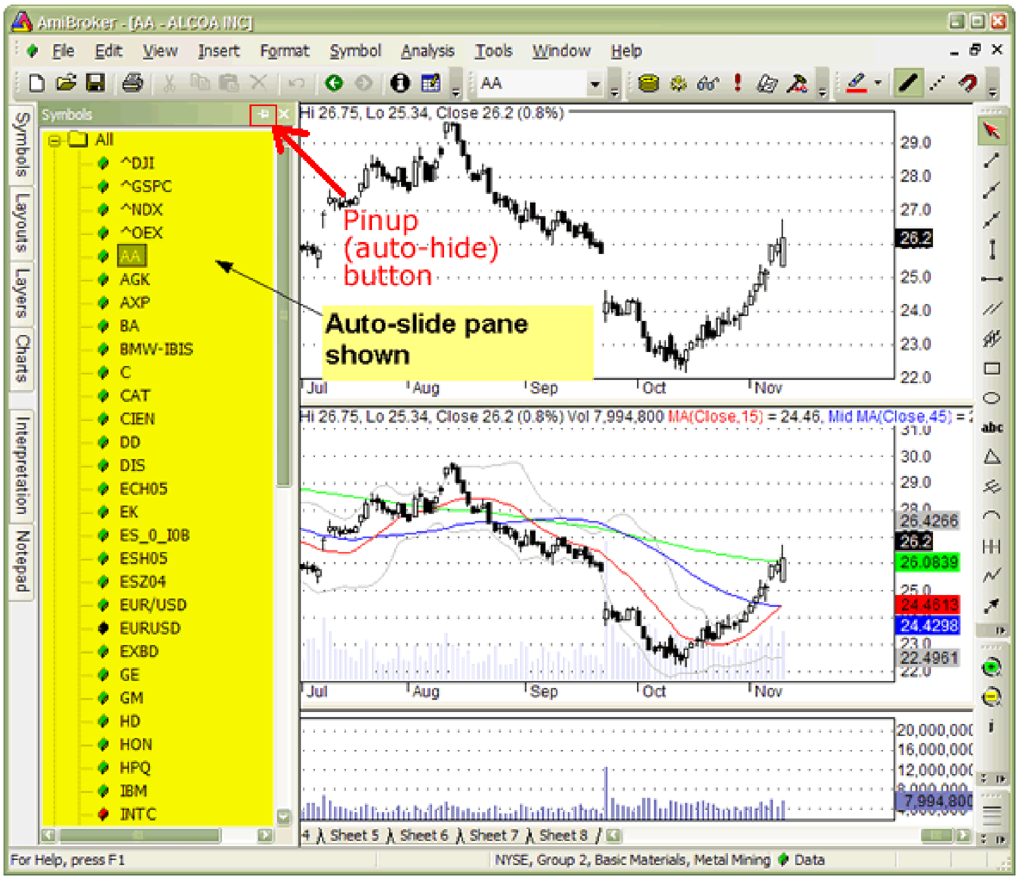
To dock a pane into any side of the application or as a tab simply click on docking window caption bar and drag it. If you do this, docking stickers will show up to make it easy to choose destination place as shown below

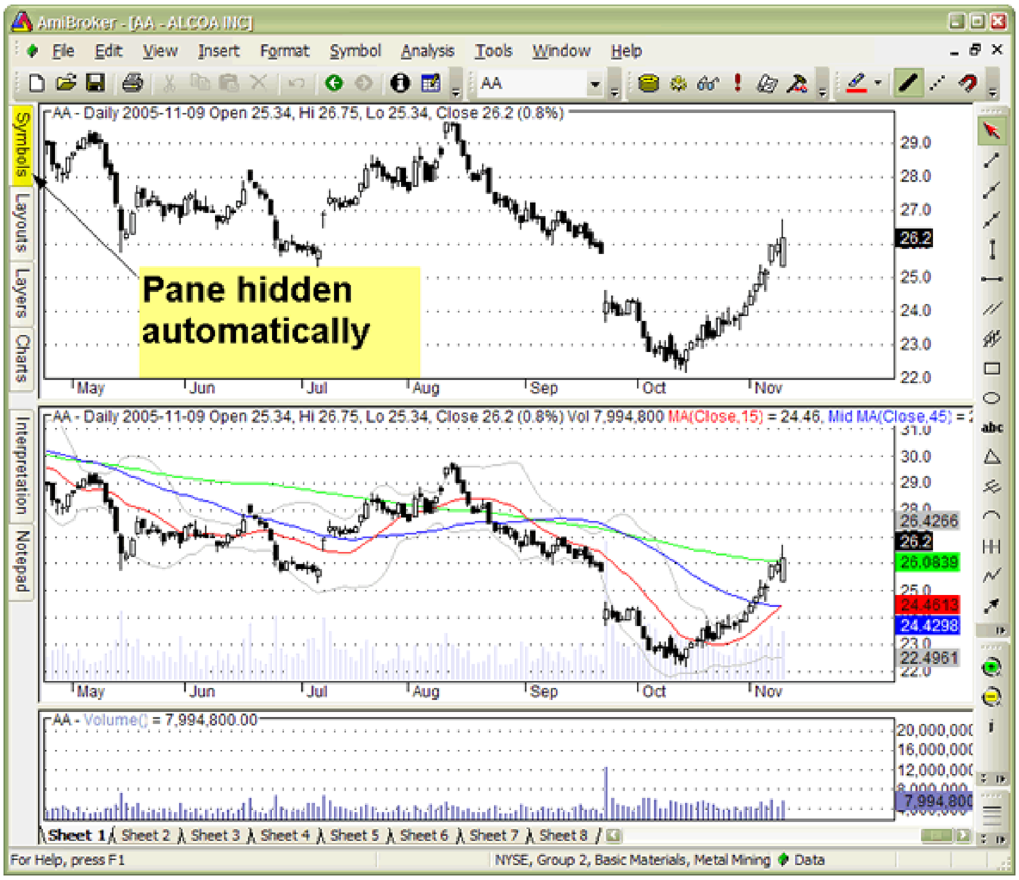

AmiBroker - [AA - ALCOA INC]
File Edit View Insert Format Symbol Analysis Tools Window Help
Interpretation
Symbols Layouts Charts Notepad Interp...
Layers
Default layer
Intraday layer
Daily layer
Weekly layer
Monthly layer
Custom layer 5
AA - Daily 2005-11-09 Open 25.34, Hi 26.75, Lo 25.34, Close 26.2 (0.8%)
AA - Volume() = 7,994,800.00
Sheet 1 Sheet 2 Sheet 3 Sheet 4 Sheet 5
For Help, press F1
NYSE, Group 2, Basic Materials, Metal Mining Data
Docking stickers allow easy control over placement of docking window
Central sticker allows to dock as tab

You can also click on docking pane tab and drag it (tear off) and dock as a separate window. This way you can arrange all docking windows either as separate windows or as tabs or as a mixture of these two approaches. You can also make window / tab floating if you drag it while holding down CTRL key.

Sliding Auto-hide panes

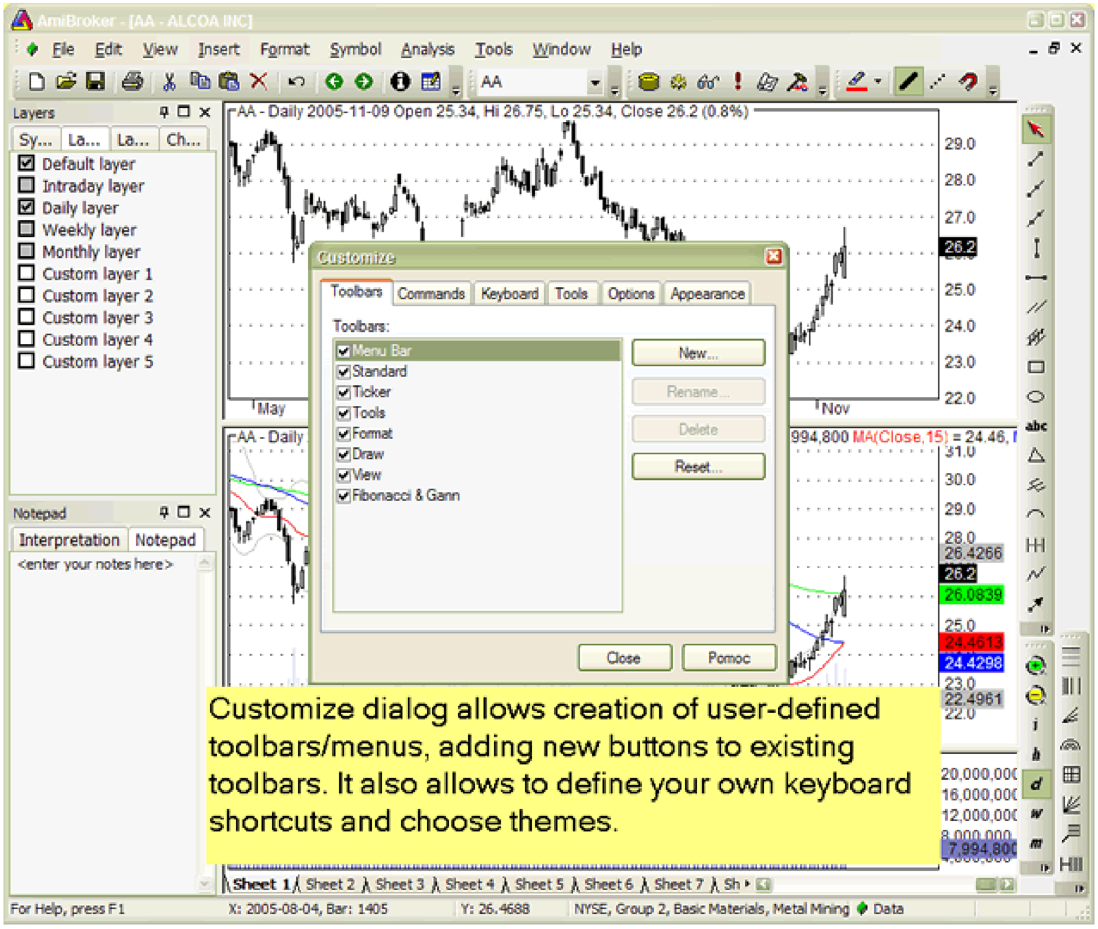
Another very useful feature that allows to conserve precious real estate on your monitor is auto-hiding of panes. To control (switch on/off) this feature there is a pinup button in the upper right corner of each docking window. If you unpin it - the pane will automatically hide when it loses focus.


AmiBroker - [AA - ALCOA INC]
File Edit View Insert Format Symbol Analysis Tools Window Help
Symbols
Pinup (auto-hide) button
Auto-slide pane shown
For Help, press F1
NYSE, Group 2, Basic Materials, Metal Mining Data


AmiBroker - [AA - ALCOA INC]
File Edit View Insert Format Symbol Analysis Tools Window Help
AA - Daily 2005-11-09 Open 25.34, Hi 26.75, Lo 25.34, Close 26.2 (0.8%)
Pane hidden automatically
For Help, press F1
NYSE, Group 2, Basic Materials, Metal Mining Data

Advanced customizable toolbars, menus and keyboard shortcuts

New user interface allows full user control over appearance, layout and position of all toolbars, buttons and menus. It allows you to add your own buttons, remove/re-arrange existing ones. Also you define or re-define new/existing keyboard shortcuts. All these customization features are available from **Tools->Customize** menu or from **Customize** chevron menu.

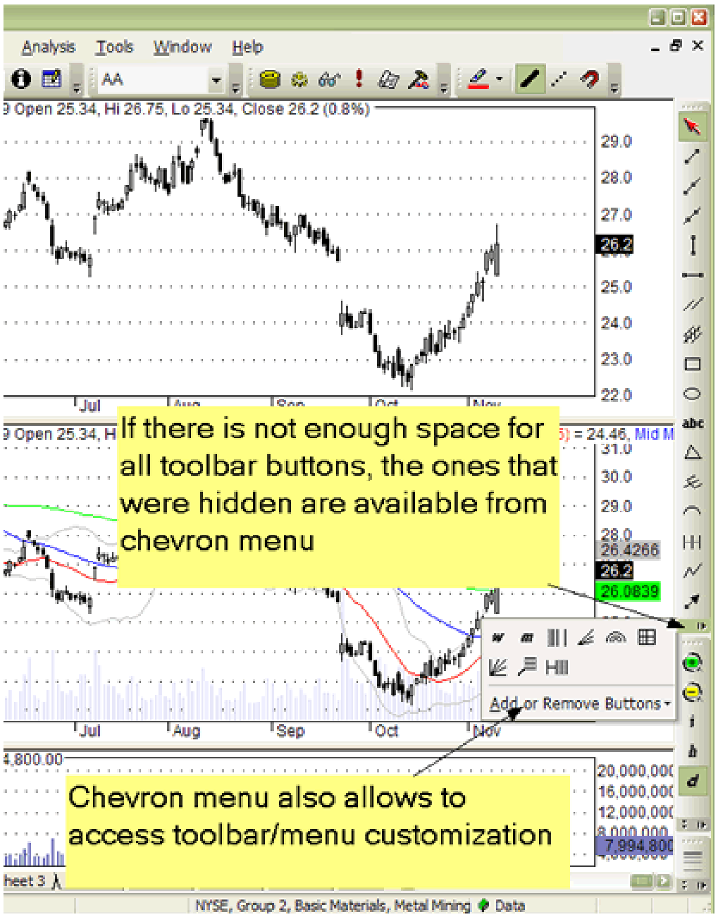


AA - Daily 2005-11-09 Open 25.34, Hi 26.75, Lo 25.34, Close 26.2 (0.8%)

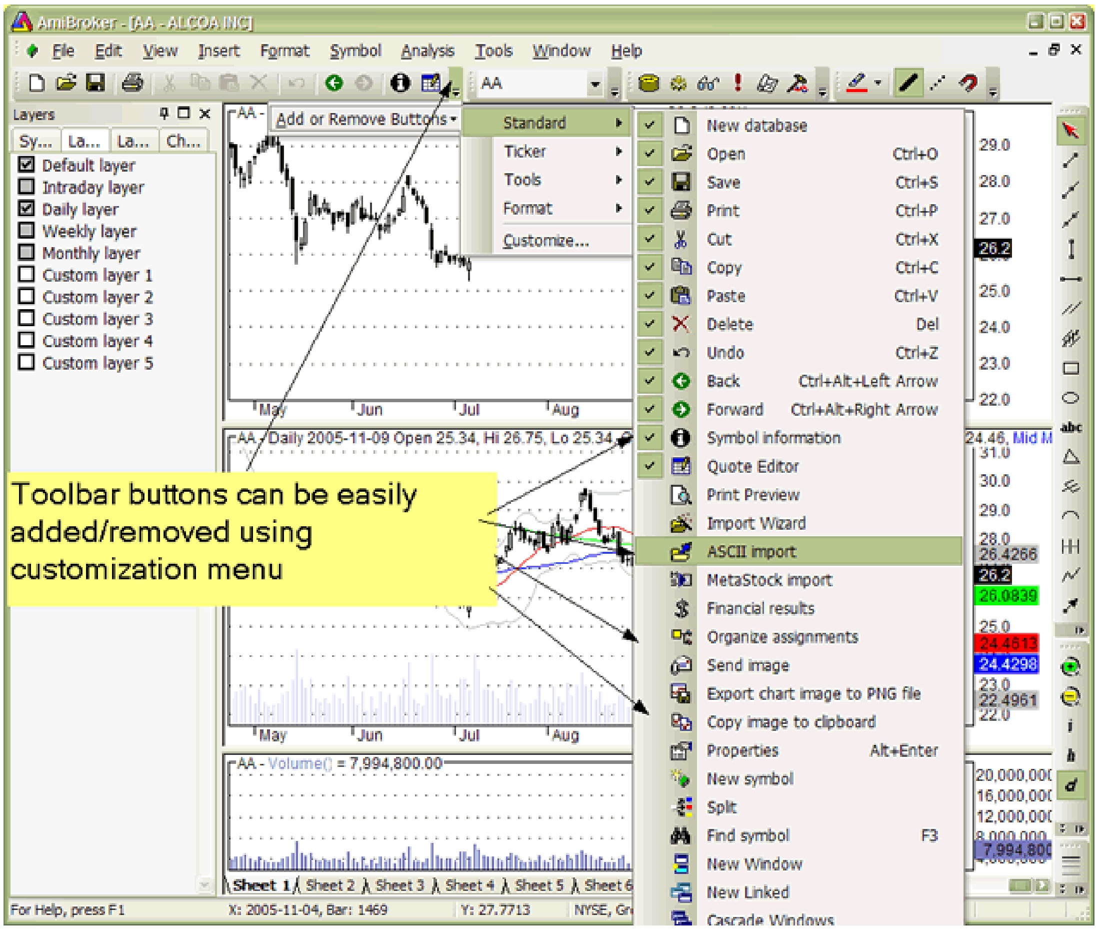
Customize dialog allows creation of user-defined toolbars/menus, adding new buttons to existing toolbars. It also allows to define your own keyboard shortcuts and choose themes.

X: 2005-08-04, Bar: 1405 Y: 26.4688 NYSE, Group 2, Basic Materials, Metal Mining Data

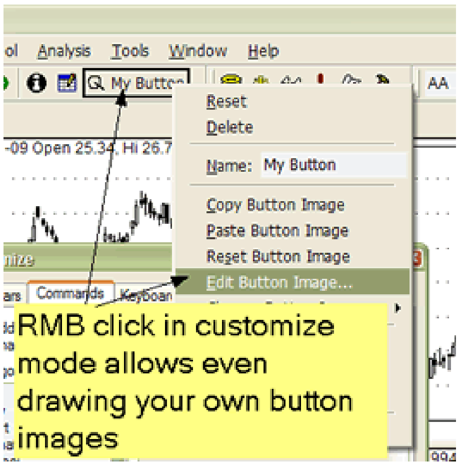
Chevron menu is available from little arrow button placed at the end of toolbar strip. It allows to access auto-hidden elements of the toolbar as well as customization features.

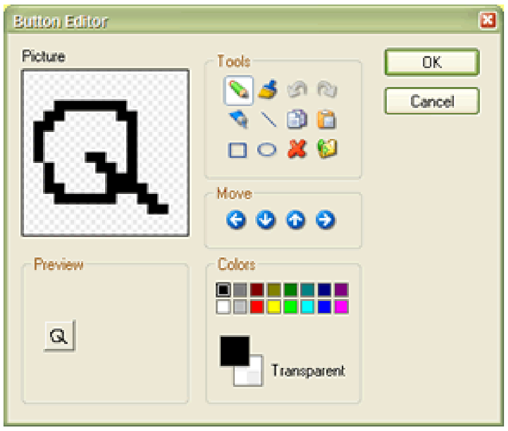

Analysis Tools Window Help
If there is not enough space for all toolbar buttons, the ones that were hidden are available from chevron menu
Add or Remove Buttons
Chevron menu also allows to access toolbar/menu customization
NYSE, Group 2, Basic Materials, Metal Mining Data

Add or Remove buttons submenu allows to quickly show / hide toolbar buttons according to your preference. In customization mode (when you enter it using Tools->Customize you can also move buttons around to change the order in which they appear, and you can also resize edit fields and combo fields (such as ticker selection field) by selecting them first and resizing the border that will show after making selection.



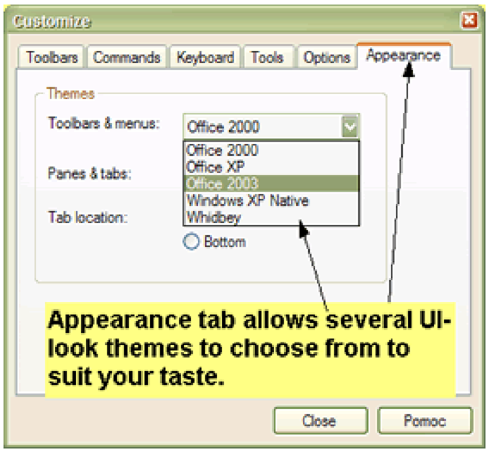
You can even add and design your own buttons using built-in image editor:





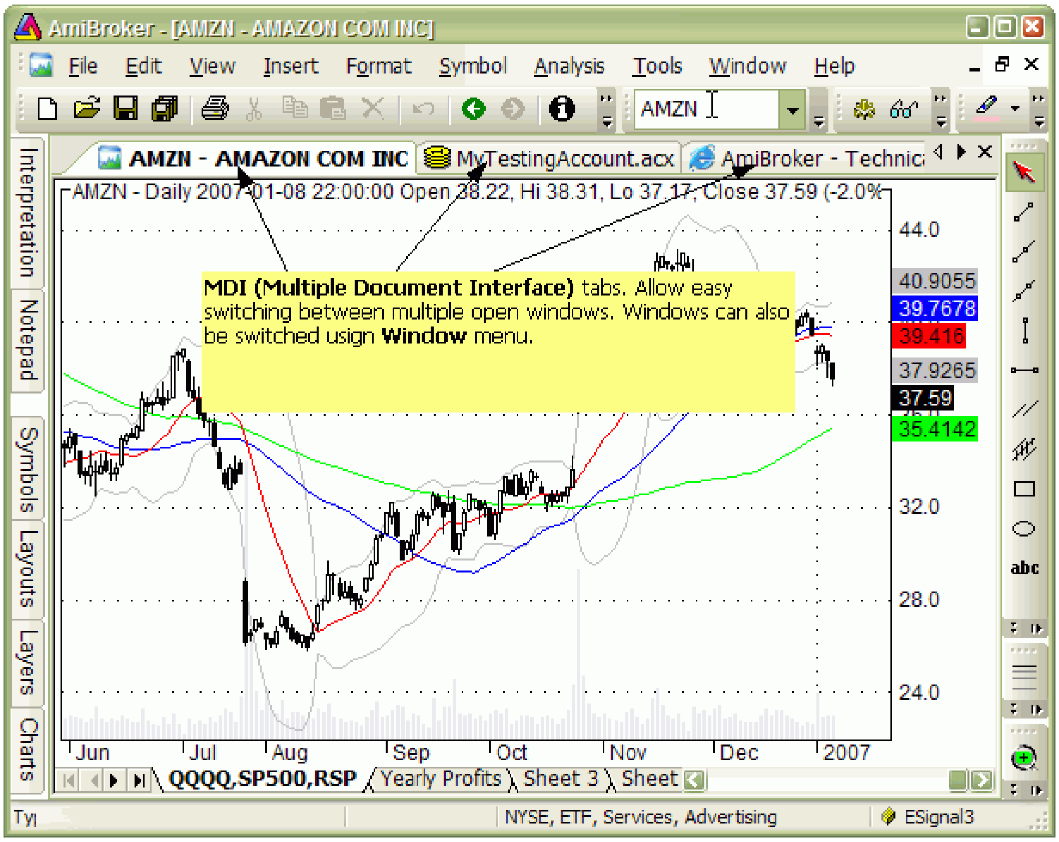
Themed appearance

AmiBroker allows also to pick your preferred user-interface "appearance" or "theme" to suit your personal taste.



Appearance tab allows several UI-look themes to choose from to suit your taste.

MDI (multiple document interface) tabs



MDI (Multiple Document Interface) tabs. Allow easy switching between multiple open windows. Windows can also be switched usign Window menu.

AmiBroker is multiple document interface (MDI) application. In short it means that it allows you to open and work with multiple windows at the same time. To learn more about what MDI is you may check this article: http://en.wikipedia.org/wiki/Multiple_document_interface

Now MDI tabs (shown in the picture above) are just an additional way to switch multiple open windows (in addition to **Window** menu where the list of open document windows is also available).

It is important to understand that MDI tabs are **not** "user definable" in the sense that you can not define their names freely, unlike chart sheets (which are definable). Their names are automatically derived from

document/window name. For chart windows the name is always in the format of: Symbol - FullName, web browser windows use HTML page title (as defined by HTML document), account manager windows use actual account file name (that you can choose when you save them).
MDI tabs are basically document window switcher (like Windows TASK BAR in the bottom) and they are automatically managed by AmiBroker whenever you open new or close window.

And it works exactly using the same idea as Windows task bar. Let us look at this analogy closer:

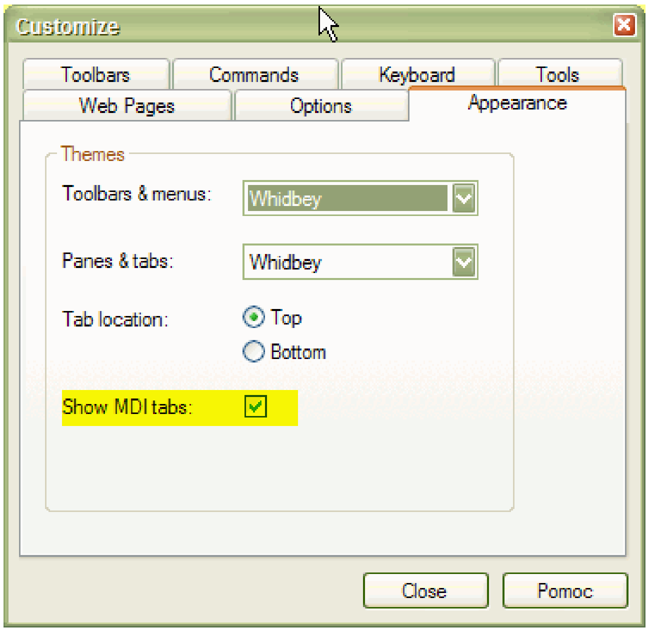
When you use Windows Task Bar:

- you open the **application** - a new button in the task bar appears
- and you can switch between open **applications** using task bar buttons.
- you can not rename the button because it represents **application** name.
- and you need to be careful with opening too many applications because all open applications consume system resources

Now using AmiBroker MDI tabs:

- you open the document (**window**) -> a new button (tab) appears
- you can switch between open **windows** using buttons (tabs)
- you can not rename the button because it represents document/**window** name
- and you need to be careful with opening too many documents/**windows** because all open documents consume system resources

You can **turn off** MDI tabs by unchecking "Show MDI tabs" box in the Tools->Customize, Appearance page, as shown below:



Toolbars & menus: Whidbey
Panes & tabs: Whidbey
Tab location: [✓ Top] Bottom
Show MDI tabs: ✓

Historical note: In pre-4.90 versions, to switch the documents you would need to use Window menu. Now in addition to that you can use tabs. But this is just convenience feature, more info at: http://en.wikipedia.org/wiki/Tabbed_Document_Interface (Note that wikipedia links describing TDI / MDI are somewhat outdated and AmiBroker actually combines advantages of BOTH TDI and MDI approaches (for example you can tile windows in AB's TDI)

For more information see Houston conference presentation: http://www.amibroker.com/docs/Houston1.pdf (PDF format), http://www.amibroker.com/docs/Houston1.html (Flash format).

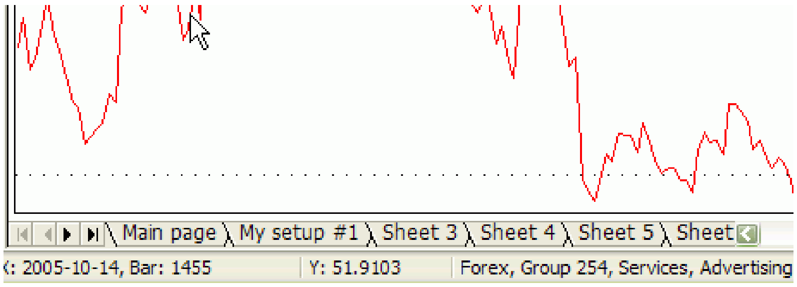
Working with chart sheets and window layouts

AmiBroker manages multiple chart sheets and **multi-window layouts** with ability to quickly load/save thems. This feature enables you to quickly switch between different indicator sets saving your time dramatically.

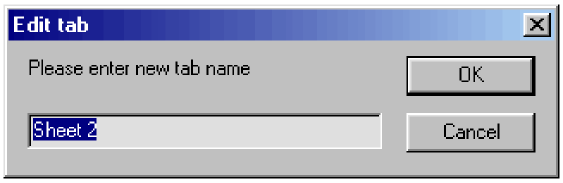
Chart sheets and templates

A chart sheet is a set of chart panes (with indicators) displayed within single frame.

You can switch between different sheets by clicking on the tabs located in the bottom of AmiBroker window as show in the following picture:

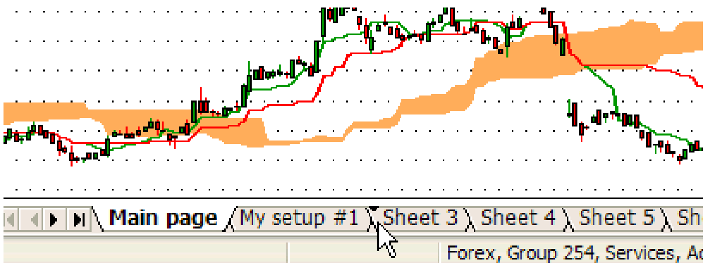


You can change the name of the tab by clicking on it with RIGHT mouse button, so the following window appears:

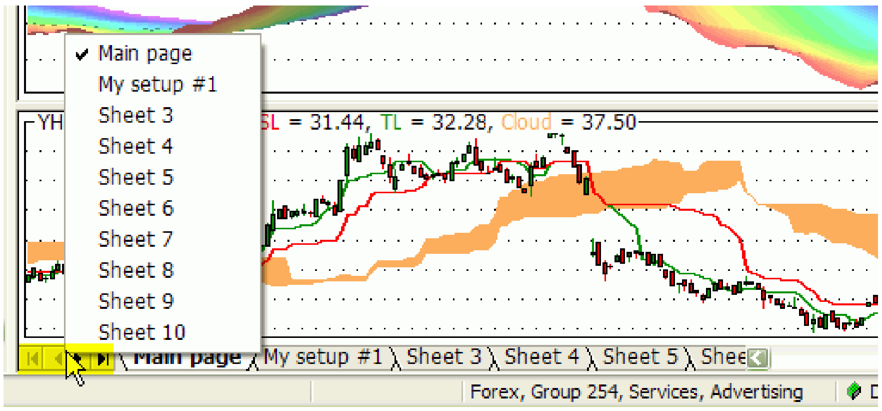


You can change all four tab names (one by one) so they are more descriptive (and they relate to the contents of the sheet).

You can scroll tabs using arrow buttons and you can re-arrange them by dragging (click on tab, hold down left mouse button and drag to desired position - an arrow will show target position).



You can also access any sheet quickly by clicking with RIGHT mouse button over arrows to pop-up the menu that lists all tabs and allows immediate selection (without scrolling)



The next step is to set up your sheets according to your personal preference. Just add/remove chart panes to/from each sheet. This way you can have upto 60 different indicator sets that you can recall very quickly by switching to appropriate tab. The actual number of sheets is definable in **Tools->Preferences->Charting "Number of chart sheets"**

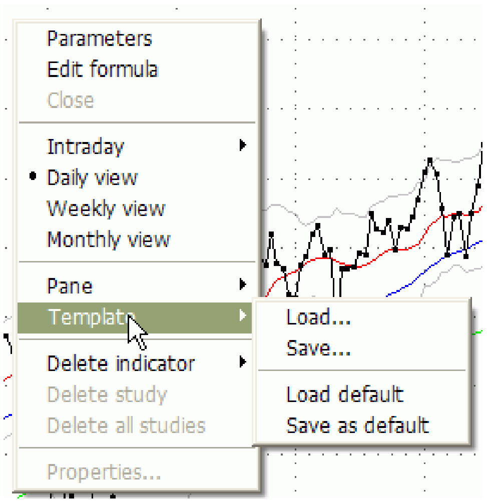
The complete set of chart sheets is called a "template" and you can make this setup permantent just right-click on the chart and select the following menu item (***Template->Save, Template->Save as default***):

The default template is used if you create a new window (**Window->New**)

You can also load once saved template by choosing **Template->Load** from chart's right mouse button menu.

In addition to old local template format a new one is added with .chart extension that keeps not only window sizes and formula references (paths) but also formulas themselves, so all you need to do is to save your chart into one file (Chart Template, Complete *.chart) and copy that file onto different computer and chart will be recreated with all formulas linked to it.

To Save chart into new format do the following:



1. Click with RIGHT MOUSE button over the chart and select **Template->Save...**

2. In the file dialog, **"Files of type"** combo select **"Chart Template, Complete (*.chart)"**

3. Type the file name and click **Save**.

To load previously saved complete chart do the following:

1. Click with RIGHT MOUSE button over the chart and select **Template->Load...**

2. In the file dialog, select previously saved *.chart file and press **"Open"**

Note: The procedure AmiBroker does internally is as follows: When you save the chart into new format it saves XML file with:

a) names of all sheets, panes, their sizes, locations and other settings
b) paths to all formulas used by all panes
c) the text of formulas themselves

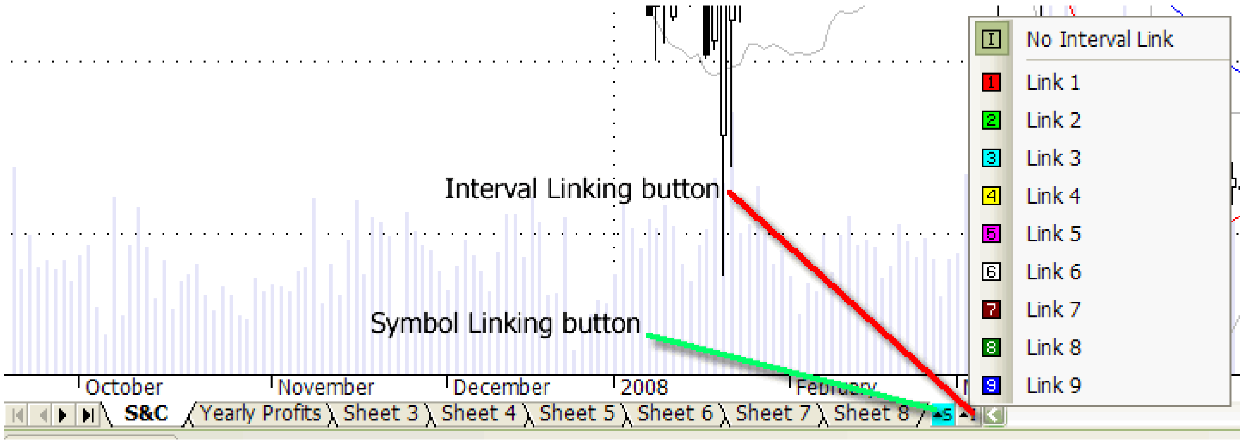
When you load the chart in new format AmiBroker:
a) sets up the sheets/panes according to information stored in the file
b) for each formula stored in the file it checks if the same formula exists already on target computer:
- if it does not exist - it will create one
- if it exists and the contents is identical to the formula stored in .chart file it will do nothing
- if it exists and the contents is different then it will create NEW formula file with _imported.afl suffix (so old file is not touched) and will reference the pane to the _imported.afl formula instead.

IMPORTANT NOTE: if you use any #include files AmiBroker will store the contents of include files as well inside chart file and will attempt to recreate them on target machine. Please note that in case of includes it will check if it exists and if it is different. If both conditions are met (different file exists already) it will ask to replace or not. If you choose to replace - it will replace and make backup of existing one with .bak extension. If you are using any files in "standard include files" and include them using <> braces, AmiBroker will restore files in target machine standard include folder as well (even if the standard include folder path is different on the source machine).

A new .chart format is intended to be used to port charts between different computers. For storing layouts/templates on local computer you should rather use old formats as they consume much less space (they store only references,not the formulas themselves). One may however use new format for archiving purposes as it keeps formulas and all references in one file that is very convenient for backups.

Symbol and Interval Linking

Now it is possible to link chart windows either by symbol and/or by time interval. To link chart windows use linking buttons located in the bottom of the chart window as shown in the picture below:

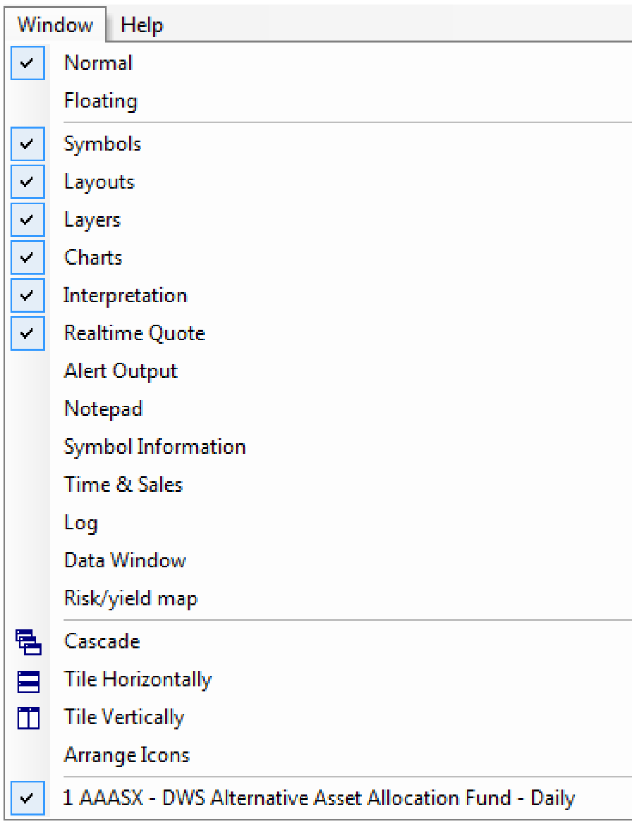


Grey "S" and "I" buttons mean no link. Any other color (red, green, magenta, yellow, pink, white, brown, dark green, blue) means that given chart belong to given color-coded linked group. All windows with same color link will switch symbol and/or interval simultaneously.

Floating windows

If you are using multiple monitors, you can find it useful to display AmiBroker charts on multiple windows. To make it easy, AmiBroker 5.10 introduces "floating" chart windows. Normally all chart windows live inside main AmiBroker application window. If you make chart window floating, you are essentially detaching the chart window from parent AmiBroker frame, so you can move it outside, for example to the other monitor.

You can switch between normal and floating state using Window menu as shown below:



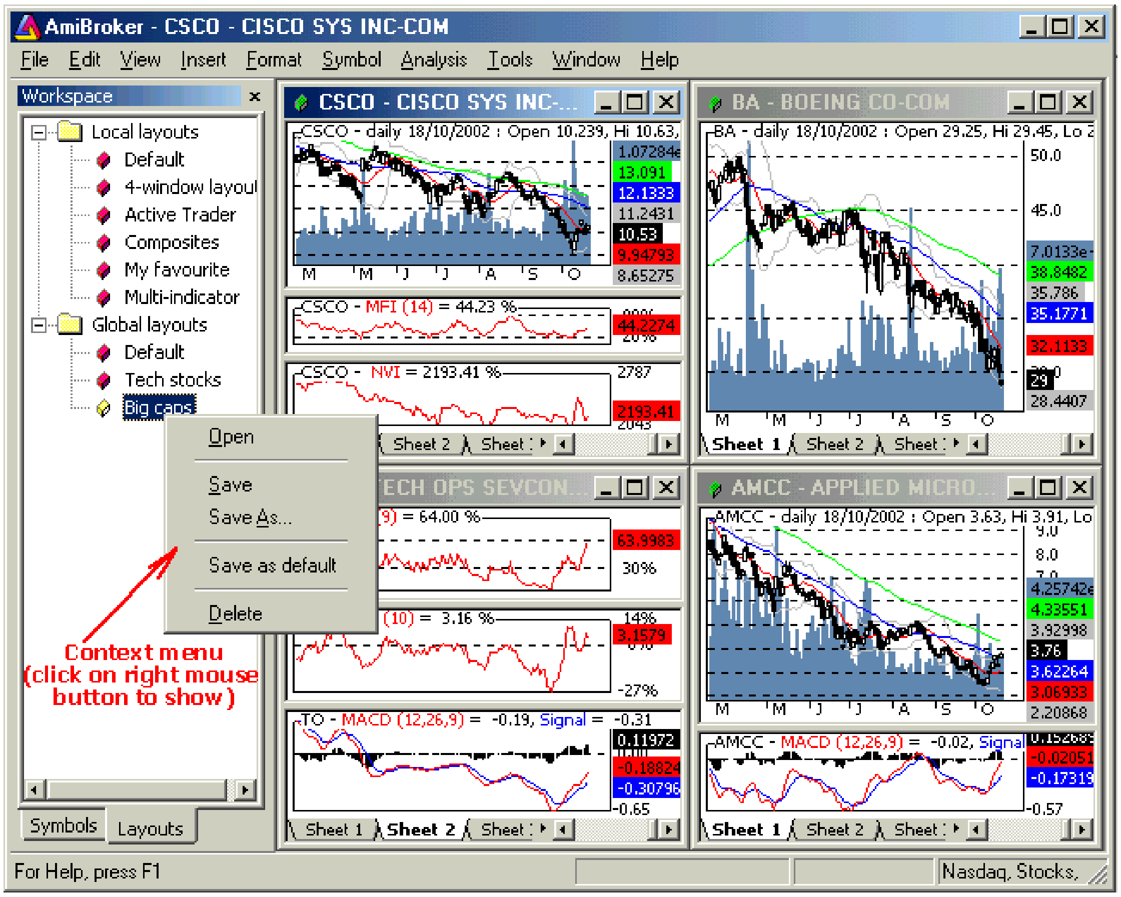
The following video tutorial shows how to use floating windows and symbol linking in practice:

http://www.amibroker.com/video/FloatAndLink.html

Window layouts

A window layout is a complete set of multiple windows open each with different symbol, different display interval, different size, different set of chart sheets.

The picture below shows 4-window layout each with different set of indicator panes. To the left you can see "Layouts" pane in the Workspace window showing the list of stored local and global layouts.



Using AmiBroker 4.20 you can now have unlimited number of custom, multiple-window templates that can be switched between with just double click on layout name in the "**Layouts**" tab of the Workspace window.

You can **open**, **save**, **delete** layout by clicking on the **Layout** tree with right mouse button and choosing appropriate function. "**Save As**" option saves current layout under new name.

Local layouts are per-database while **Global layouts** are visible from all databases.

Information saved in layouts include: window sizes and postions, maximized/minimized state chart panes available on each sheet (independent for each window), selected bar interval, selected symbol, selected chart sheet

Most recently used layout can be saved on exit and database switch automatically (see: **Tools->Preferences->Miscellaneous** "Save on exit: Layouts")

Note: since version 4.90 multiple windows can be switched not only using old-style Window menu but also using new MDI tabs. More on MDI tabs can be found in the "User-interface customization" chapter.

Using layers

What layers are

Layers are like pieces of transparent plastic. You can put drawings on them. Layers can be made visible or invisible. This allows to show/hide drawings placed on given layer without affecting the drawings placed on other layers.

How to work with layers.

First of all make sure that Workspace window is visible (Window->Layers)

Then switch to "Layers" tab. Here you can see the list of pre-defined layers.

The checkboxes on the left side of each layer control layer visibility. If checkbox is marked than given layer is visible, if it is unmarked - the layer is invisible. Initially first five layers will be "locked" to intervals.
These built-in layers are:

Default layer - always visible
Intraday layer - visible only when viewing intraday charts
Daily layer - visible only when viewing daily charts
Weekly layer - visible only when viewing weekly charts
Monthly layer - visible only when viewing monthly charts

A locked layer changes its visibility automatically when interval changes and you can not change its visibility by clicking on the left-side checkbox.

The remaining layers are not locked and they can be shown/hidden freely by marking the checkbox.

To draw a study in a given layer simply
a) SELECT the layer first (click on name to highlight it)
b) DRAW the study as usual

As long you select the other layer all drawings will be placed on selected layer. After drawing a study you can assign it to any other layer via object properties box.

Context menu

If you click on layer name with right mouse button you will see the context menu containing the following options:

Add layer
Remove layer
Show all layers
Hide all layers
Toggle
Unlock built-in layers
Lock built-in layers
Properties.

Add/Remove layer are self-explanatory. Please note that you can not remove first 5 (built-in) layers

Show all/Hide all - shows and hides all NOT LOCKED layers
Toggle - toggles visibility of all NOT LOCKED layers

Unlock/Lock built-in layers - allows you to unlock/lock 5 first (built-in) layers. Once layer is unlocked its visibility does not change automatically when interval changes and you can show/hide it manually.

Properties - this launches properties box that allows you to rename layer and decide if given layer should or should not be locked to interval displayed.

If you mark "Lock visibility to interval" box the layer will show/hide automatically depending on what interval is
currently displayed. You can define visibility for **each** layer using "Interval" combo and "Show/hide automatically" buttons. Note that there is a *separate* visibility setting for EACH interval. The layer properties box ALWAYS shows "monthly" interval at start but this is just a startup condition you just switch to particular interval

and modify visibility. To setup locked layer completely you have to set visibility for **every layer listed** in the "Interval" combo-box. Simply select the interval and choose if layer should be shown or hidden for this interval, select next interval and again choose show or hide, select next and so on...until you define visibility for all intervals.

Using Web Research window

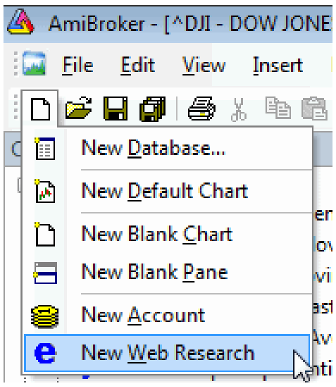
Web Research window allows you to view on-line news, research, profiles, statistics and all kind of information related to currently selected symbol available over the Internet (World Wide Web). Using Web Research instead of plain web browser has speed advantage as you don't need to type complicated/long addresses (URLs) each time you need to get desired information.

Web Research window introduced in version 4.90, replaces and enhances previously available Profile window. Now it allows unlimited number of user-definable web research (profile) pages, browsing to any web page (just type URL), tab-browsing, opening multiple pages at once, selective auto-synchronization.

Web-Reasarch uses Internet Explorer engine so you can be sure that pages are rendered with the same quality you would get from stand-alone browser.

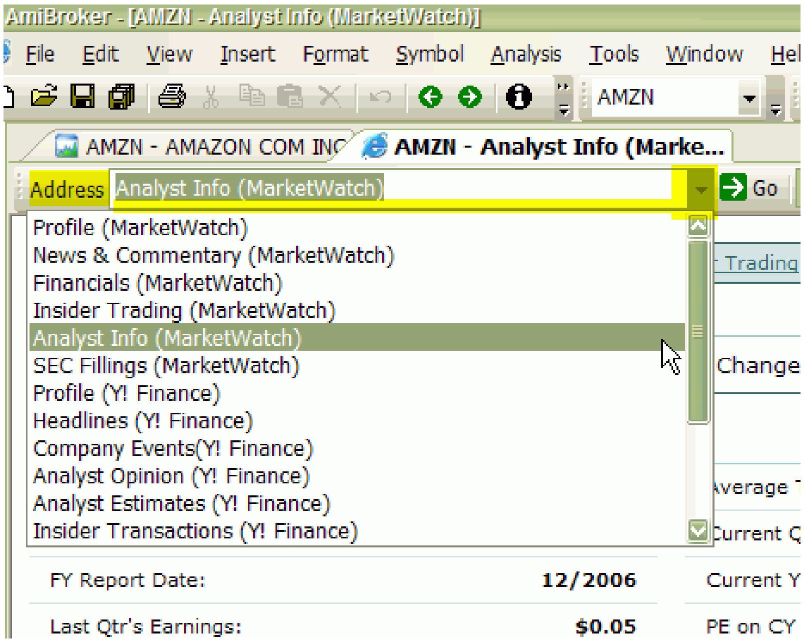
OPEN NEW WEB RESEARCH WINDOW

Use **File->New->Web Research** menu to create new web research window

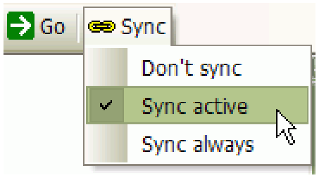


PICKING PRE-DEFINED WEB RESEARCH PAGE:

To display any pre-defined web research page, simply click on the drop down arrow in the Address combo-box and pick one item from the list. Once you do so, the web page relevant to currently selected symbol will be automatically displayed.


AmiBroker - [AMZN - Analyst Info (MarketWatch)]
File Edit View Insert Format Symbol Analysis Tools Window Hel
AMZN
AMZN - AMAZON COM INC
AMZN - Analyst Info (Marke...
Address
Analyst Info (MarketWatch)
Go
Profile (MarketWatch)
News & Commentary (MarketWatch)
Financials (MarketWatch)
Insider Trading (MarketWatch)
Analyst Info (MarketWatch)
SEC Fillings (MarketWatch)
Profile (Y! Finance)
Headlines (Y! Finance)
Company Events(Y! Finance)
Analyst Opinion (Y! Finance)
Analyst Estimates (Y! Finance)
Insider Transactions (Y! Finance)
FY Report Date: 12/2006
Last Qtr's Earnings: $0.05

Now you can specify if and when displayed page should change automatically if you select different symbol.

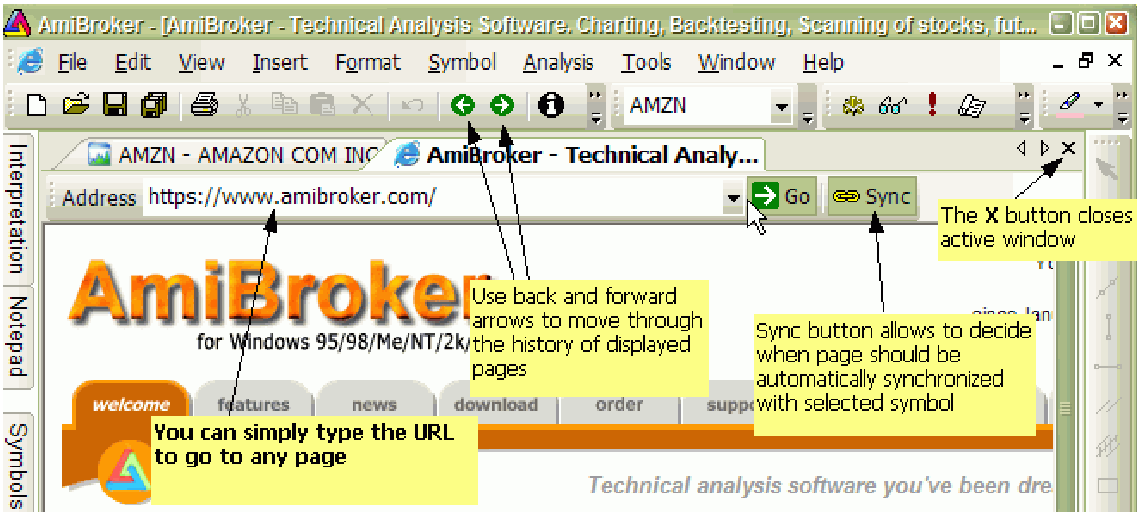


The **Sync** button allows to decide when page should be automatically synchronized with currently selected symbol.

- **Don't sync** - means that page should not be synchronized with currently selected symbol at all
- **Sync active** - means that page should be synchronized ONLY when it is currently active or becomes active (by user clicking on given tab) - this is recommended setting for web-research profiles since it conserves bandwidth and resources (not active pages are not synchronized and do not consume any bandwidth)
- **Sync always** - means that page is synchronized with currently selected symbol always, no matter if it is active or not.

NAVIGATION

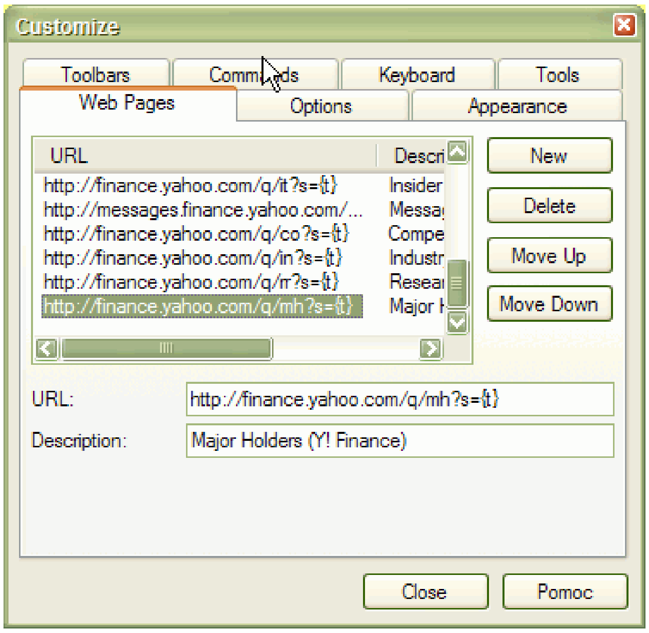
Web Research window operates in a way very similar to stand-alone browser. To display any web page just type the URL address to "Address" field and press ENTER (RETURN) key. To navigate back and forward in the history use **<-** and **->** buttons.


AmiBroker - [AmiBroker - Technical Analysis Software. Charting, Backtesting, Scanning of stocks, fut...
File Edit View Insert Format Symbol Analysis Tools Window Help
AMZN
AMZN - AMAZON COM INC
AmiBroker - Technical Analy...
Address https://www.amibroker.com/
Go
Sync
The X button closes active window
Use back and forward arrows to move through the history of displayed pages
Sync button allows to decide when page should be automatically synchronized with selected symbol
You can simply type the URL to go to any page
Interpretation
Notepad
Symbols

To close currently displayed page use regular window close **X** button as shown in the picture above

DEFINING YOUR OWN WEB RESEARCH PLACES

In addition to web-research pre-defined pages you can define any number of your own places. To do so use **Tools->Customize** menu, **Web Pages** tab.



URL	Descri
http://finance.yahoo.com/q/it?s={t}	Insider
http://messages.finance.yahoo.com/...	Messa
http://finance.yahoo.com/q/co?s={t}	Compe
http://finance.yahoo.com/q/in?s={t}	Industr
http://finance.yahoo.com/q/r?s={t}	Resea
http://finance.yahoo.com/q/mh?s={t}	Major I

URL: http://finance.yahoo.com/q/mh?s={t}

Description: Major Holders (Y! Finance)

To add new place press **New** button, then type the URL template in the **URL** field and web page description in the **Description** field.

The URL template is the web address in that has parts that depend on selected symbol. The URL template is parsed by AmiBroker to make actual URL to the web page. For example to see Yahoo's profiles page you can use following URL template:

http://biz.yahoo.com/p/**{t0}**/**{t}**.html.

Symbols enclosed in brackets **{}** define fields which are evaluated in execution time. **{t0}** symbol is evaluated to the first character of the ticker name and **{t}** is evaluated to the whole ticker name. So if AAPL is selected AmiBroker will generate following URL from above template:

http://biz.yahoo.com/p/a/aapl.html

Then AmiBroker uses built-in web browser (Web Research window) to display the contents of the page.

Special fields encoding scheme

As shown in above example template URL can contain special fields which are substituted at run time by values corresponding to the currently selected symbol. The format of the special field is {*x*} where *x* is describes field type. Currently there are three allowable field types: ticker symbol in original case **{t}**, ticker symbol in lowercase **{s}**, ticker symbol in UPPERCASE **{S}**, alias **{a}**, web id **{i}**. You can specify those fields anywhere within the URL and AmiBroker will replace them with appropriate values entered in the Information window. You can also reference to single characters of ticker, alias or web id. This is useful when given web site uses first characters of, for example, ticker to group the html files (Yahoo Finance site does that), so you have files for tickers beginning with 'a' stored in subdirectory 'a'. To reference to single character of the field use second format style {*xn*} where *x* is field type described above and *n* is zero-based index of the character. So **{a0}** will evaluate to the first character of the alias string. To get first two characters of a ticker write simply **{t0}{t1}**. Note about web id field: this new field in Information window was added to handle situations when web sites do not use ticker names for storing profile files. I found some sites that use their own numbering system so they assign unique number to each symbol. AmiBroker allows you to use this nonstandard coding for viewing profiles. All you have to do is to enter correct IDs in Web ID field and use appropriate template URL with **{i}** keyword.

Pages stored locally

You may want to have all pages stored on your local hard disk. This has an advantage that profiles are accessible instantly but they can take significant amount of storage space and you will need to update them from time to time. To access locally stored files use the following template URL (example, C: denotes drive): file://C:\the_folder_with_profile_files\{t}.html. You are not limited to HTML files, you can use simple TXT files instead. Then create (or download) the .html (or txt) files for each symbol in the portfolio. These files should obey the following naming convention: <ticker>.html. So for example for APPLE (ticker AAPL) the profile should have the name AAPL.html (or AAPL.txt)

Web-based profiles

If you want to display the profiles from remote web pages you will need to find out how they are accessible (the URL to the web page) and how the data for different symbols are accessible. I will describe the problem on the example of Sharenet (www.sharenet.co.za) site providing the data for companies listed on Johannesburg Stock Exchanges. Sharenet provides company information that is accessible at the following address (URL):

http://www.sharenet.co.za/free/free_company_na.phtml?code=**JSECODE**&scheme=default

The problem is that database provided by Sharenet uses long ticker names and **JSECODE** is a short symbol code. For example for "Accord Technologies" company the ticker in Sharenet database is ACCORD but the code is ACR. To solve the problem we will need to use **Web ID** field in the symbol Information window. If you have Sharenet database just choose the ACCORD from the ticker list, open *Symbol->Information* window and enter ACR to the **Web ID** edit box and click OK. Then enter the following URL template to the **URL** edit box:

http://www.sharenet.co.za/free/free_company_na.phtml?code={i}&scheme=default

To be 100% sure please select the text above with a mouse. Then copy it to the clipboard (Edit->Copy, CTRL-C). Then switch to AmiBroker and click on the Profile URL edit box. Delete everything from it and press CTRL-V (this will paste the text). Type "Sharenet" into **Description** field.

Please note that we have used **{i}** special field in the template that will be replaced by AmiBroker with the text entered in the Web ID field of the symbol information window. Now please select *File->New->Web Research* and pick Sharenet from Address combo box. You should see the profile for ACCORD company.

You can also delete any entry by selecting it from the list and pressing **Delete** button. You can change the order in which pages appear in the Web Research address combo using **Move Up** and **Move Down** buttons (select the item first and then use buttons).

Configuration data are stored in webpages.cfg plain text file that holds any number of URL templates in the form of:

URLTemplate|Description

(each entry in separate line)

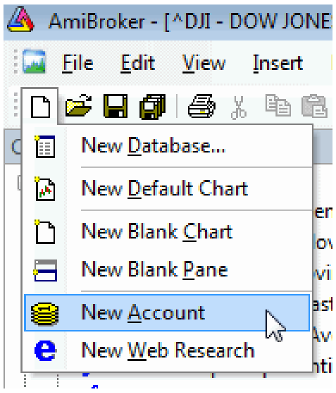
Using account manager

Account manager is a tool for keeping track of your trades and your performance. You are able to enter trades you make, deposit/withdraw funds, check the statistics and historical performance. All transactions are recorded so you will never forget what happened in the past. Account manager allows you to keep track of unlimited number of accounts.

New account manager replaces and enhances functionality provided by portfolio manager in pre-4.90 versions.

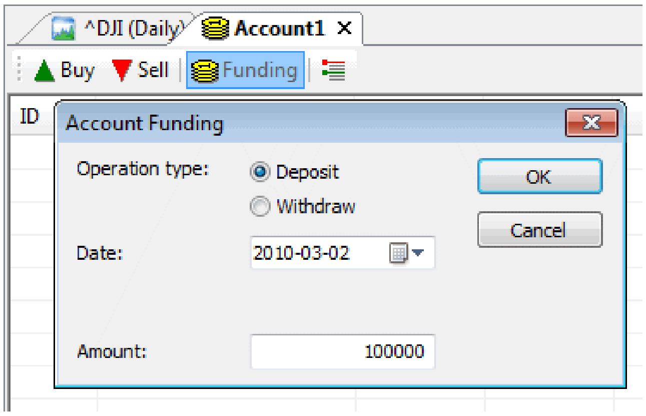
CREATE A NEW ACCOUNT

Use File->New->Account menu to create new account



FUNDING AN ACCOUNT

Before you do any trading, you have to fund your account. To do so press "FUNDING" button on the account manager toolbar, then select "Deposit" as operation type, enter the DATE when you have funded your account and enter the amount.
Note that funding date must PRECEDE any trading, as account manager won't allow you to trade prior to funding date. Initial deposit will show as "initial equity" in summary tab.



Operation type: [✓ Deposit] Withdraw

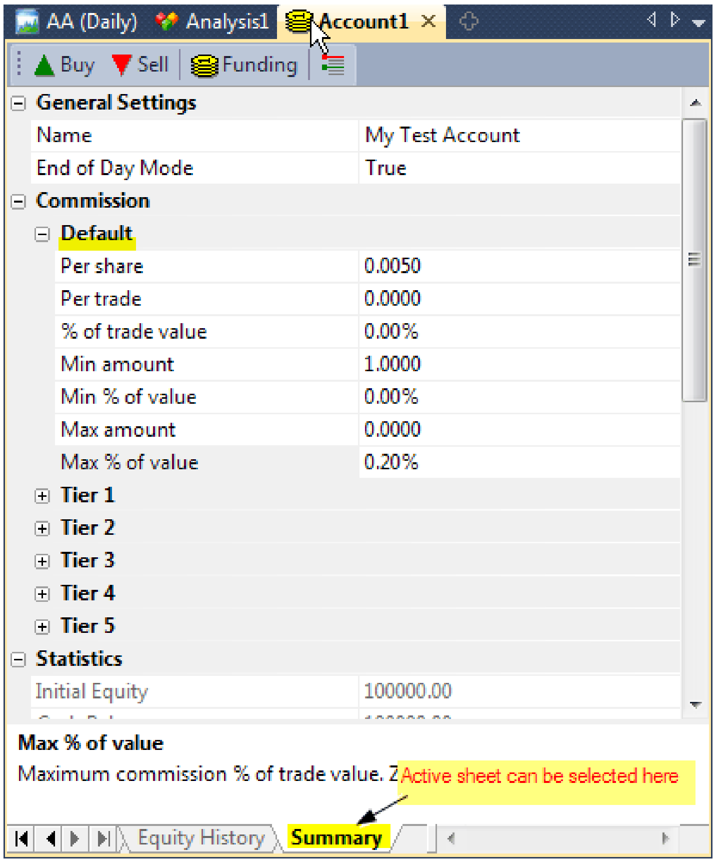
Date: 2010-03-02

Amount: 100000

THE SETTINGS

It is good idea to go to "Summary tab" and setup commissions and trading mode. If this account is used for End-of-day trading you should set "EOD Mode" to YES, otherwise (if you trade intraday) you should set "EOD

Mode" to NO. Depending on this setting Buy/Sell dialogs will allowyou to enter date and time of the trade or only date.



General Settings	
Name	My Test Account
End of Day Mode	True
Commission	
Default	
Per share	0.0050
Per trade	0.0000
% of trade value	0.00%
Min amount	1.0000
Min % of value	0.00%
Max amount	0.0000
Max % of value	0.20%
Tier 1	
Tier 2	
Tier 3	
Tier 4	
Tier 5	
Statistics	
Initial Equity	100000.00

Max % of value
Maximum commission % of trade value. Z

Commission table allows to enter both per-share (per-contract) commissions and commissions that are expressed as percent of trade value. Or a combination of both. You can also set minimums and maximums expressed in dollar amount and/or percent of trade value. For example if your broker may use 0.01$ (one cent) per share commission, then you would use PerShare = 0.01 and %OfTradeValue = 0. If your broker uses say 0.2% of trade value then you would use PerShare = 0 and %OfTradeValue = 0.2;

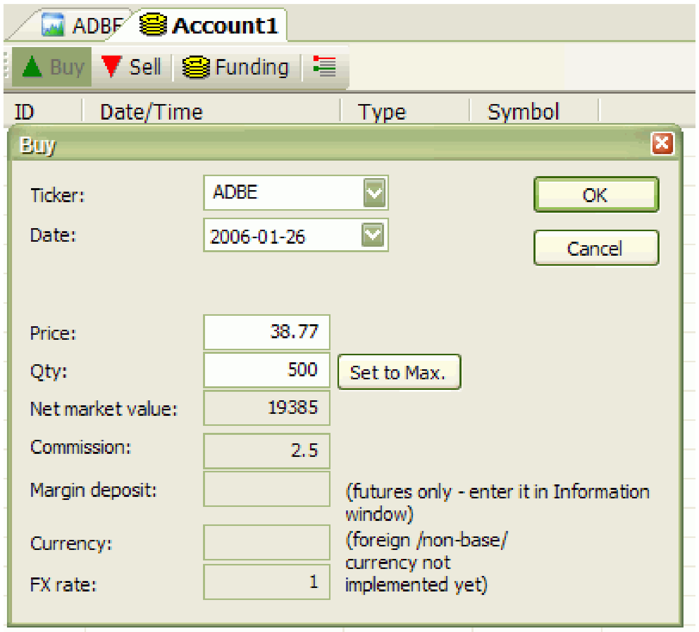
Practical example: Interactive Brokers default commission for U.S. stocks is: 0.005 per share but not less than 1 dollar and not more than 0.2% of trade value. Appropriate settings for such schedule are shown in the screenshot above.

Commission table works as follows: first sum of per-share commission and % of trade value is calculated. Then the result is checked against minimum and maximum limits and if calculated value exceeds the limit then commission is set to value of such the limit, otherwise calculated value is used without change.

Summary page contains a little bit of basic statistics as well.

ENTERING TRADES

Once you funded an account you can enter trades.To buy (enter long position or cover short position) click on "BUY" button.



Ticker:	ADBE
Date:	2006-01-26
Price:	38.77
Qty:	500
Net market value:	19385
Commission:	2.5
Margin deposit:	(futures only - enter it in Information window)
Currency:	(foreign /non-base/ currency not implemented yet)
FX rate:	1

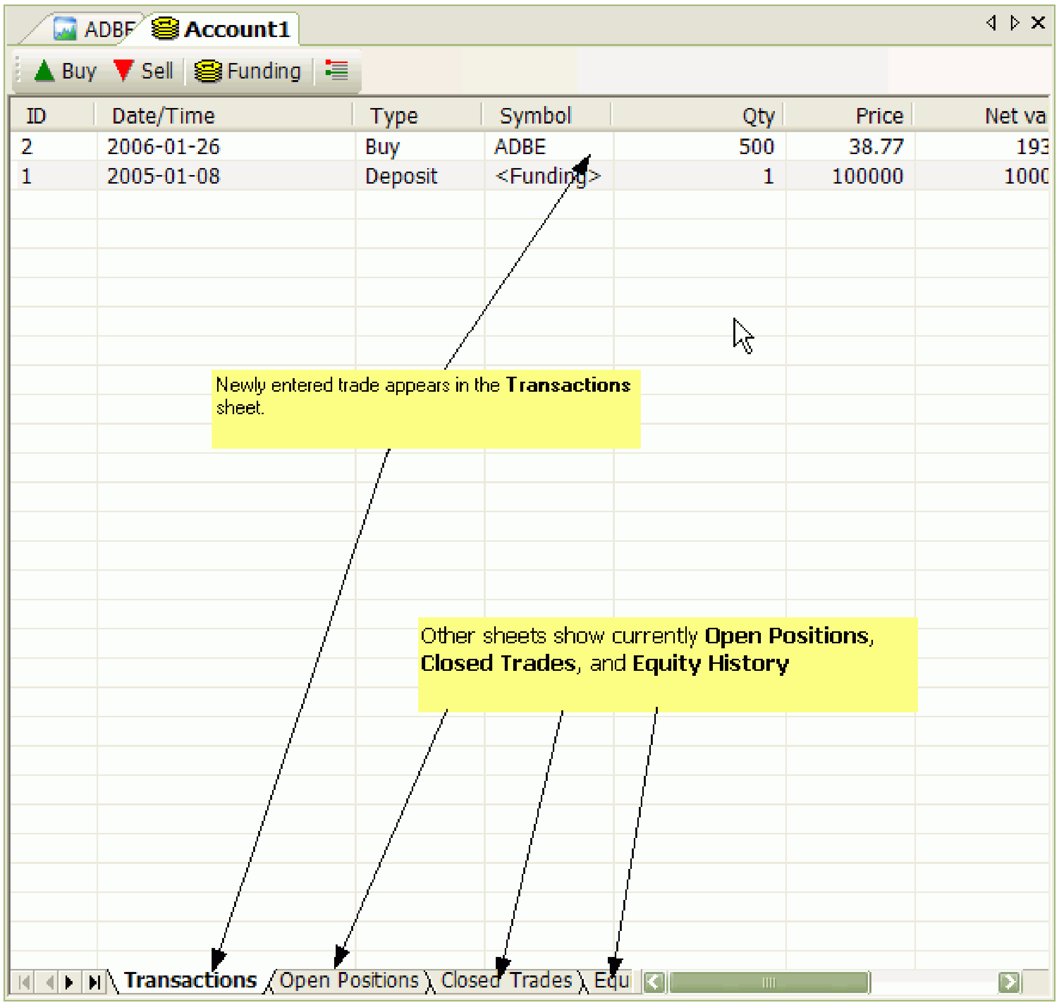
Then in the Buy dialog you need to select the symbol, the trade date/time. Once they are entered AmiBroker will display price of given symbol at the selected date/time (or preceding one if no exact match is found). It will also calculate maximum possible quantity taking price and available funds into account.

You can change the price and quantity manually.

All other values (net market valye, commission, market deposit, currency, fx rate) are calculated or retrieved automatically from Symbol->Information page. Once values are good, click OK to confirm transaction. If you made mistake, you can press UNDO (Edit->Undo) to revert last transaction.

Similar procedure is for selling (entering short positions or closing longs) with the exception that you should press "SELL" button instead.

All transactions that you made are listed in the "Transactions" sheet. All open positions are listed in "Open Positions" sheet. If you enter the trade for symbol that has position already open, AMiBroker will adjust "open positions" accordingly (perform scaling in/out). Once open position is closed it is removed from "open positions" list and moved to "Closed trades" sheet.



ID	Date/Time	Type	Symbol	Qty	Price	Net va
2	2006-01-26	Buy	ADBE	500	38.77	193
1	2005-01-08	Deposit	<Funding>	1	100000	1000

Newly entered trade appears in the **Transactions** sheet.

Other sheets show currently **Open Positions**, **Closed Trades**, and **Equity History**

After each transaction, "Equity history" sheet is updated with current account equity value and also "Summary" page is updated with basic open/long/short trade stats.(More stats are to come).

IMPORTANT

You have to remember that you must enter all transactions in chronological manner(oldest first, newest last), as account manager won't allow you to add trades out-of-order. If you make mistake, there is one-level undo that you can use to revert to state before last transaction. If you made more mistakes, the only option is to close account without saving and re-open original file.

SAVING YOUR ACCOUNT DATA

To save edits made to account use File->Save (or File->Save As to save under new name). Note that **account files are NOT encrypted now**, and it is quite easy to read the file for everyone who has the access to it. So make sure not to leave your files on some public computer. Password protection/encryption is planned but NOT implemented yet.

OPENING PREVIOUSLY CREATED ACCOUNT

To open account file, go to File->Open, in the File dialog, select "Account (*.acx)" from "Files of type" combo-box, and select the account file you want to load.

MULTIPLE ACCOUNTS

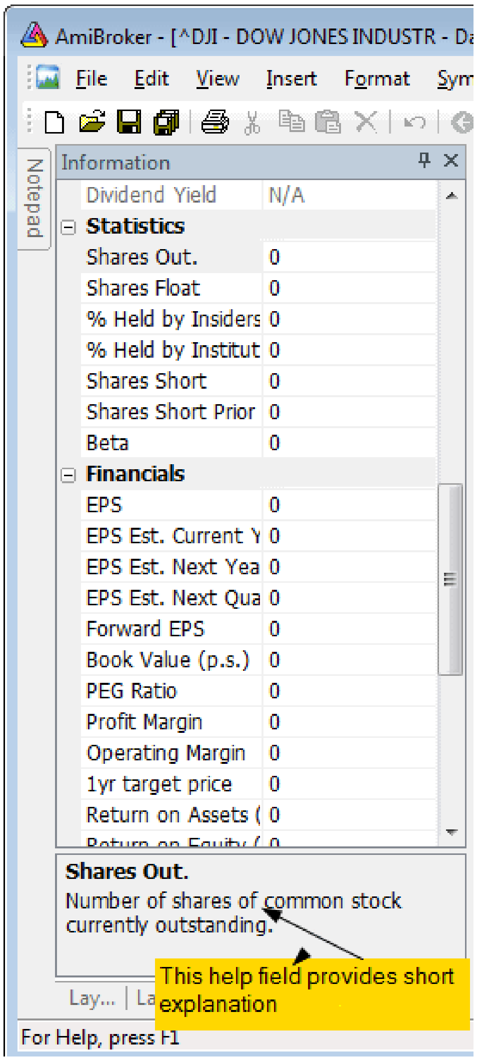
You can create/open multiple accounts at once (just use File->New->Account, File->Open many times).

Using fundamental data

AmiBroker 4.90 adds ability to use 32 fundamental data items. Fundamental data can be automatically downloaded for all U.S. stocks for free using AmiQuote. New Information window allows you to view these items, while new AFL function: GetFnData allows to access fundamentals programmatically.

INFORMATION WINDOW

To display fundamental data in Information window, please use **Symbol->Information** menu. This will open Information window with several fundamenta data fields as shown in the picture below (if you created new database, it probably will not have these data present initially and you would need to download them)



Dividend Yield	N/A
Statistics	
Shares Out.	0
Shares Float	0
% Held by Insiders	0
% Held by Institut	0
Shares Short	0
Shares Short Prior	0
Beta	0
Financials	
EPS	0
EPS Est. Current Y	0
EPS Est. Next Yea	0
EPS Est. Next Qua	0
Forward EPS	0
Book Value (p.s.)	0
PEG Ratio	0
Profit Margin	0
Operating Margin	0
1yr target price	0
Return on Assets (	0

Shares Out.
Number of shares of common stock currently outstanding.

This help field provides short explanation

DOWNLOADING FREE FUNDAMENTAL DATA FROM YAHOO

New version of AmiQuote now features ability to download free fundamental data from Yahoo Finance web site. This is implemented using 2 different Yahoo pages:

1. **Yahoo Fundamental - Basic** data source (free basic fundamental data, 200 symbols in one request). Data are retrieved from the following URL: http://finance.yahoo.com/q?s={Ticker} (Download data link).

That page provides the following data:

EPS (ttm)
EPS Est Current Year
EPS Est Next Year
EPS Est Next Quarter
PEG Ratio
Book Value
EBITDA
Sales Revenue
Dividend Pay date
Ex Dividend date
Dividend Per Share
1yr Target Price
Shares Float
Shares Outstanding

Explanation of values: http://help.yahoo.com/help/us/fin/quote/quote-03.html

2. **Yahoo Fundamental - Extra** data source (extended fundamental data, 1 symbol in one request, more data - available in registered version only).
   Data are retrieved from the following URL: http://finance.yahoo.com/q/ks?s={Ticker} (Key Statistics page)

   That page provides following data:

   Forward P/E
   PEG Ratio
   Profit Margin
   Operating Margin
   Return on Assets
   Return on Equity
   Revenue (ttm)
   Qtrly Revenue Growth
   Gross Profit
   EBITDA
   (Diluted) EPS
   Qtrly Earnings Growth
   Book Value Per Share
   Operating Cash Flow
   Levered Free Cash Flow
   Beta
   Shares Outstanding
   Float
   % Held by Insiders
   % Held by Institutions
   Shares Short (prior month)
   Shares Short
   Forward Annual Dividend Rate
   Trailing Annual Dividend Rate
   Dividend Date
   Ex-Dividend Date
   Last Split Factor

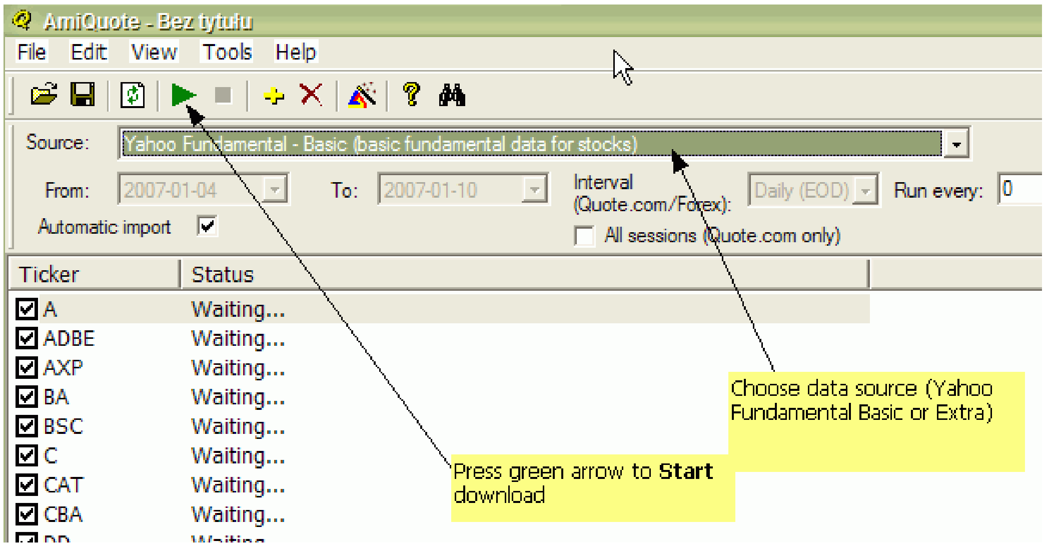
Last Split Date

Explanation of values: http://help.yahoo.com/help/us/fin/research/research-12.html

IMPORTANT NOTE: Unregistered version of AmiQuote allows you to download fundamental-ex data for first 20 tickers in the list. To download data for more symbols you need to register AmiQuote.

Downloading data is easy and staightforward:

1. Run AmiQuote
2. In AmiQuote, select **Tools->Get tickers from AmiBroker**
3. Select **Yahoo Fundamental - Basic** or **Yahoo Fundamental - Extra** from **Source** drop down list
4. Make sure that **Automatic import** box is checked
5. Press Green Arrow to **Start Download**


AmiQuote - Bez tytułu
File
Edit
View
Tools
Help
Press green arrow to **Start** download
Choose data source (Yahoo Fundamental Basic or Extra)

Source: Yahoo Fundamental - Basic (basic fundamental data for stocks)
From: 2007-01-04 To: 2007-01-10 Interval (Quote.com/Forex): Daily (EOD) Run every: 0
Automatic import ✓
All sessions (Quote.com only)

Ticker	Status
✓ A	Waiting...
✓ ADBE	Waiting...
✓ AXP	Waiting...
✓ BA	Waiting...
✓ BSC	Waiting...
✓ C	Waiting...
✓ CAT	Waiting...
✓ CBA	Waiting...

Once download is complete, you should see fundamental data updated in Information window in AmiBroker.

ACCESSING FUNDAMENTAL DATA FROM FORMULA (AFL) LEVEL

To access fundamental data from AFL level you can use new GetFnData function. It has quite simple syntax:

GetFnData("field")

where "field" is any of the following fundamental data field supported. For detailed list please see GetFnData function reference.

The function returns the number (scalar) representing current value of fundamental data item. There is no history of values (no arrays are returned), so it is useful for scanning, explorations (for current situation), market commentary / interpretation, but not for backtesting. Example exploration formula looks as follows:

```
AddColumn( Close / GetFnData( "EPS" ) , "Current P/E ratio" );
AddColumn( Close / GetFnData( "EPSEstNextYear" ) , "Est. Next Year P/E ratio" );
Filter = Status("lastbarinrange");
```

IMPORTING FUNDAMENTAL DATA FROM OTHER SOURCES

AmiBroker allows also to import fundamentals using its flexible ASCII importer and/or OLE interface as all new fields are exposed as properties of Stock object.

ASCII importer $FORMAT command now supports the following extra fields for fundamental data:

DIV_PAY_DATE
EX_DIV_DATE
LAST_SPLIT_DATE
LAST_SPLIT_RATIO
EPS
EPS_EST_CUR_YEAR
EPS_EST_NEXT_YEAR
EPS_EST_NEXT_QTR
FORWARD_EPS
PEG_RATIO
BOOK_VALUE (requires SHARES_OUT to be specified as well)
BOOK_VALUE_PER_SHARE
EBITDA
PRICE_TO_SALES (requires CLOSE to be specified as well)
PRICE_TO_EARNINGS (requires CLOSE to be specified as well)
PRICE_TO_BV (requires CLOSE to be specified as well)
FORWARD_PE (requires CLOSE to be specified as well)
REVENUE
SHARES_SHORT
DIVIDEND
ONE_YEAR_TARGET
MARKET_CAP (requires CLOSE to be specified as well - it is used to calculate shares outstanding)
SHARES_FLOAT
SHARES_OUT
PROFIT_MARGIN
OPERATING_MARGIN
RETURN_ON_ASSETS
RETURN_ON_EQUITY
QTRLY_REVENUE_GROWTH
GROSS_PROFIT
QTRLY_EARNINGS_GROWTH
INSIDER_HOLD_PERCENT
INSTIT_HOLD_PERCENT
SHARES_SHORT_PREV
FORWARD_DIV
OPERATING_CASH_FLOW
FREE_CASH_FLOW
BETA

Note that if you want to import only fundamental data with ASCII importer (without quotes) you need to use `$NOQUOTES 1` command. See Formats\aqfe.format and Formats\aqfn.format files for example usage - these are files actually used by AmiQuote to implement automatic import of fundamental data downloaded from Yahoo.

The names of extra properties of Stock object are the same as used by GetFnData function and they are listed in detail in OLE objects reference.

Using New Analysis window

Introduction

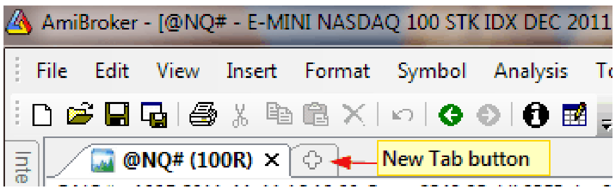
New Analysis window introduced in version 5.50 (first time actually in 5.41.0 BETA) brings the following improvements over old Automatic Analysis:

- **multi-threaded operation = speed -** new Analysis window uses all available CPUs/cores to execute formulas in many threads in parallel providing significant speed ups. For example on 4 core Intel i7 that can run upto 8 threads, it can run upto 8 times faster than old Analysis window. Exact speed up depends on complexity of the formula (the more complex it is, the more speedup is possible), amount of data processed (RAM access may be not as fast as CPU thus limiting possible speed gains).
- **non-blocking operation** - you can now view, scroll and sort results of analysis while they are still generated, also as user interface thread is not used for processing for most part, charts and other GUI-driven program parts are way more responsive than with old automatic analysis
- **multiple instances** - you can run more than one instance of New Analysis at a time, so you can run many scans/backtest/explorations/optimizations in parallel without waiting for one to complete
- **slicker user interface -** New Analysis window can act as tabbed document, can be floated, buttons can be re-arranged for better workflow. There is way more space for the result list, extra information about execution is provided on the new "Info" tab. Also walk-forward results are now displayed within New Analysis window for less clutter.

User interface

You can open New Analysis window in a number of ways:

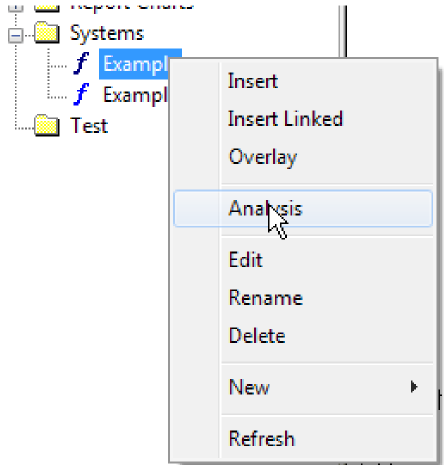
1. click on the New Tab (+) button and selecting **New Analysis**

   
   AmiBroker - [@NQ# - E-MINI NASDAQ 100 STK IDX DEC 2011
   File Edit View Insert Format Symbol Analysis T
   @NQ# (100R)
   New Tab button

   or
2. **File -> New -> New Analysis** menu

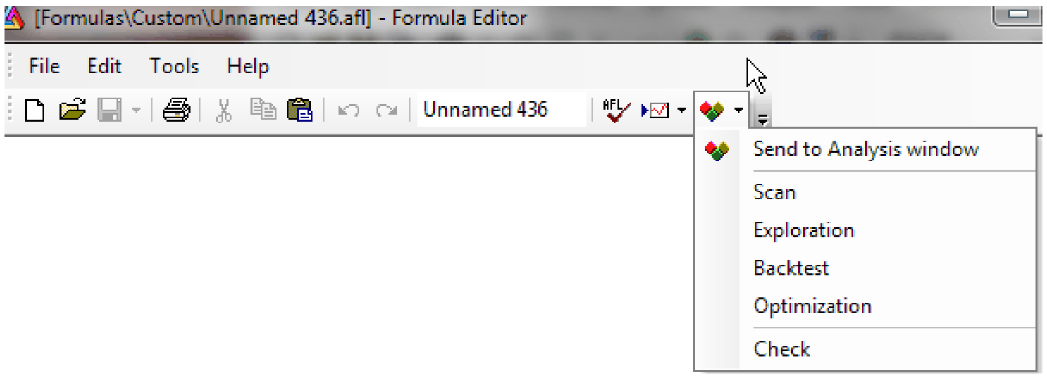
   or
3. **Analysis-> New Analysis** menu

   or
4. right click on the formula in the **Charts** window and selecting **Analysis**

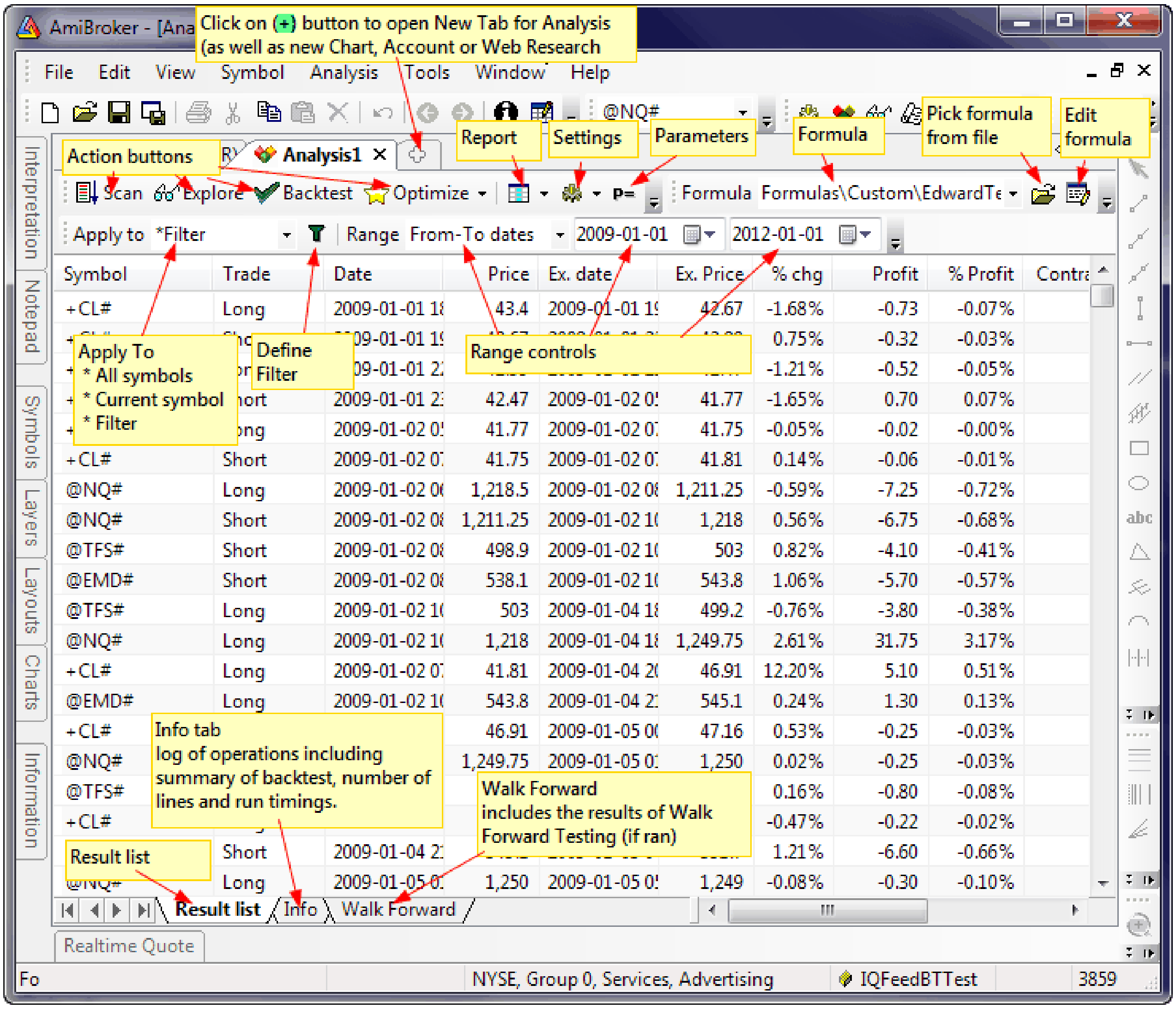


or

5. from the Formula Editor, pressing **Send to Analysis** button



Basically the user interface for New Analysis window is functionally similar to old automatic analysis and looks as follows:

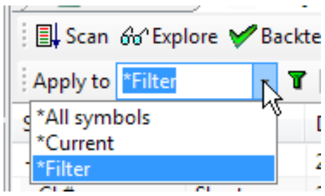

AmiBroker - [Ana
Click on (+) button to open New Tab for Analysis (as well as new Chart, Account or Web Research
File Edit View Symbol Analysis Tools Window Help
@NQ#
Pick formula from file
Edit formula
Report
Settings
Parameters
Formula
Action buttons
Analysis1
Scan Explore Backtest Optimize
Formula Formulas\Custom\EdwardTe
Apply to *Filter
Range From-To dates 2009-01-01 2012-01-01
Apply To
* All symbols
* Current symbol
* Filter
Define Filter
Range controls
Info tab
log of operations including summary of backtest, number of lines and run timings.
Walk Forward
includes the results of Walk Forward Testing (if ran)
Result list
Result list Info Walk Forward
Realtime Quote
Interpretation
Notepad
Symbols
Layers
Layouts
Charts
Information
Fo
NYSE, Group 0, Services, Advertising
IQFeedBTTest
3859

Symbol	Trade	Date	Price	Ex. date	Ex. Price	% chg	Profit	% Profit	Contra
+CL#	Long	2009-01-01 18	43.4	2009-01-01 19	42.67	-1.68%	-0.73	-0.07%	
+						0.75%	-0.32	-0.03%	
+						-1.21%	-0.52	-0.05%	
+		2009-01-01 2	42.47	2009-01-02 0	41.77	-1.65%	0.70	0.07%	
+		2009-01-02 0	41.77	2009-01-02 0	41.75	-0.05%	-0.02	-0.00%	
+CL#	Short	2009-01-02 0	41.75	2009-01-02 0	41.81	0.14%	-0.06	-0.01%	
@NQ#	Long	2009-01-02 0	1,218.5	2009-01-02 0	1,211.25	-0.59%	-7.25	-0.72%	
@NQ#	Short	2009-01-02 0	1,211.25	2009-01-02 1	1,218	0.56%	-6.75	-0.68%	
@TFS#	Short	2009-01-02 0	498.9	2009-01-02 1	503	0.82%	-4.10	-0.41%	
@EMD#	Short	2009-01-02 0	538.1	2009-01-02 1	543.8	1.06%	-5.70	-0.57%	
@TFS#	Long	2009-01-02 1	503	2009-01-04 1	499.2	-0.76%	-3.80	-0.38%	
@NQ#	Long	2009-01-02 1	1,218	2009-01-04 1	1,249.75	2.61%	31.75	3.17%	
+CL#	Long	2009-01-02 0	41.81	2009-01-04 2	46.91	12.20%	5.10	0.51%	
@EMD#	Long	2009-01-02 1	543.8	2009-01-04 2	545.1	0.24%	1.30	0.13%	
+CL#			46.91	2009-01-05 0	47.16	0.53%	-0.25	-0.03%	
@NQ#			1,249.75	2009-01-05 0	1,250	0.02%	-0.25	-0.03%	
@TFS#						0.16%	-0.80	-0.08%	
+CL#						-0.47%	-0.22	-0.02%	
	Short	2009-01-04 2				1.21%	-6.60	-0.66%	
@NQ#	Long	2009-01-05 0	1,250	2009-01-05 0	1,249	-0.08%	-0.30	-0.10%	

Basic operations

Selecting the symbol to apply analysis to.

Click on the drop down arrow in the **Apply to** combo to select operation mode: **All symbols** / **Current** symbol / **Filter**

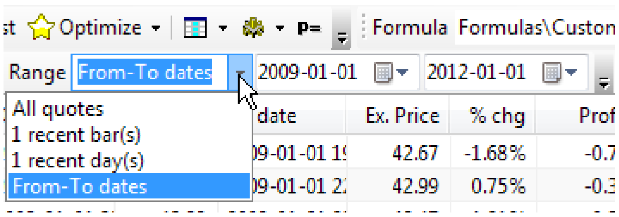


Defining Filter

If **Apply To** is set to **Filter**, Analysis window will be run on the symbols that match filtering criteria that are definable in the Filter Settings window. To open Filter Settings window press **Filter** button

Defining date/time Range

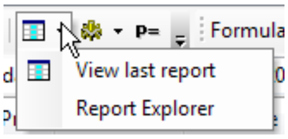
Click on the drop down arrow in the **Range** combo to select range selection mode: **All symbols** / **N recent bar**(s) / **N recent day**(s) / **From-To dates**



The 'N' can represent any number. For example to define range of 15 recent days, select **1 recent day(s)** first and then type in **15** and press ENTER. You will see the text automatically update to **15 recent day(s)**. Remember you don't need to type whole thing, just a number is perfectly enough.

Viewing reports / Running Report Explorer

To view the report from last backtest, click on the **Report** button. To run the Report Explorer use a **Report** button drop down menu as shown below

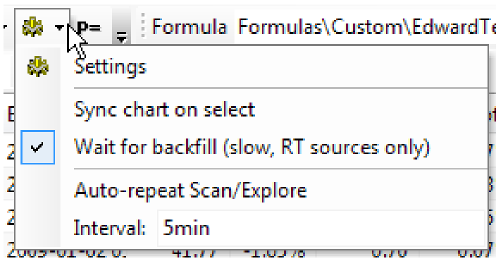


Changing settings / options

To change backtester settings click on the **Settings** button. To turn on/off additional options like:

- Sync chart on select
- Wait for backfill
- Auto repeat Scan/Explore
- Auto repeat interval

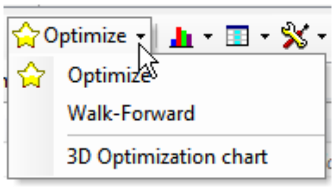
click on the drop down arrow on the **Settings** button to display the menu as shown below


Formula Formulas\Custom\EdwardT
Settings
Sync chart on select
✓ Wait for backfill (slow, RT sources only)
Auto-repeat Scan/Explore
Interval: 5min

Auto-repeat interval can be entered in the **Interval** field. Note that plain numbers (like 5) represent minutes. To get seconds you need to enter **5sec** or **5s** and press **ENTER**

Running Walk forward Test

Click on the arrow on the Optimize button to display the menu as shown below and select **Walk-Forward**

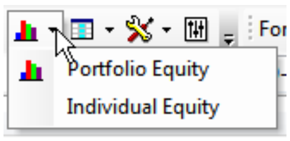


The results of Walk forward test will be displayed in **Walk Forward** tab (see bottom of Analysis window).

Displaying 3D optimization chart

To display 3D optimization chart, first run the **Optimize** that has exactly two optimization parameters and then click on the arrow on the Optimize button to display the menu as shown above and select **3D Optimization Chart**.

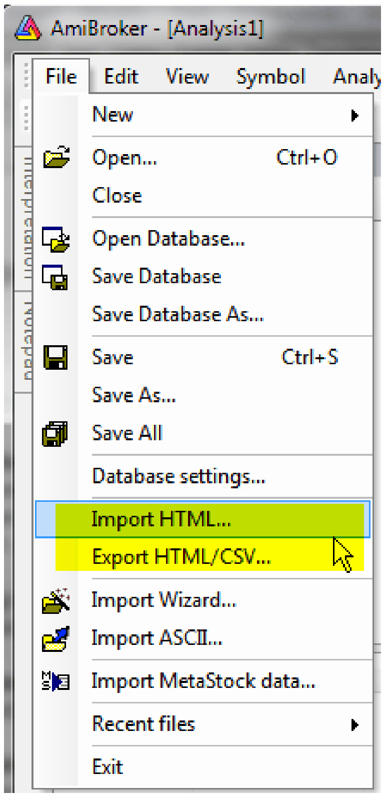
Displaying Equity charts



Equity charts (portfolio and individual) can be added to chart windows using **Portfolio Equity** / **Individual Equity** options as shown above.

Exporting and importing result list

To export data to CSV file or HTML file use **File -> Export HTML/CSV** menu (from the main window). To import previously exported HTML file use **File->Import HTML...** as shown in the picture below. Note that these menu items appear only if you have New Analysis window active.



Running a sequence of actions

In addition to comprehensive batch capabilities AmiBroker 6.40 brings ability to run sequences of actions within Analysis window alone.

This new feature is not intended to replace batches but to augment available choices and allow "quick hacking" so you can run sequences directly in Analysis window without resorting to using full-fledged batches.

There is a new **#pragma sequence**(list of actions) statement that will define sequence of actions to be performed and a toolbar button "Run Sequence" to allow you to run sequence of actions via single click.

This allows you to say create composites and later run exploration that uses those composites all via single click on "Run Sequence" button

The code that uses **#pragma sequence** looks like this:

```
// the statement below defines sequence of actions
#pragma sequence(scan,explore)

if( Status("action") == actionScan )
{
   AddToComposite( .. );
   StaticVarAdd(...);
   _exit(); // quick exit in scan mode
}

Filter = ...
AddColumn( ... ); // exploration code that can use composites / static vars
created in scan step
```

Using Batch window

Introduction

The Batch window introduced in version 6.20 (first time actually in 6.17.0 BETA) allows the user to automate mundane repetitive tasks by providing batch processing.

Now the user can easily **define and automatically run sequences of Analysis operations such as scan, exploration, backtest and optimization and file export.** Previously such automation was available only for programmers by means of OLE automation. Now it is available to everyone via easy-to-use interface.

Generally a batch consists of any number of following basic operations:

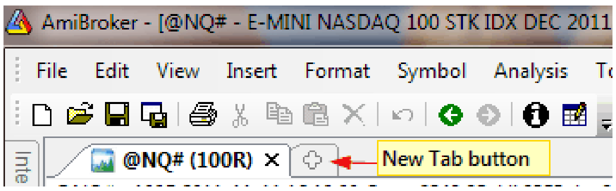
- **loading an Analysis project file to be used -** in order to run a batch you need to first save your analysis project into .APX (Analysis Project) file that can be later loaded by batch processor. An Analysis Project (.APX) is self-contained and it holds everything needed to run your scan, exploration, backtest or optimization including the formula, all options & settings, as well as apply-to and range selections.
- (optional) **Set current symbol** - if your formula relies on specific current symbol you can set it from the batch prior to running Analysis

- **run Scan/Exploration/Backtest/Optimization/Walk forward** - once project is loaded you can start execution of any Analysis operation
- (optional) **Export results** - using Export to CSV/HTML you can export result list. Also backtest reports are automatically generated after backtest so you do **not** need to explicitly export them to see them later in the Report Explorer.
- (optional) **Notify with sound / speech -** Batch processor has ability to play any .WAV file or speak any text supplied so you can be notified when batch has reached certain stage.
- (optional) **Write text** to log file
- (optional) **Load/Save Database**
- (optional) **Execute** external program and wait for completion
- (optional) run vendor-specific **Data Plugin command**
- (optional) **Import** quotation data from ASCII files

User interface

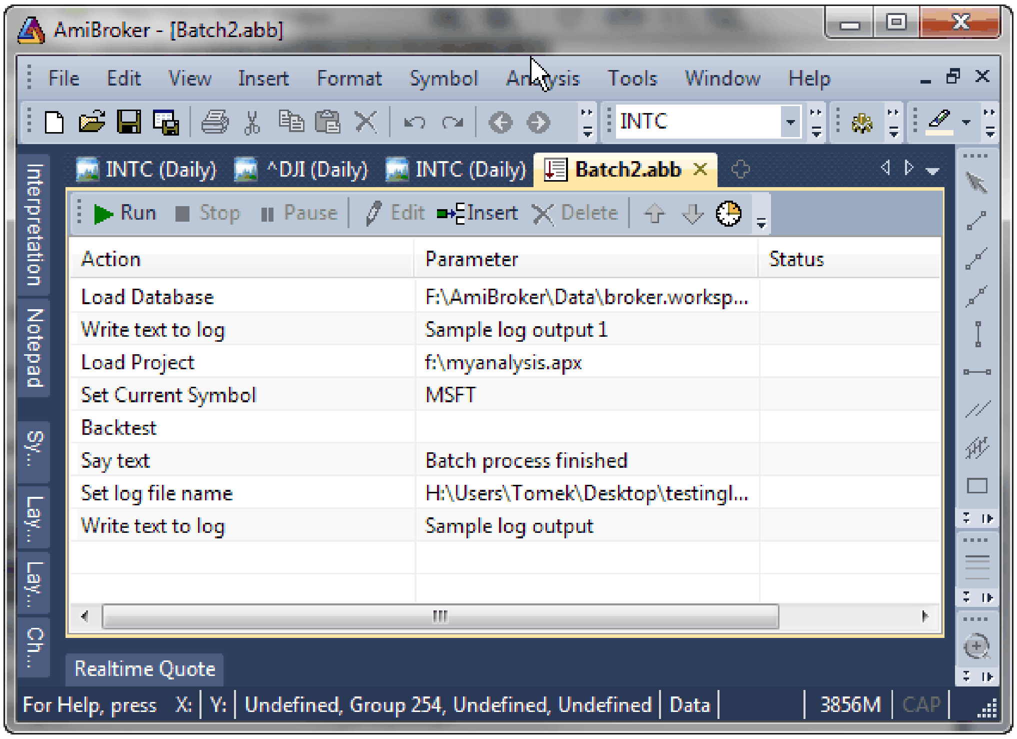
You can open New Analysis window in a number of ways:

1. click on the New Tab (+) button and selecting **New Batch**



   or
2. **File -> New -> New Batch** menu

The user interface for Batch window is very simple:


AmiBroker - [Batch2.abb]
File Edit View Insert Format Symbol Analysis Tools Window Help
INTC
INTC (Daily) ^DJI (Daily) INTC (Daily) Batch2.abb
Run Stop Pause Edit Insert Delete
Interpretation Notepad Sy... Lay... Lay... Ch...
Realtime Quote
For Help, press X: Y: Undefined, Group 254, Undefined, Undefined Data 3856M CAP

Action	Parameter	Status
Load Database	F:\AmiBroker\Data\broker.worksp...	
Write text to log	Sample log output 1	
Load Project	f:\myanalysis.apx	
Set Current Symbol	MSFT	
Backtest		
Say text	Batch process finished	
Set log file name	H:\Users\Tomek\Desktop\testingl...	
Write text to log	Sample log output	

Basic operations

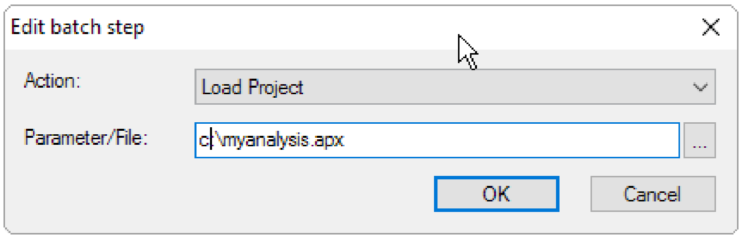
Adding/inserting new batch step

You can add a new batch step by clicking on the **Insert** button. It will add a new step at the end if nothing was selected prior to clicking, or it will insert a new step before previously selected step. You can also add a new step if you just double click on the list past the last listed step.

If you add/insert new step the Edit batch step window will appear (see below).

Editing existing batch step

To edit existing item double click on it, or use **Edit** button. The Edit batch step will appear allowing you to define batch step action and parameters:



Action: Load Project

Parameter/File: c:\myanalysis.apx

In this window you can select an action to be performed using **Action** combo-box and enter the **Parameter** or **File** name to be used for given action.

Actions like **Load Project**, **Export to File** or **Play sound (.wav)** need a file name to be entered / picked via **(...)** button. Actions like **Set Current Symbol** or **Say** need a text parameter (symbol or text to be spoken) to be entered. Other actions (like Scan, Backtest, ...) do not need any extra parameters. When selected action does not require parameter, the second entry field (**Parameter/File**) is hidden.

Once you select desired action along with parameters press **OK** to accept this step. If you changed your mind press **Cancel**. If you cancel **Insert** operation, newly inserted item will be removed from the list.

Deleting existing batch step

To delete existing batch step, select it by clicking on the list and press **Delete** button.

Moving / re-arranging batch steps

You can change the sequence of existing steps by moving individual items up and down. To move item up or down please click on it once and use blue **up** and **down arrow buttons** in the toolbar. You can also re-arrange items by dragging them. To do so, click on item(s) once (to multi-select use CTRL or SHIFT key), and then pressing and holding down the left mouse button drag them to desired location.

Running batch

Once all batch steps are defined you can run the sequence by pressing **Run** (green arrow) button in the toolbar. All steps will be executed in the sequence, you will see subsequent items' status changing to "In progress...", then to "Completed" or "Failed". If any steps ends up with "Failed" status, entire batch execution is stopped.

To temporarily pause execution of batch press **Pause** button. To abort execution of running batch press **Stop** button. (Those buttons are only active when batch is running, otherwise they are disabled/grayed).

Saving batch for future use

A batch sequence can be saved using **File->Save** / **File->Save As...** into .abb file (stands for AmiBroker Batch).

Once batch is saved it can be later loaded back using **File->Open** or using **File->Recent Files** menu.

Miscellaneous commands

Running external programs from batch

A batch command **Execute And Wait** allows to execute external program and wait for its completion (allows among other things launching AmiQuote download and waiting until it is complete)

The following example launches AmiQuote, loads "YourTickerList.tls", performs download and closes AmiQuote:

```
ExecuteAndWait amiquote\quote.exe YourTickerList.tls /download /close
```

Running data plugin commands

Some data vendors may provide special data-specific commands that can be run from the batch. For example PremiumData will provide database maintenance capability.

To run data plugin command use

```
DataPluginCmd commandname
```

Where *commandname* is data-vendor specific command. Please consult data vendor for details. If you specify non-existing command the status of operation will be "Failed" and batch execution will stop.

Importing quotation data from ASCII files

DataImportASCII command allows to automate imports via batch. The parameter is file name to be imported. The importer uses file extension to determine format file used, so if you are importing File.aqh then aqh.format definition file will be used automatically. Import would fail if corresponding definition file does not exist in the "Formats" subdirectory.

Triggering batch run programmatically from AFL level

It is possible to start the batch programmatically from AFL. The ShellExecute function in AmiBroker 6.20 supports "runbatch" command.

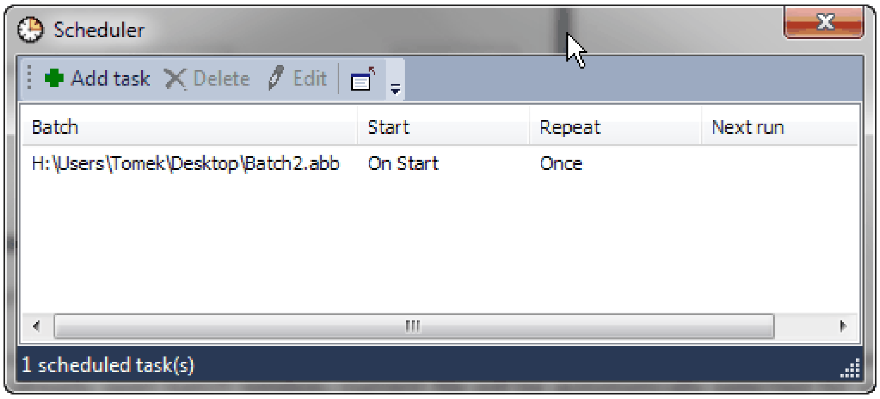
```
if( ParamTrigger("batch", "run me" ) )
{
    ShellExecute("runbatch", "path_to_batch_file.abb", "" );
}
```

Please note that this is asynchronous call, i.e. ShellExecute would start a batch and return immediately even though batch may (and usually will be) still running, so it is "fire and forget" kind of operation. Be careful as repeated executions of AFL would trigger repeated batch runs. After running batch batch and analysis windows that were opened by batch are automatically closed.

Scheduling

Batches can be scheduled for automated run on certain time or at the application start-up. To open batch **Scheduler** either click on Scheduler icon (clock) in the Batch toolbar or use **Tools->Scheduler** item from the main application menu.

This will open the Scheduler window



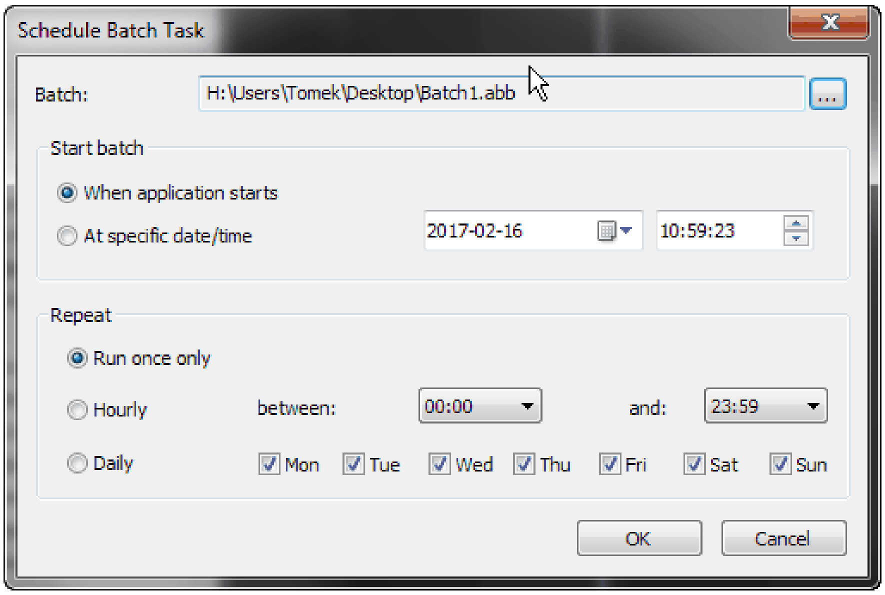
Batch	Start	Repeat	Next run
H:\Users\Tomek\Desktop\Batch2.abb	On Start	Once	

1 scheduled task(s)

In the Scheduler window's toolbar you can see the icons to: Add, Delete, and Edit tasks and to decide whenever to close batch window on completion.

The list displays the path to the batch file, **Start** date/time and **Repeat** interval and the date/time of **Next run**. If batch is scheduled to be run on startup then **Next run** field will be empty.

To add a task, click **Add task** icon. This will open the following dialog:



Batch: H:\Users\Tomek\Desktop\Batch1.abb

Start batch
- [✓ When application starts]
- At specific date/time 2017-02-16 10:59:23

Repeat
- [✓ Run once only]
- Hourly between: 00:00 and: 23:59
- Daily ☑ Mon ☑ Tue ☑ Wed ☑ Thu ☑ Fri ☑ Sat ☑ Sun

To definine scheduled run of batch please do the following:

1. Pick a batch file to be run (with .ABB extension). Such file can be created by simply saving an open Batch via **File->Save** / **File->Save As.** If you opened scheduler from already open batch window, current file will be automatically selected for you.
2. Define when to run the batch: either at application start or at specific date/time
3. Define if runs should be repeated at certain intervals or not. Runs can be done daily or hourly on specific week days and specific time range

Once you are done click OK and you will see scheduled task in the Scheduler window list.

IMPORTANT NOTES:

- Check period is every 10 seconds, so if you schedule something for 12:14:47, it will actually run on 12:14:50.
- If multiple tasks are scheduled on the same time they will be run in order of appearance in the Scheduler list.
- Scheduled tasks are run sequentially and next one would not start if previous did not finish. For this reason avoid batches that require user intervention such as use of "PAUSE" command.
- Scheduled tasks are only started when UI is not in the modal state, i.e. UI is NOT waiting for user input (such as File open dialog, or any other modal input dialog).
- Once modal dialog is closed, missed tasks will be run.
- Scheduled tasks are only run when AmIbroker is running. AmiBroker does not create background / hidden processes. Windows already have too many wasteful processes running all the time. If you need 24/7 operation, just do not close AmiBroker. If you close AmiBroker, it won't run any tasks until you start AmiBroker again, then all missed tasks will be run once and afterwards they will be run on normal schedule (if repeat option is used).

How to get quotes from various markets

IMPORTANT: Data are provided by 3rd party companies. Information provided below is for orientation only. It may be not current, as data vendor offerings change often. For details about current data vendor offerings including prices, data retention & coverage consult their web pages.

THIS PAGE IS CURRENT AS OF March 2, 2023

REAL-TIME DATA (Professional Edition only)

Country/Exchange	**Data source**	**Type**	**Price*** *(subject to change)*	**Download**	**Update**	**Comments**
All US Stock and Futures markets. **FOREX** **Major European markets.**	eSignal	Real time streaming quotes. Tick, 5-, 15-second 1-, 5-, 15-, 60-minute intraday 10-day tick, 60-day minute bar backfill. Historical EOD (10 years)	$61/month (DELAYED, not streaming) $204/month (REALTIME, streaming) More pricing information	Automatic	Automatic	**Dedicated RT plug-in** - details here
US stocks, futures, options, FOREX	DTN IQFeed	**REAL TIME STREAMING** 500 symbols, tick, 5-sec, 15-sec, 1-minute and up, 100+ days tick data 10 years 1-minute data 20+ years EOD (note: unfiltered feed)	$99/month basic fee (real time) More pricing information (special offer for AmiBroker users - setup fee $50 WAIVED)	Automatic	Automatic	**Dedicated RT plug-in** - details here
US, Canada and European	Interactive Brokers	100 symbols streaming RT,	$10 per month in commissions, or	Automatic	Automatic	**Dedicated RT plug-in**

exchanges		1-sec, 1-minute bars and up. **30 day backfill available for IB customers**	free if your monthly commissions are >$30			- details here
Various exchanges / various sources **(detailed list)**	Quote Tracker	Real time streaming quotes. 1-, 5-, 15-, 60-minute intraday Limited (max. 5 days, usually one day) backfills	Various (including free) More pricing information	Automatic	Automatic	**Dedicated RT plug-in** - details here
Warsaw Stock Exchange	Statica	**30/90 days intraday + mixed mode EOD**		Automatic	Automatic	**Dedicated RT plugin** - details here
Various (any data source that has DDE interface)	DDE link	just streaming quotes, **no backfill**	Free	Automatic	Automatic	**Dedicated RT plug-in** - details here

END-OF-DAY, INTRADAY DELAYED DATA

AmiBroker can handle virtually EVERY exchange in the world if only plain ASCII data for that exchange are available. The table below list some of the data sources.

AmiBroker comes preloaded with sample DJIA components database. You can update this sample database (and any other US & Canada market databases) with a new quotes using supplied AmiQuote program.

Later in this tutorial you will find detailed instructions on how to use AmiQuote.

Quote sources for AmiBroker (this list is not complete - keep in mind the fact that almost any source can be used). Use links to find out more (note that some links require internet connection)

Country/Exchange	Data source	Type	Price	Download	Update	Con
	Yahoo Finance	Historical + Current EOD	**Free**	Automatic (**AmiQuote**)	Automatic	Deta desc here
	Premium Data (Norgate Investor Services) (Premium Data - legacy product)	Historical EOD + Daily updates + Sectors / Industries / etc +	Paid	Automatic	Automatic (via MS plugin)	Deta here

		Delisted symbols				
	Norgate Data - new offering	Historical EOD + Daily updates + Sectors / Industries / etc + Delisted symbols	Paid	Automatic	Automatic (via dedicated plugin)	Deta
	TC 2000/TCNet v7 (US stocks) TC2000 Mutual Funds	Historical + Current EOD + Sectors/Industries	Paid	Automatic	Automatic	**Dire** - de here
	FastTrack (mutual funds)	Historical + Current EOD + Families	Paid	Automatic	Automatic	**Dire** - de here
	FOREX	Historical EOD + Intraday	**Free**	Automatic (**AmiQuote**)	Automatic	Deta desc here
	CSI http://www.csidata.com	Historical EOD	Paid: Details here	Automatic	Automatic	Upd usin Unfa Adv
Australia (Australian Stock Exchange)	Premium Data (Norgate Investor Services) (Premium Data - legacy product)	Historical EOD + Daily updates + Sectors / Industries / etc + Delisted symbols	Paid	Automatic	Automatic (via MS plugin)	Deta here
	Norgate Data - new offering	Historical EOD + Daily updates + Sectors / Industries / etc + Delisted symbols	Paid	Automatic	Automatic (via dedicated plugin)	Deta
	BodhiFreeway	Historical	Paid	Automatic (Bodhi downloader)	Automatic (via METASTOCK plugin)	How setu to M data
	Yahoo Finance Australia	Current EOD	Free	Automatic (**AmiQuote**)	Automatic	Deta desc here
50+ International Exchanges	Yahoo Finance	Historical + Current EOD	Free	Automatic (**AmiQuote**)	Automatic	Deta desc here
Poland (Warsaw Stock Exchange)	Bossa.pl	Historical + Current EOD	Free	Automatic (script-based)	Automatic (script-based)	Deta new
South Africa (Johannesburg Stock	Sharenet	Historical + Current EOD	Paid	Automatic (**Sharenet**	Automatic (script- based)	Sha Ami

Exchange)				**downloader**)		corr
	Investor Data	Historical	Paid	Manual	Manual	
Holand (Amsterdam - Euronext)	PF-online	Historical + Current EOD	Free	Manual	ASCII Import	

How to set up AmiBroker with eSignal feed (RT version only)

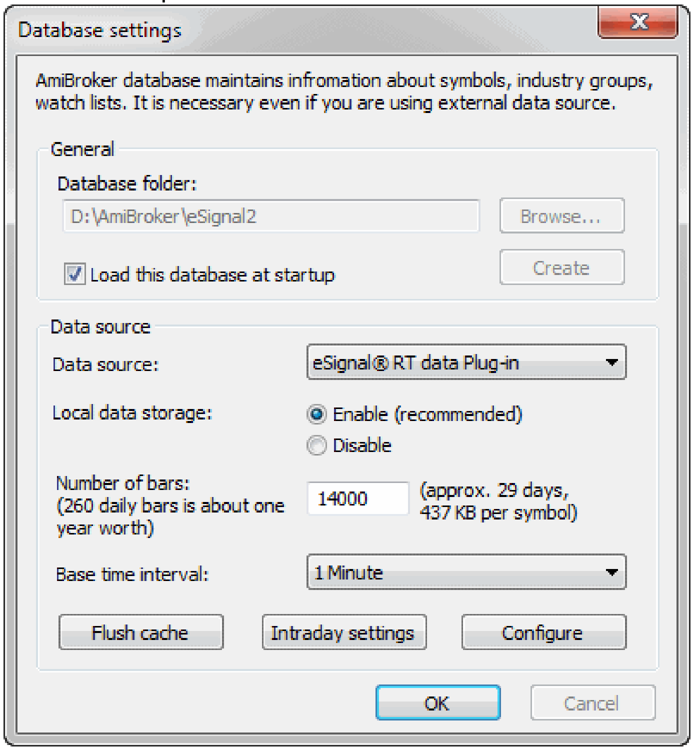
Requirements

IMPORTANT: **You have to have eSignal application installed on your machine and a valid eSignal subscription**.

One-time setup

To use AmiBroker with eSignal feed you will need to perform a one-time setup described below:

- Run AmiBroker
- Choose **File->New database**
- Type a new folder name (for example: C:\Program Files\AmiBroker\eSignal) and click **Create** as shown in the picture below:



Database settings

AmiBroker database maintains infromation about symbols, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: D:\AmiBroker\eSignal2

☑ Load this database at startup

Data source

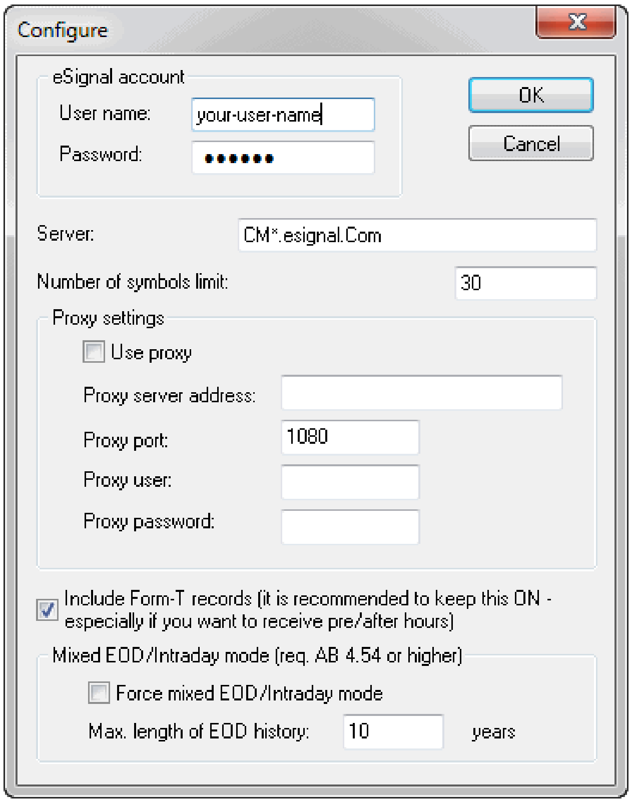
Data source: eSignal® RT data Plug-in

Local data storage: [✓ Enable (recommended)] Disable

Number of bars: (260 daily bars is about one year worth) 14000 (approx. 29 days, 437 KB per symbol)

Base time interval: 1 Minute

- Choose **eSignal RT data Plug-in** from Data source combo and "**Enable**" from **Local data storage**
- Enter appropriate **number of bars to load:**
  90000 for 1-minute database combined with long history daily database
- Click on **Configure** button to show plugin configuration dialog as shown below



Configure

eSignal account

User name:	your-user-name
Password:	••••••
Server:	CM*.esignal.Com
Number of symbols limit:	30

Proxy settings

- [ ] Use proxy

Proxy server address:	
Proxy port:	1080
Proxy user:	
Proxy password:	

- [✓] Include Form-T records (it is recommended to keep this ON - especially if you want to receive pre/after hours)

Mixed EOD/Intraday mode (req. AB 4.54 or higher)

- [ ] Force mixed EOD/Intraday mode

Max. length of EOD history: 10 years

Enter here your eSignal user and password (if you have eSignal properly installed AmiBroker will pre-set these fields to user/password entered in eSignal software). You may also adjust **Number of symbols.** This should not exceed your account limit and you may consider lowering this value if you want to use AmiBroker in parallel with another Data manager client application. (If you exceed the limit of your subscription AmiBroker will re-adjust this number down)

Click OK

- Now choose Base periodicity. Note that **recommended periodicity is 1 minute**, but you can select all base periods starting from tick upto hourly.
  Note that selecting tick, 1-second, 5-seconds or 15-seconds periodicities will cause transmission of huge amounts of data from eSignal servers (for actively traded security it can be several megabytes for just one symbol and very few days of history). If you have a modem connection this setting is highly discouraged. Also if you should consider using 1-second bars instead of pure ticks since this mode is faster.

  Also note that to get long end-of-day histories together with intraday data you should go to File->Database Settings->Intraday Settings and turn ON "**Allow mixed EOD/intraday data**" option.

- Click OK.

From now on your AmiBroker reads quotes directly from the eSignal.

To learn how to use AmiBroker in Real Time mode read this tutorial article.

How to set up AmiBroker with myTrack feed (RT version only)

Note: the most recent version of this document can be found at: http://www.amibroker.com/mytrack.html. Please check this page for updates.

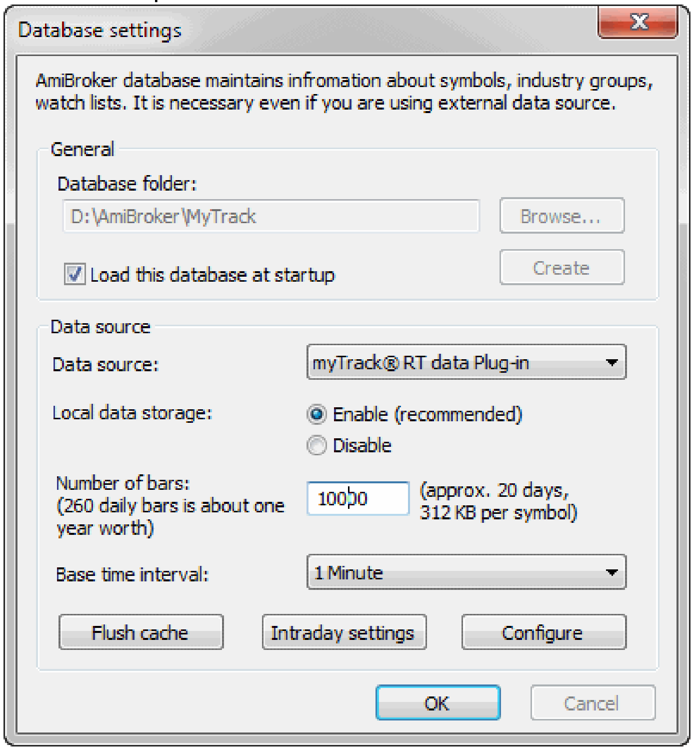
Requirements

IMPORTANT: You have to have myTrack subscription with SDK feature enabled.
To have the SDK working, run the myTrack program, click on CHAT, then on Entitlements and then on Features, check the box SDK.

One-time setup

To use AmiBroker with myTrack feed you will need to perform a one-time setup described below:

- Run AmiBroker
- Choose **File->New database**
- Type a new folder name (for example: C:\Program Files\AmiBroker\myTrack) and click **Create** as shown in the picture below:



Database settings

AmiBroker database maintains infromation about symbols, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: D:\AmiBroker\MyTrack

[✓] Load this database at startup

Data source

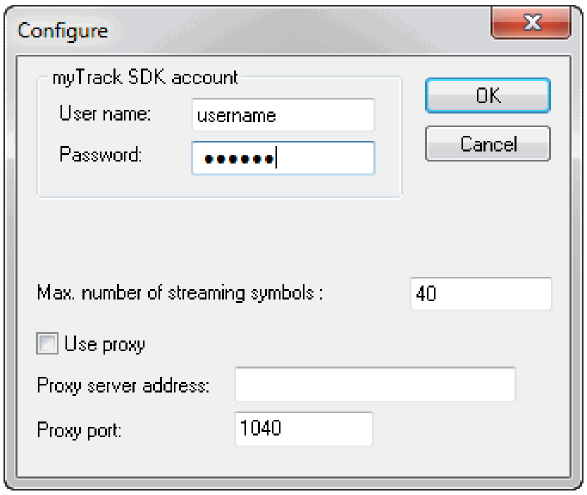
Data source: myTrack® RT data Plug-in

Local data storage: [● Enable (recommended)] ○ Disable

Number of bars: (260 daily bars is about one year worth) 100000 (approx. 20 days, 312 KB per symbol)

Base time interval: 1 Minute

- Choose **myTrack RT data Plug-in** from Data source combo and "**Enable**" from **Local data storage**
- Click on **Configure** button to show plugin configuration dialog as shown below



User name: username
Password: ••••••
Max. number of streaming symbols : 40
Use proxy
Proxy server address:
Proxy port: 1040

Enter here your myTrack user and password . You may also adjust **Number of symbols.** This should not exceed your account limit.

Click OK

- Now choose Base time interval. Note that supported bar intervals are **1 minute** and **daily (end-of-day)**.

  If you want to have long daily histories AND intraday charts you should consider running TWO instances of AmiBroker. One for EOD charts and second for intraday charting. Both instances may use myTrack as a data source.

- Click OK.

From now on your AmiBroker reads quotes directly from the myTrack.

To learn how to use AmiBroker in Real Time mode read this tutorial article.

How to use AmiBroker with external data source (Quote Tracker)

IMPORTANT: You need QuoteTracker 2.4.9C OR ABOVE (3.1.0 recommended). Can operate on standard edition but AmiBroker RT is recommended.

VERY IMPORTANT: QuoteTracker has to be CONFIGURED so its internal server is running. Click here for the explanation.

CAVEAT: QuoteTracker should be considered as poor-man's real-time substitute. Its performance can not be compared to true real-time feed as eSignal or myTrack that offer very reliable, long back-fills and true tick-by-tick updates.

QuoteTracker plugin currently works in TWO modes:

daily mode - plugin adds and updates the last (todays) bar with the most recent quotes in nearly real time- it means that you have to use it in conjunction with already existing end-of-day database.

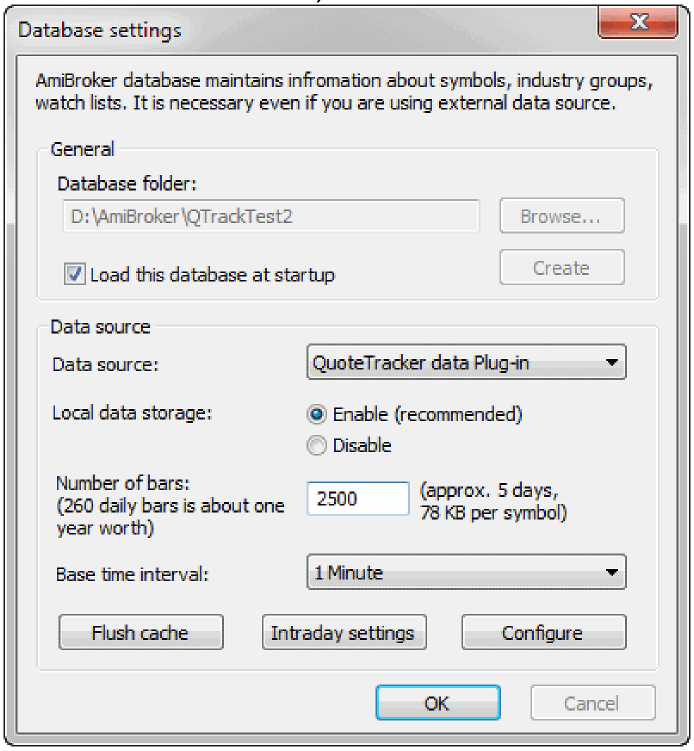
intraday mode - plugin provides one day intraday historical data - more days can be accumulated if AmiBroker with QT is launched everyday so AmiBroker can save histories to its local database.

One-time setup

Make sure that your QuoteTracker has enabled QT HTTP server: **Options->Edit Preferences : Misc tab: HTTP Server Settings If you are using unregistered version of QuoteTracker make sure you click on ads often enough.**

To use an external data source with AmiBroker you will need to perform a one-time setup described below:

Run AmiBroker Choose **File->New database** Type a new folder name (for example: C:\Program Files\AmiBroker\NewData) and click **Create** as shown in the picture below:


Database settings

AmiBroker database maintains infromation about symbols, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: D:\AmiBroker\QTrackTest2

[✓] Load this database at startup

Data source

Data source: QuoteTracker data Plug-in

Local data storage: [● Enable (recommended)] Disable

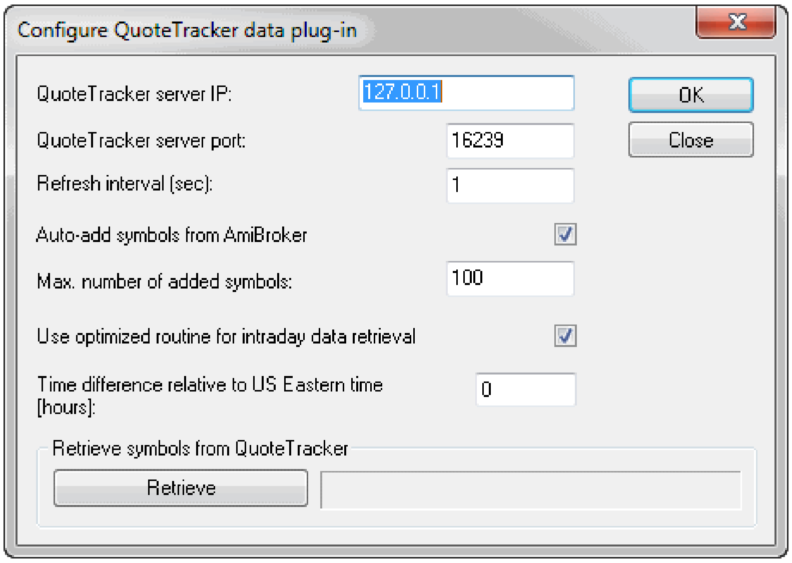
Number of bars: (260 daily bars is about one year worth) 2500 (approx. 5 days, 78 KB per symbol)

Base time interval: 1 Minute

Choose appropriate entry from Data source combo:

- **Quote Tracker users** select "**Quote Tracker plug-in**" as a **Data Source** and "**Enable**" from **Local data storage**

Click on **Configure** button to show plugin configuration dialog as shown below


Configure QuoteTracker data plug-in

QuoteTracker server IP: 127.0.0.1

QuoteTracker server port: 16239

Refresh interval (sec): 1

Auto-add symbols from AmiBroker ✓

Max. number of added symbols: 100

Use optimized routine for intraday data retrieval ✓

Time difference relative to US Eastern time [hours]: 0

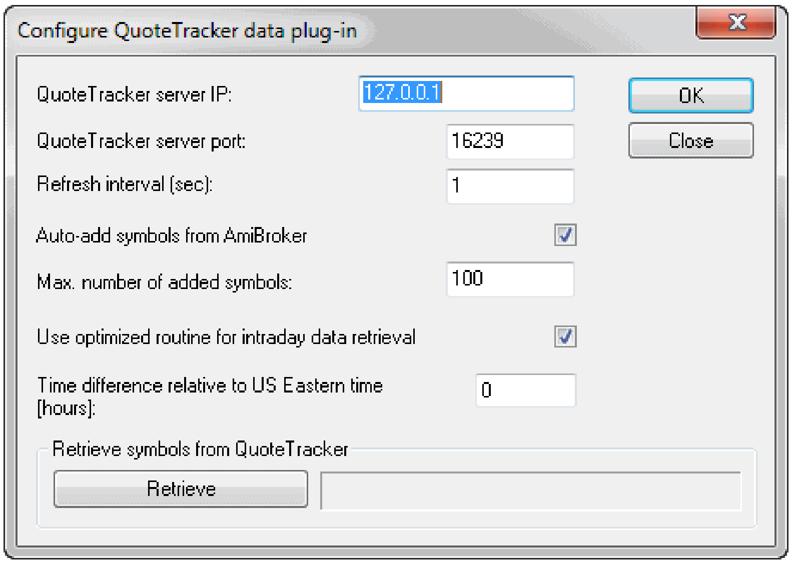
Retrieve symbols from QuoteTracker

You may also click on **Retrieve** button to pre-fill AmiBroker database with symbols already present in QuoteTracker. From now on your AmiBroker reads quotes from Quote Tracker in nearly real time.

To learn how to use AmiBroker in Real Time mode read this tutorial article.

Description of QuoteTracker plugin configuration options

QT plugin configuration dialog looks as follows:



Configure QuoteTracker data plug-in

QuoteTracker server IP:	127.0.0.1
QuoteTracker server port:	16239
Refresh interval (sec):	1
Auto-add symbols from AmiBroker	✓
Max. number of added symbols:	100
Use optimized routine for intraday data retrieval	✓
Time difference relative to US Eastern time [hours]:	0

Retrieve symbols from QuoteTracker

Here is a description of the settings:

QuoteTracker server port: defines the port on which QT HTTP internal server is visible. 16239 is the default value used by QuoteTracer and you should not change this in most cases. If in doubt please check QuoteTracker HTTP server settings: **Options->Edit Preferences : Misc tab: HTTP Server Settings menu of QT.**

Refresh inteval - defines how often AmiBroker will ask QT for quotes. 5 second is default. You may consider changing it to 10 or 15 seconds in case you have lots of symbols and slow machine

Auto-add symbols from AmiBroker - if this option is turned ON (by default it is) if you switch in AmiBroker to the symbol that is not present in any of QT portfolios - it will be automatically added to default QT portfolio. It also applies to any other kind of access (for example if you try to import symbols to AmiBroker and they do not exist in QT - they will be added if this option is turned on). Switching it OFF disables auto-add feature.

Max. number of added symbols - defines the maximum number of symbols that get added using auto-add feature descibed above. This protects QuoteTracker from becoming overloaded (AmiBroker can handle tens of thousands symbols with ease but QuoteTracker can NOT)

Use optimized routine for intraday data retrieval - turning this on (default, recommended) significantly speeds-up data retrieval in intraday modes. If this option is enabled and AmiBroker already has partial intraday data for today AmiBroker asks QT just for a few last time and sales records that occurred since last update upto current time, if this option is disabled AmiBroker always asks QT for time&sales records from entire day.
Time difference relative to US Eastern time - the time difference (in hours) between your local time and US Eastern time (EST). This field is needed because QuoteTracker's server reports all times in EST time zone. This means that if you live in Australia QuoteTracker will report ASX quotes with EST time zone and they will be 15 hours off from your local time. While AmiBroker has the setting for shifting intraday charts and this is not a problem when running Intraday mode, it becomes a problem when using daily (EOD) mode because quotes reported by QuoteTracker are one day off then. This setting solves this as AmiBroker adds the number of hours entered here to the time reported by QuoteTracker to get the valid date of quote in daily mode. This field is filled in with the difference calculated using your Windows Time settings.

Retrieve symbols from QuoteTracker - pressing "Retrieve" button adds all symbols present in QuoteTracker to AmiBroker symbol list.

How to set up AmiBroker with IQFeed feed (RT version only)

Note: the most recent version of this document can be found at: http://www.amibroker.com/iqfeed.html . Please check this page for updates.

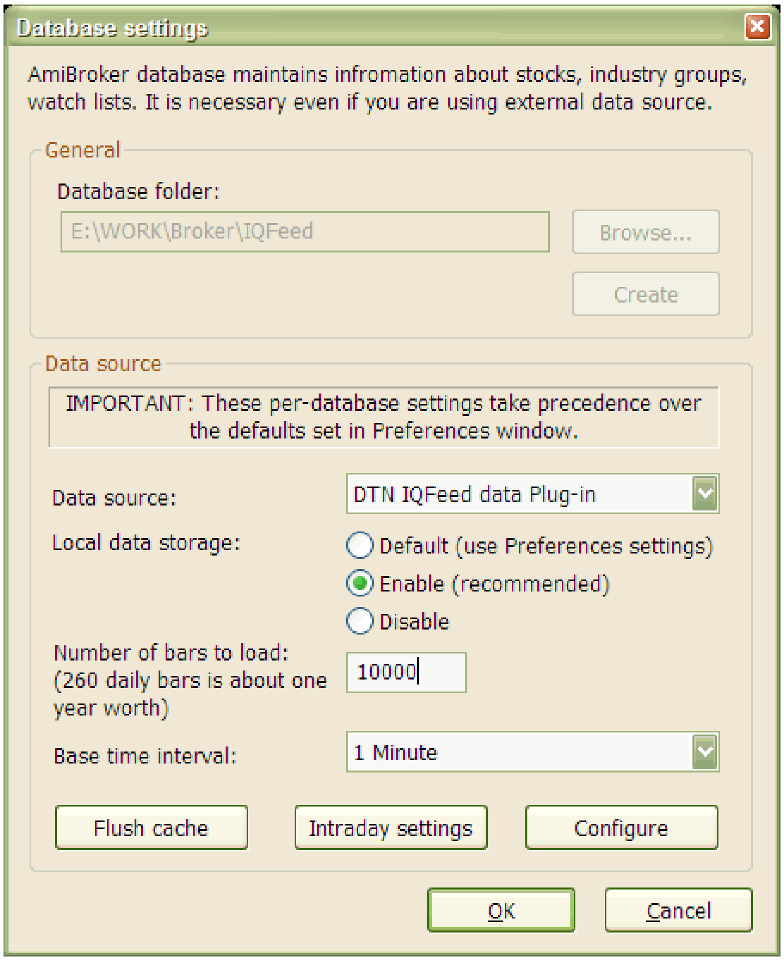
Requirements

If you don't have IQFeed CONNECTION MANAGER already installed you have to install it first. You can download IQFeed client setup from here (version 4.2.0.7).

http://www.amibroker.com/video/IQFeed.html

To use AmiBroker with IQFeed you will need to perform a one-time setup described below:

- Run AmiBroker
- Choose **File->New database**
- Type a new folder name (for example: C:\Program Files\AmiBroker\IQFeed) and click **Create** as shown in the picture below:



AmiBroker database maintains infromation about stocks, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: E:\WORK\Broker\IQFeed

Data source

IMPORTANT: These per-database settings take precedence over the defaults set in Preferences window.

Data source: DTN IQFeed data Plug-in

Local data storage: Default (use Preferences settings) [✓ Enable (recommended)] Disable

Number of bars to load: (260 daily bars is about one year worth) 10000

Base time interval: 1 Minute

- Choose **DTN IQFeed data Plug-in** from Data source combo and "**Enable**" from **Local data storage**
- Now choose Base time interval. Select 1-minute
- Enter appropriate **number of bars to load:**
  100000 for 1-minute database to get max history (8 months) available from IQFeed
- **Click on "Intraday Settings". Check "Allow mixed EOD/Intraday data" box. Click OK**
- Click OK.

From now on your AmiBroker reads quotes directly from the IQFeed.

To learn how to use AmiBroker in Real Time mode read this tutorial article.

How to use AmiBroker with Interactive Brokers TWS

Note: the most recent version of this document can be found at: http://www.amibroker.com/ib.html . Please check this page for updates.

IB PLUGIN FEATURES SUMMARY:

- supports upto 100 streaming symbols in real time (equal to IB TWS limit)
- automatic connection (no need to manually "accept incoming connection" in TWS)
- supports upto **30 DAYS intraday data BACKFILL in 1-minute bar interval**
- upto 2000 bars backfill using 1-sec/5-sec/15-second bar intervals

INSTRUCTIONS:

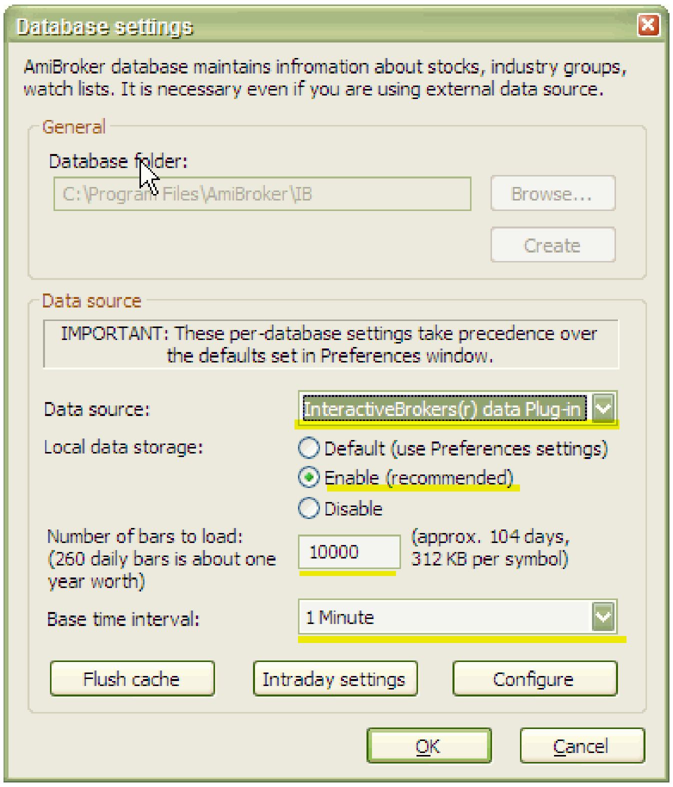
NOTE: Interactive Brokers TWS is CPU-hungry application, therefore for best results we recommend using machine with 1GHz processor or faster.

*NOTE 2: There is a **VIDEO tutorial** showing how to set it up at http://www.amibroker.com/video/ib.html*

To use Interactive Brokers data plugin with AmiBroker you need to:

1. run web-based TWS or download standalone TWS
2. In TWS, select **Configure -> API -> Enable Active X and Socket clients**
   Also **enter 127.0.0.1 in TWS, Configure->API->Trusted IP addresses** menu to prevent "Allow incoming connection?" dialog.
3. Run AmiBroker and create new database with Interactive Brokers plugin as a data source, following these steps:

- Run AmiBroker
- Choose **File->New database**
- Type a new folder name (for example: C:\Program Files\AmiBroker\IB) and click **Create** as shown in the picture below:


Database settings

AmiBroker database maintains infromation about stocks, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: C:\Program Files\AmiBroker\IB

Data source

IMPORTANT: These per-database settings take precedence over the defaults set in Preferences window.

Data source: InteractiveBrokers(r) data Plug-in

Local data storage: Default (use Preferences settings) [✓ Enable (recommended)] Disable

Number of bars to load: (260 daily bars is about one year worth) 10000 (approx. 104 days, 312 KB per symbol)

Base time interval: 1 Minute

- Choose **InteractiveBrokers(r) data Plug-in** from Data source combo and "**Enable**" from **Local data storage**
- Enter 30000 or more into "**Number of bars to load**" field
- Now choose Base time interval. **Supported intervals are: EOD, hourly, 15-minute, 5-minute, 1-minute. Professional Edition of AmiBroker allows also to select Tick, 5-second, 15-second intervals.**

  Note that backfill is in bar interval of 1-minute or less (TWS limitation).

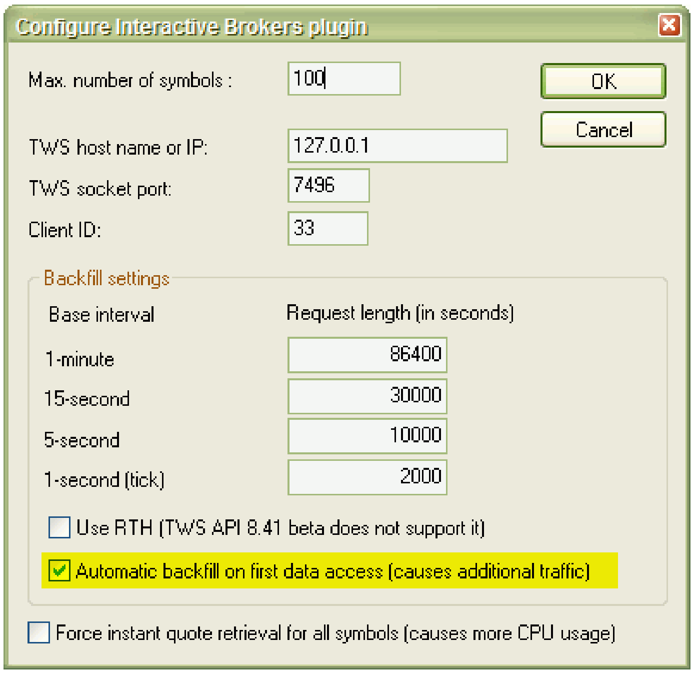
  If you want to have long daily histories AND intraday charts you should consider running TWO instances of AmiBroker. One for EOD charts and second for intraday charting. Both instances may use IB as a data source.
- Click OK.

From now on your AmiBroker reads quotes directly from the Interactive Brokers.

HOW TO USE BACKFILL FEATURE

Backfill feature in plugin 1.3.7 allows to download 24 intraday historical data to fill-in the gaps that may have occurred when AmiBroker / TWS is not running.

IB Backfill feature is configurable from **File->Database Settings**, **Configure :**



Max. number of symbols : 100
TWS host name or IP: 127.0.0.1
TWS socket port: 7496
Client ID: 33

Backfill settings

Base interval	Request length (in seconds)
1-minute	86400
15-second	30000
5-second	10000
1-second (tick)	2000

☐ Use RTH (TWS API 8.41 beta does not support it)
☑ Automatic backfill on first data access (causes additional traffic)
☐ Force instant quote retrieval for all symbols (causes more CPU usage)

Two main backfill-related settings are:

1. request length
2. automatic backfill

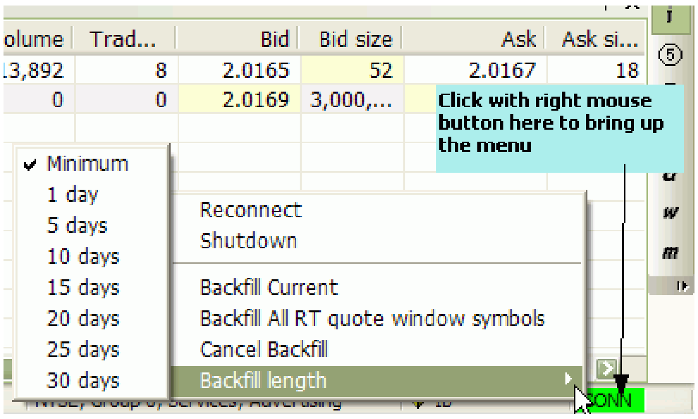
When request length is considered, as explained in TWS API Release Notes at: http://www.interactivebrokers.com/en/software/apiReleaseNotes/apiBetanotes.php currently IB backfill feature is limited to some fixed duration / bar interval ranges. For example you can get maximum 2000 1-second ticks, maximum 10000 seconds in 5-second interval (2000 bars), maximum 30000 seconds in 15-second interval (also 2000 bars) and **maximum of 5 DAYS of 1-minute bars**.

By default AmiBroker uses maximum allowable amounts.

As for "automatic backfill on first data access" - when it is checked AmiBroker attempts to backfill symbol when you display a chart for given symbol (or perform backtest or scan). Please note that TWS API currently allows only one backfill at a time so when there is a backfill already running in the background, automatic backfill request for next symbol will be ignored, until previous backfill is complete.

It is convenient to have this option turned on, however it can cause additional load on your internet connection because of data needed to be downloaded during backfill process.

If you switch "automatic backfill on first data access" option off, you will still be able to backfill data for current symbol or all symbols in real-time quote window list usign appropriate menu options from plugin status menu.

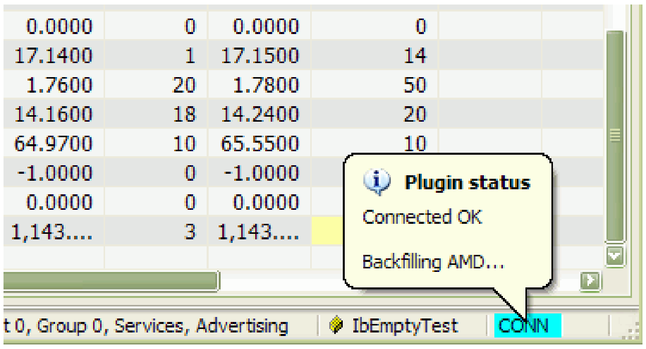


olume	Trad...	Bid	Bid size	Ask	Ask si...
13,892	8	2.0165	52	2.0167	18
0	0	2.0169	3,000,...		

Backfill Current option allows to force backfill of currently selected symbol, while **Backfill All RT quote window symbols** allow to force backfill of all symbols listed in Real-Time Quote window. Backfill of multiple symbols is performed sequentially (one at a time) due to limitations of TWS.

Backfill length submenu allows to select desired backfill length.

During backfilling a tooltip pops up informing the user about symbol being currently backfilled and plugin status color changes to light blue (turquoise) as shown below:



0.0000	0	0.0000	0
17.1400	1	17.1500	14
1.7600	20	1.7800	50
14.1600	18	14.2400	20
64.9700	10	65.5500	10
-1.0000	0	-1.0000	
0.0000	0	0.0000	
1,143....	3	1,143....	

Plugin status
Connected OK
Backfilling AMD...

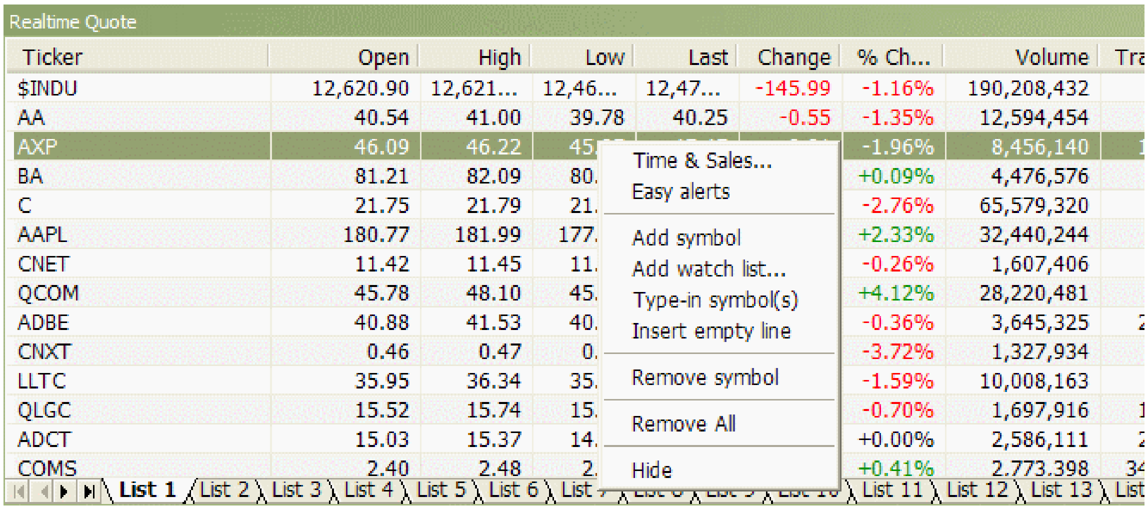
t 0, Group 0, Services, Advertising | IbEmptyTest | CONN

BACKFILLING ALL SYMBOLS AT ONCE

To backfill all symbols at once do the following:

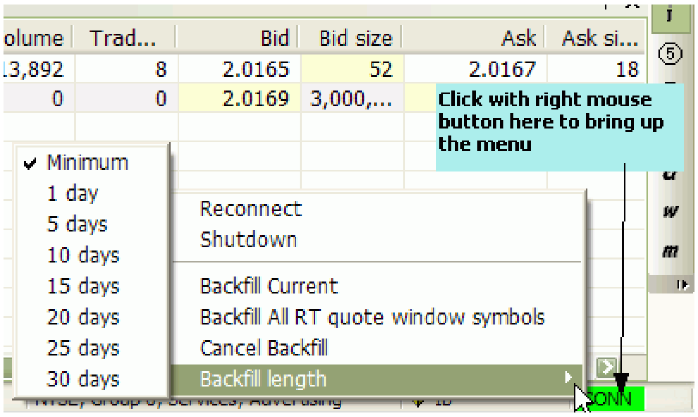
1. Open **Realtime Quote** window (by selecting **Window->Realtime Quote** menu)

2. Right click on the **Realtime Quote** window and choose **Add symbol / Add watch list** / **Type-in symbol** to add any symbols you want to backfill.



Ticker	Open	High	Low	Last	Change	% Ch...	Volume	Tra
$INDU	12,620.90	12,621...	12,46...	12,47...	-145.99	-1.16%	190,208,432	
AA	40.54	41.00	39.78	40.25	-0.55	-1.35%	12,594,454	
AXP	46.09	46.22	45.			-1.96%	8,456,140	1
BA	81.21	82.09	80.			+0.09%	4,476,576	
C	21.75	21.79	21.			-2.76%	65,579,320	
AAPL	180.77	181.99	177.			+2.33%	32,440,244	
CNET	11.42	11.45	11.			-0.26%	1,607,406	
QCOM	45.78	48.10	45.			+4.12%	28,220,481	
ADBE	40.88	41.53	40.			-0.36%	3,645,325	2
CNXT	0.46	0.47	0.			-3.72%	1,327,934	
LLTC	35.95	36.34	35.			-1.59%	10,008,163	
QLGC	15.52	15.74	15.			-0.70%	1,697,916	1
ADCT	15.03	15.37	14.			+0.00%	2,586,111	2
COMS	2.40	2.48	2.			+0.41%	2,773,398	34

3. Right-click on the **plugin Status indicator** and select desired **Backfill length**



olume	Trad...	Bid	Bid size	Ask	Ask si...
13,892	8	2.0165	52	2.0167	18
0	0	2.0169	3,000,...		

4. Choose **Backfill All RT quote window symbols** option from the same menu.

Since Interactive Brokers severely limits number of backfills that customer may request within given time it is advised to use backfill length as short as possible, like 1-day or 5-day and avoid long ranges like 30-days.

SYMBOLOGY

Symbol format now uses the symbol mode of TWS, not the underlying mode. The symbol mode in TWS can be seen in the '**View->Symbol Mode**' menu option in TWS.

The format is: *SYMBOL-EXCHANGE-TYPE*

where

SYMBOL is the same as the symbol column as displayed in TWS while under symbol mode

EXCHANGE (optional) is the exchange d in TWS while under symbol mode

TYPE (optional) is one the following:

STK - stocks, FUT - futures, FOP - options on futures, OPT - options, IND - indexes, CASH -cash (ideal FX)
Note that for stocks only the *EXCHANGE* and *TYPE* fields are optional. The exchange will be set to BEST (SMART) and the TYPE will be set to STK.

Please take special care when typing symbols as some of them (futures) have MULTIPLE SPACES in the symbol name. You have to type EXACTLY THE SAME number of spaces as provided in the examples below (see the dashes below symbol name that make it easier to see the number of characters)

Examples:

IB SYMBOL	Type	Description
`CSCO`	Stock	Cisco Corporation, Nasdaq
`GE`	Stock	General Electric, NYSE
`VOD-LSE`	Stock	VODAFONE GROUP, London Stock Exchange
`ESM4-GLOBEX-FUT`	Future	Emini ES Jun04 futures, Globex
`QQQFJ-CBOE-OPT`	Option	Jun 04, 36.0 CALL option QQQFJ
`INDU-NYSE-IND`	Index	Dow Jones Industrials Index

`YM   JUN 04-ECBOT-FUT`
`  ---   -`
	Future	YM Jun 04 future, ECBOT
		(note 3 spaces between symbol and month and 1 space between month and year)

`QMN5-NYMEX-FUT`	Future	QM (Crude) June 2005 future contract, NYMEX
`XAUUSD-SMART-CMDTY`	Commodity	London Gold Spot
`IBUS500-SMART-CFD-USD`	CFD (contract for difference)	IB US500 contract for difference
`EUR.USD-IDEAL-CASH` `EUR.USD-IDEALPRO-CASH`	Cash Forex	EURUSD currency pair, IDEAL EURUSD currency pair, IDEALPRO

Again:

ECBOT futures symbols have length of 21 characters with 3 spaces between contract symbol and month name and one space between month and 2 digit year

Contract		3 spaces			Month			spa ce	Year		-	E	C	B	O	T	-	F	U	T
Z	B				J	U	N		0	4	-	E	C	B	O	T	-	F	U	T
Z	F				J	U	N		0	4	-	E	C	B	O	T	-	F	U	T
Z	N				J	U	N		0	4	-	E	C	B	O	T	-	F	U	T
Y	M				J	U	N		0	4	-	E	C	B	O	T	-	F	U	T

NOTES ON IB API LIMITATIONS:

1. Backfill is available for REAL IB accounts only (not on demo)

2. **Open** price is NOT provided by IB. For that reason Open field is empty in real time quote window

3. The data from IB does not include a timestamp on the trades. The current system time is used to timestamp each tick.

4. **IB TWS streaming data are NOT tick-by-tick, but rather 0.2-0.3 second snapshots, read this for details: http://www.interactivebrokers.com/cgi-bin/discus/board-auth.pl?file=/2/37364.html**

How to use AmiBroker with external DDE data source

Note: the most recent version of this document can be found at: http://www.amibroker.com/dde.html . Please check this page for updates.

WHAT IS DDE

DDE (Dynamic Data Exchange) is a Windows protocol used to allow applications to exchange data. For example, when you change a form in your database program or a data item in a spreadsheet program, they can be set up to also change these forms or items anywhere they occur in other programs you may use. DDE uses a client/server model in which the application requesting data is considered the client and the application providing data is considered the server.
Thousands of applications use DDE, including Microsoft's Excel, Word, Lotus 1-2-3, and Visual Basic.

For more information about DDE as communication mechanism in Windows please follow this link:
http://msdn.microsoft.com/library/en-us/winui/WinUI/WindowsUserInterface/DataExchange/DynamicDataExchange/Ab

DDE FOR TRADERS

What DDE offers for traders? Basically real time streaming quotes. **There is NO BACKFILL via DDE**. Many real-time data providers and brokerages provide ability to get real-time data by means of DDE. You should ask your brokerage/real-time data vendor if they offer DDE link. The DDE plugin now available for AmiBroker allows to link to (almost) any DDE source (server) supplying real-time quotes. This makes it attractive option for all data sources that do not have dedicated plugin.

WHEN NOT TO USE DDE PLUGIN

If you are using eSignal, IQFeed, Quote.com, and any other source that has dedicated plugin - you should use this dedicated plugin instead of DDE. This is so because dedicated plugins are ALWAYS better option (provide more features plus they are faster) than generic DDE.

DDE PLUGIN FEATURES SUMMARY

- user-definable DDE server/topic/item for each field (open, high, low, close, volume, trade size, total volume, bid, bid size, ask, ask size, time)
- supports upto 500 streaming symbols in real time (version 1.1.0)
- supports all base time intervals: daily, hourly, 15-,5-,1-minute, 15-,5-second, tick
- NO BACKFILL (due to the fact that most DDE sources do not provide backfill)

HISTORY

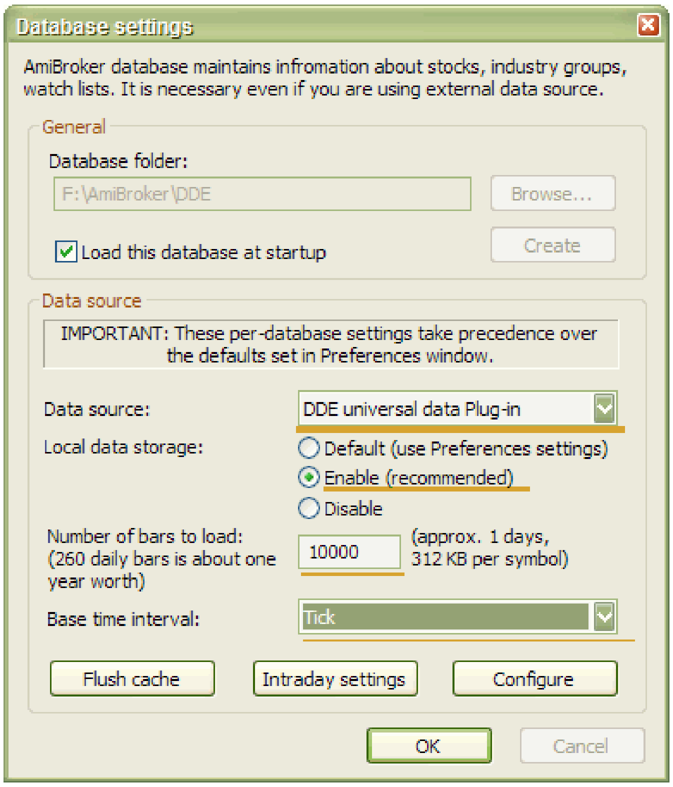
- 1.2.2 - includes "Time shift" field in the context dialog, stores configuration per-database in dde.config file instead of in the registry plus other small improvements
- 1.2.1 - fixed problem with 'type mismatch'
- 1.2.0 - by default plugin uses regional settings numeric format now
  and CPU load is decreased
- 1.1.0 - symbol limit increased from 40 to 500
- 1.0.0 - initial release (BETA)

INSTRUCTIONS

To use DDE data plugin with AmiBroker you need to:

1. (optional *) Download the latest version DDE plugin from http://www.amibroker.com/bin/DDE.dll and copy it to PLUGINS subfolder of AmiBroker directory.
   *Version 1.2.2 of DDE.DLL (Jun 7, 2007) is already included in AmiBroker 5.00 full setup
2. Enable DDE in the 3rd party software you are using as DDE server (consult data vendor/brokerage software documentation for details on how to enable DDE)
3. Run AmiBroker and create new database with "**DDE universal data plugin**" as a data source, following these steps:

- Run AmiBroker
- Choose **File->New database**
- Type a new folder name (for example: C:\Program Files\AmiBroker\DDE) and click **Create** as shown in the picture below:


Database settings

AmiBroker database maintains infromation about stocks, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: F:\AmiBroker\DDE

[✓] Load this database at startup

Data source

IMPORTANT: These per-database settings take precedence over the defaults set in Preferences window.

Data source: DDE universal data Plug-in

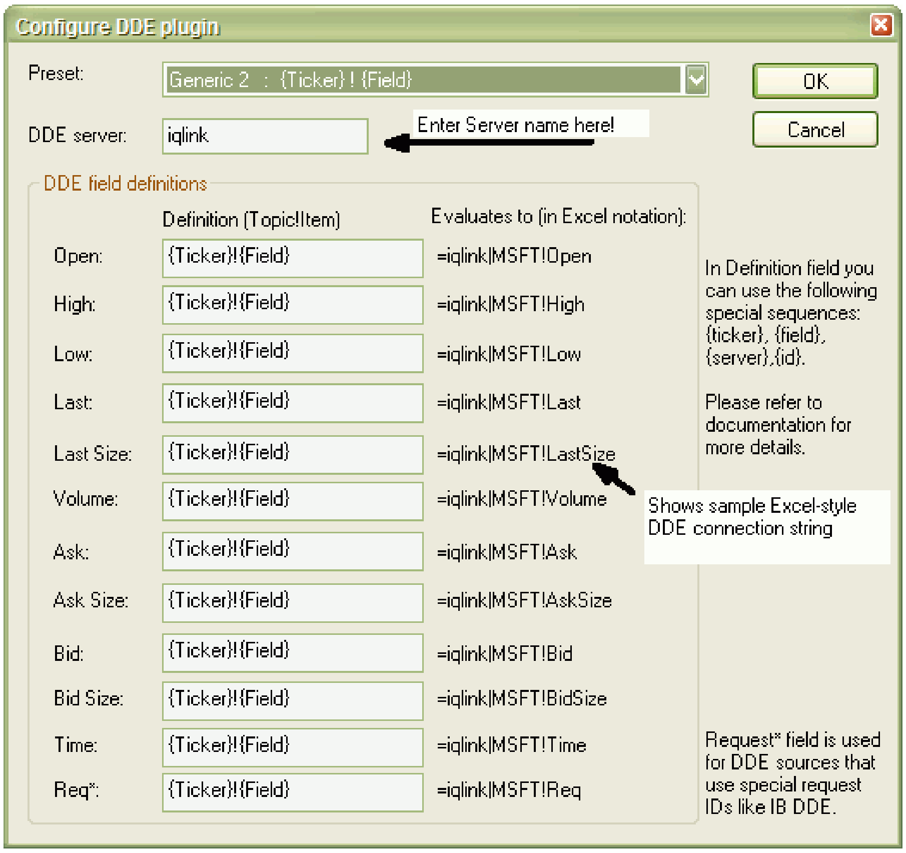
Local data storage: Default (use Preferences settings) [✓ Enable (recommended)] Disable

Number of bars to load: (260 daily bars is about one year worth) 10000 (approx. 1 days, 312 KB per symbol)

Base time interval: Tick

- Choose **DDE universal data plugin** from Data source combo and "**Enable**" from **Local data storage**
- Enter 10000 or more into "**Number of bars to load**" field
- Now choose Base time interval. **Supported intervals are: EOD, hourly, 15-minute, 5-minute, 1-minute. Professional Edition of AmiBroker allows also to select Tick, 5-second, 15-second intervals.**
- Click **CONFIGURE** button - IMPORTANT: in the "CONFIGURE" dialog you have to setup all fields following the description of your data vendor.

  Please check also paragraph below ("**CONFIGURING DDE PLUGIN TO WORK WITH YOUR VENDOR**") for detailed description. ATTENTION: you can not skip this part - without setting up fields specifically for your data vendor, the DDE WILL NOT WORK.


Configure DDE plugin

Preset: Generic 2 : {Ticker} ! {Field}

DDE server: iqlink

Enter Server name here!

DDE field definitions

	Definition (Topic!Item)	Evaluates to (in Excel notation):
Open:	{Ticker}!{Field}	=iqlink|MSFT!Open
High:	{Ticker}!{Field}	=iqlink|MSFT!High
Low:	{Ticker}!{Field}	=iqlink|MSFT!Low
Last:	{Ticker}!{Field}	=iqlink|MSFT!Last
Last Size:	{Ticker}!{Field}	=iqlink|MSFT!LastSize
Volume:	{Ticker}!{Field}	=iqlink|MSFT!Volume
Ask:	{Ticker}!{Field}	=iqlink|MSFT!Ask
Ask Size:	{Ticker}!{Field}	=iqlink|MSFT!AskSize
Bid:	{Ticker}!{Field}	=iqlink|MSFT!Bid
Bid Size:	{Ticker}!{Field}	=iqlink|MSFT!BidSize
Time:	{Ticker}!{Field}	=iqlink|MSFT!Time
Req*:	{Ticker}!{Field}	=iqlink|MSFT!Req

In Definition field you can use the following special sequences: {ticker}, {field}, {server},{id}.

Please refer to documentation for more details.

Shows sample Excel-style DDE connection string

Request* field is used for DDE sources that use special request IDs like IB DDE.

• Click OK.

The Plugin status indicator should change from Yellow "WAIT" to Green "OK" within a few seconds. If it does not turn to "OK" state it means that either:
a) server name and/or fields are not set up correctly
or
b) DDE server (3rd party application) is not running or is not enabled

If indicator shows "OK" - then real time qutoes flow into AB. You can check it by displaying Window->Real time quote. Note: since there is no backfill you would need to wait for at least 3 bars of data to be collected before chart shows up.

CONFIGURING DDE PLUGIN TO WORK WITH YOUR VENDOR

Various data vendors come use different DDE connection strings, here a few typical exampels will be shown.

Most documentation of DDE uses Excel DDE syntax which looks as follows:**=SERVER|TOPIC!ITEM**

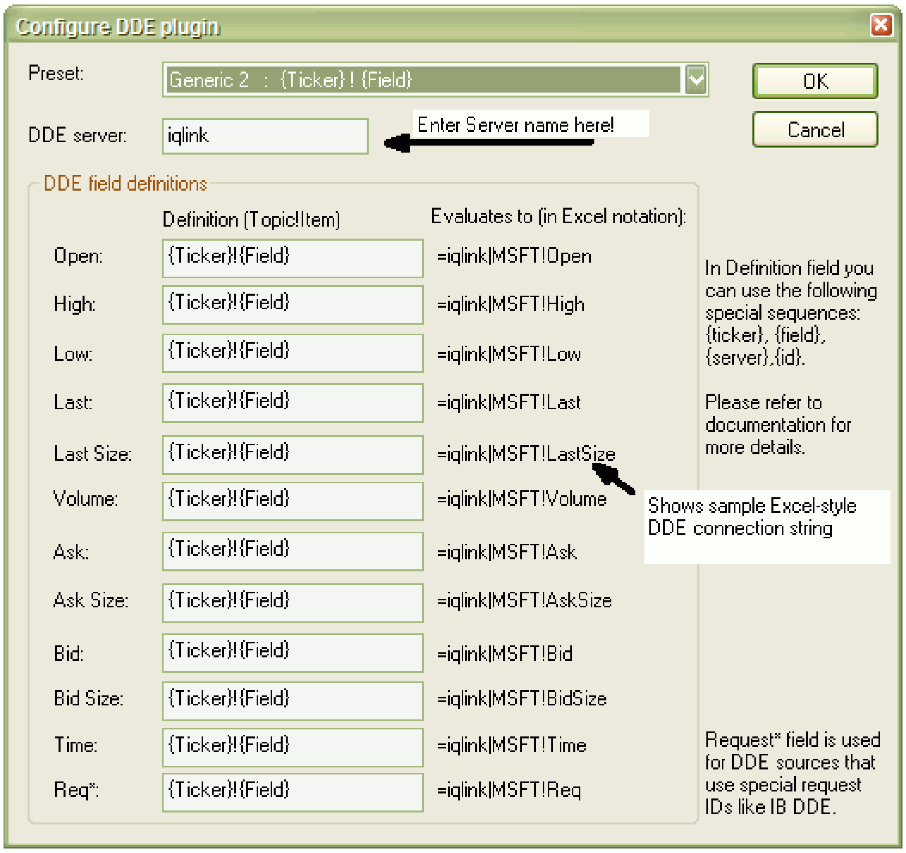
Server is a name of the DDE server such as WINROS, IQLINK, REUTER, CQGPC, MT, MTLink, etc.
Topic is the topic of DDE conversation. Depending on Data source topic may be just the ticker symbol (like in IQFeed), or the field name (like in winros).
Item is the item of DDE conversation. Depending on data source it can be field name (like in IQFeed) or ticker symbol (like in Winros).

So DDE connection string in two most common standards look as follows:

=WINROS|LAST!MSFT

=IQLINK|MSFT!LAST

Now DDE plugin configuration screen looks like this:



Preset: Generic 2 : {Ticker} ! {Field}

DDE server: iqlink

DDE field definitions

	Definition (Topic!Item)	Evaluates to (in Excel notation):
Open:	{Ticker}!{Field}	=iqlink|MSFT!Open
High:	{Ticker}!{Field}	=iqlink|MSFT!High
Low:	{Ticker}!{Field}	=iqlink|MSFT!Low
Last:	{Ticker}!{Field}	=iqlink|MSFT!Last
Last Size:	{Ticker}!{Field}	=iqlink|MSFT!LastSize
Volume:	{Ticker}!{Field}	=iqlink|MSFT!Volume
Ask:	{Ticker}!{Field}	=iqlink|MSFT!Ask
Ask Size:	{Ticker}!{Field}	=iqlink|MSFT!AskSize
Bid:	{Ticker}!{Field}	=iqlink|MSFT!Bid
Bid Size:	{Ticker}!{Field}	=iqlink|MSFT!BidSize
Time:	{Ticker}!{Field}	=iqlink|MSFT!Time
Req*:	{Ticker}!{Field}	=iqlink|MSFT!Req

In Definition field you can use the following special sequences: {ticker}, {field}, {server},{id}.

Please refer to documentation for more details.

Request* field is used for DDE sources that use special request IDs like IB DDE.

In the UPPER part of the dialog you can see "DDE Server" field. In this field you should enter **SERVER** part of DDE connection string **(=SERVER|TOPIC!ITEM)** without equation mark and without | character.

Below you can see 12 text entry boxes where you can define DDE topic and item for each data field your data source provides. Here you should enter **TOPIC!ITEM** pair of the DDE connection string **(=SERVER|TOPIC!ITEM)** with exlamation mark between DDE topic and DDE item.

As you can see in the picture above, DDE plugin allow you to use a few special strings, namely: {Ticker}, {Field}, {FieldSp}, {Server}, {Id} which are evaluated in run-time for each symbol separately allowing to construct dynamic DDE strings (depending on selected ticker for example) required by most data sources:

- {Ticker} - evaluates to ticker symbol of given security
- {Field} - evaluates to the corresponding field name (without spaces), i.e. Open, High, Low, Last, LastSize, Volume, Ask, AskSize, Bid, BidSize, Time, Req
- {FieldSp} - similar to {field} but 2-word field names have spaces, namely: "Last Size", "Ask Size", "Bid Size"
- {Server} - evaluates to server name
- {Id} - evaluates to unique ID (running counter incremented by 1 with each symbol)

All other texts are carbon-copied, so if you write for example:

PREFIX_{Ticker }_SUFFIX!MYTEXT

it will evaluate to =SERVER|PREFIX_MSFT_SUFFIX!MYTEXT (provided that current symbol is MSFT)

Next to field definitions we can see what given definition will evaluate to (in Excel notation). This makes it easy to verify if definition is correct.

Sample evaluation uses always "MSFT" as a {Ticker}, and 34 as {id}.

If your data source does not provide all fields you can make given field empty. Note that for proper operation the "Last" price (the price of last trade) is required. If your data source does not provide "last" price (most of forex sources don't have "last") you can force DDE plugin to use "Bid" instead. For that you should make "Last" field blank and provide appropriate DDE topic!item pair in "Bid" field. Please also note that Topic!Item pairs should evaluate to unique values.

In the top part of the dialog you can see "Preset" combo-box.

As of now it allows to pre-set the fields using two generic schemes:
a) {Field}!{Ticker} - "last price" evaluates to =SERVER|Last!MSFT
b) {Ticker}!{Field} - "last price" evaluates to =SERVER|MSFT!Last

In the future "Preset" box will contain more presets for various DDE source that you submit.

A FEW EXAMPLES

Connection examples are shown on the web page: http://www.amibroker.com/dde.html

TEST PLATFORMS

DDE plugin has been tested and it is known to work properly on Windows XP (32 bit DDE) and Windows 9x (16 bit DDE).The following DDE servers are verified by us to work properly:

- IQLINK (DTN)
- WINROS (eSignal)
- MT (Metaquote)

DDE plugin does NOT work with the following DDE servers:

- VTSPOT (Visual Trader) - due to improper coding in VisualTrader that causes Microsoft DDEML library DdeConnect function to hang on the very first connection attempt

All other DDE servers not listed above should work properly. Contact support at amibroker.com in case of problems.

HELP US TO HELP THE OTHERS:

In order to help the others to configure DDE plugin for their data vendor, once you succeded to link with your particular vendor please drop as a note with a screenshot of the CONFIGURE dialog and name of the source. This will be later included in this document as a reference how to use various data sources. Also working setups will be added to "presets" combo for easy one-click configuration.

NOTES ON DDE PLUGIN:

1. There is **NO BACKFILL** in DDE plugin. You can use however ASCII importer (this includes AmiQuote) to import historical data right into the database that you will update later in real time using DDE plugin.

2. **Change, % change** fields are NOT available (yet)

3. **Time** and **Req** fields are now ignored (this may change in the future)

4. The current system time is used to timestamp each tick.

5. When your source does not offer "LAST" price (like several Forex sources) you should make "Last" field EMPTY in the configuration dialog. This will tell the plugin to use "BID" field instead.

6. Plugin status (connected/disconnected) always initially comes up with "Wait" state (Yellow indicator). It means that no DDE conversation has been established. If at least ONE DDE conversation starts successfully it will turn to "OK" state (green indicator). If DDE server was not running at first attempt to connect, the plugin wil NOT attempt to reconnect automatically. Instead you should force reconnection manually (see point 7).
The indicator may turn to "Disconnected" (red indicator) only in two cases:
a) you were connected properly but DDE server (3rd party app) has been closed
b) you selected "shutdown" from plugin status menu

7. You can reconnect at any time by selecting "reconnect" from plugin status menu.

How to work with Real-Time data plugins

One-time setup

In order to use AmiBroker with any real-time data source you have to set up the database with appropriate data plug-in first. This is required only once at the database creation time. Instructions for setting up are available here: eSignal, myTrack, IQFeed, QuoteTracker.

Check also on-line data sources page at http://www.amibroker.com/quotes.html for new plugins.

Adding symbols

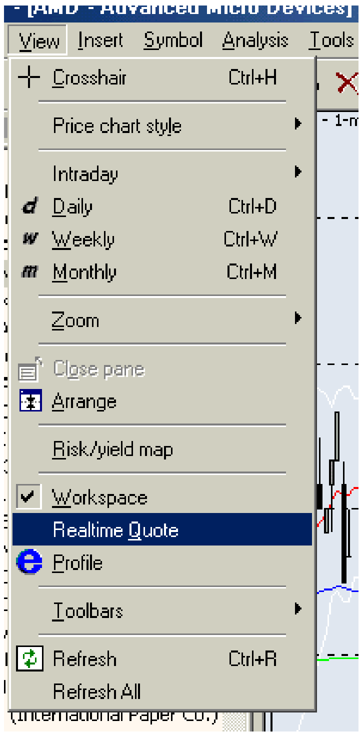
Now you can add symbols to your database. To do so please go to **Symbol->New** menu. In the add symbol dialog enter one or more tickers (comma separated) you wish to add to your database. If you want to see the chart for newly added symbol just select it from the Symbol tree in the workspace window. Please allow few seconds (depending on the speed of your internet connection) to backfill historical data.

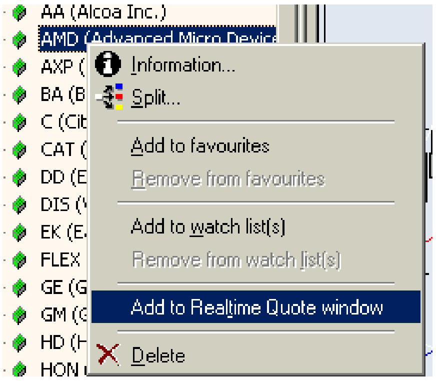
You may add more tickers that your RT account allows. AmiBroker will automatically switch/update/refresh symbols so the most recently used symbols are active and older ones are automatically removed from Data manager. Doing so however may lead to some problems if you exceed your subscription limits too much. So it is advised to use this feature responsibly and not expecting getting 500 symbols while your subscription is limited to only 50.

Note that the above mechanism does not apply to real time quote window and it can not hold more symbols than your subscription limit.

Showing real time quote window

AmiBroker RT features real-time watch window that allows you to watch streaming quotes. To show this window choose **Window->Realtime Quote** menu. (see image to the right ---->)

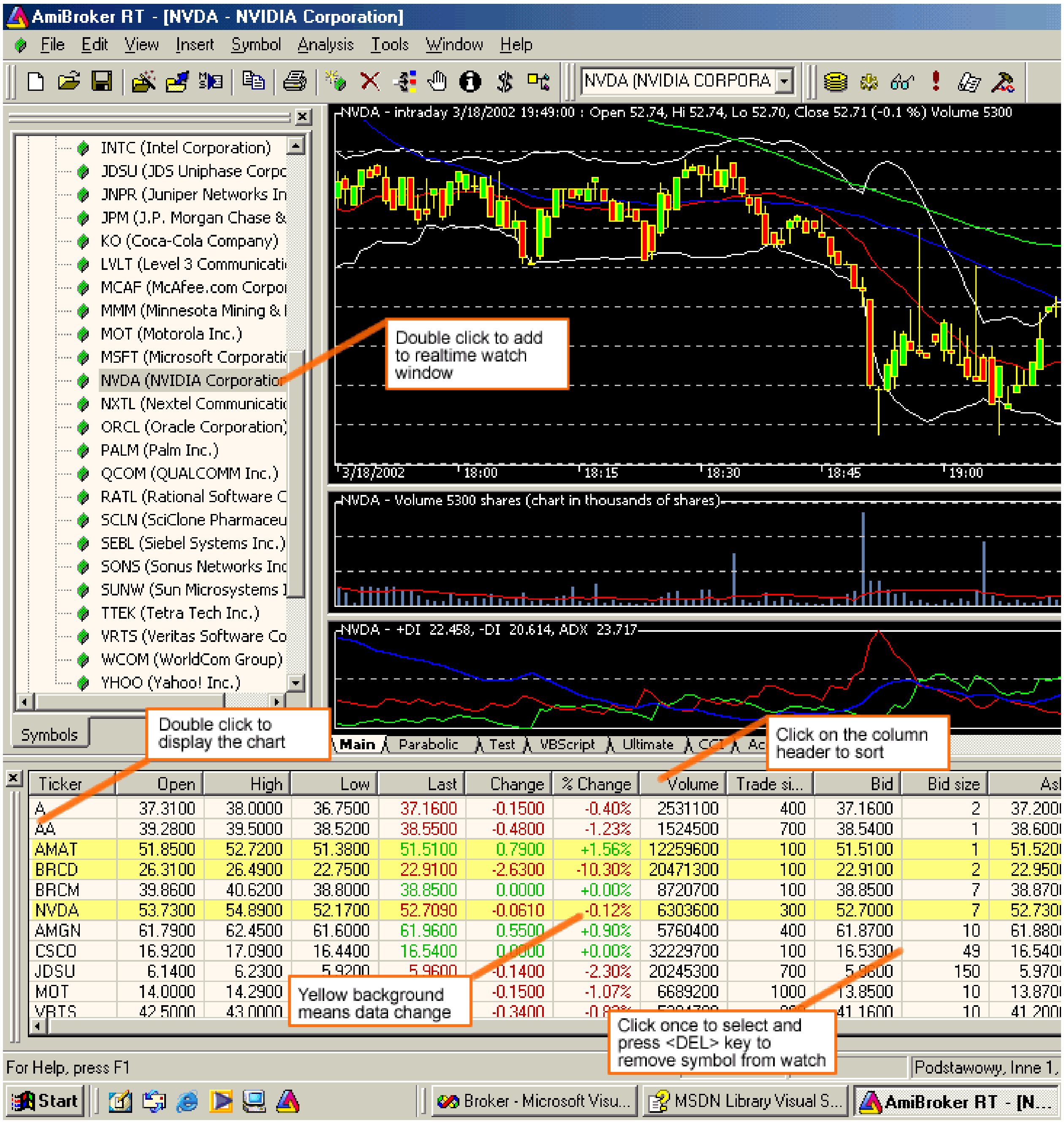




To add symbols to Realtime quote window you either double click on the symbol tree or use right mouse button menu Add to Realtime quote option as shown in the picture above.

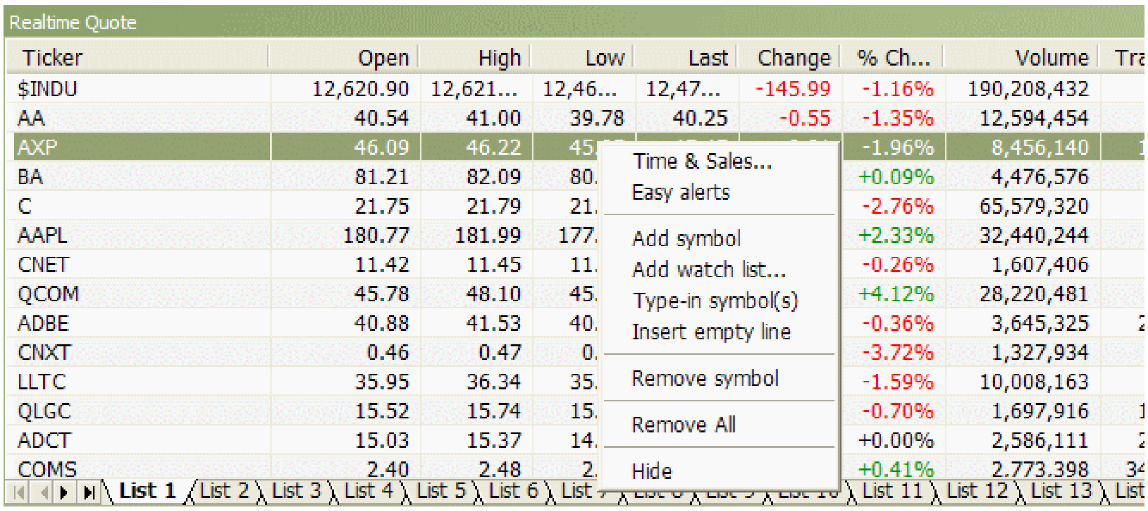
Working with real time quote window

The RT quote window provides real-time streaming quotes and some basic fundamental data. It is fairly easy to operate as shown in the picture below:


AmiBroker RT - [NVDA - NVIDIA Corporation]
File Edit View Insert Symbol Analysis Tools Window Help
NVDA (NVIDIA CORPORA
INTC (Intel Corporation)
JDSU (JDS Uniphase Corpo
JNPR (Juniper Networks In
JPM (J.P. Morgan Chase &
KO (Coca-Cola Company)
LVLT (Level 3 Communicati
MCAF (McAfee.com Corpo
MMM (Minnesota Mining &
MOT (Motorola Inc.)
MSFT (Microsoft Corporati
NVDA (NVIDIA Corporation
NXTL (Nextel Communicati
ORCL (Oracle Corporation)
PALM (Palm Inc.)
QCOM (QUALCOMM Inc.)
RATL (Rational Software C
SCLN (SciClone Pharmaceu
SEBL (Siebel Systems Inc.)
SONS (Sonus Networks Inc
SUNW (Sun Microsystems I
TTEK (Tetra Tech Inc.)
VRTS (Veritas Software Co
WCOM (WorldCom Group)
YHOO (Yahoo! Inc.)
Symbols
NVDA - intraday 3/18/2002 19:49:00 : Open 52.74, Hi 52.74, Lo 52.70, Close 52.71 (-0.1 %) Volume 5300
3/18/2002 18:00 18:15 18:30 18:45 19:00
NVDA - Volume 5300 shares (chart in thousands of shares)
NVDA - +DI 22.458, -DI 20.614, ADX 23.717
Main Parabolic Test VBScript Ultimate CCI Ac
Double click to add to realtime watch window
Double click to display the chart
Click on the column header to sort
Yellow background means data change
Click once to select and press <DEL> key to remove symbol from watch
For Help, press F1
Podstawowy, Inne 1,
Start
Broker - Microsoft Visu...
MSDN Library Visual S...
AmiBroker RT - [N...

Ticker	Open	High	Low	Last	Change	% Change	Volume	Trade si...	Bid	Bid size	Asl
A	37.3100	38.0000	36.7500	37.1600	-0.1500	-0.40%	2531100	400	37.1600	2	37.200
AA	39.2800	39.5000	38.5200	38.5500	-0.4800	-1.23%	1524500	700	38.5400	1	38.600
AMAT	51.8500	52.7200	51.3800	51.5100	0.7900	+1.56%	12259600	100	51.5100	1	51.520
BRCD	26.3100	26.4900	22.7500	22.9100	-2.6300	-10.30%	20471300	100	22.9100	2	22.950
BRCM	39.8600	40.6200	38.8000	38.8500	0.0000	+0.00%	8720700	100	38.8500	7	38.870
NVDA	53.7300	54.8900	52.1700	52.7090	-0.0610	-0.12%	6303600	300	52.7000	7	52.730
AMGN	61.7900	62.4500	61.6000	61.9600	0.5500	+0.90%	5760400	400	61.8700	10	61.880
CSCO	16.9200	17.0900	16.4400	16.5400	0.0000	+0.00%	32229700	100	16.5300	49	16.540
JDSU	6.1400	6.2300	5.9200	5.9600	-0.1400	-2.30%	20245300	700	5.9600	150	5.970
MOT	14.0000	14.2900	[illegible]	[illegible]	-0.1500	-1.07%	6689200	1000	13.8500	10	13.870
VRTS	42.5000	43.0000	[illegible]	[illegible]	-0.3400	[illegible]	[illegible]	[illegible]	41.1600	10	41.200

You can also display context menu by pressing RIGHT mouse button over RT quote window.

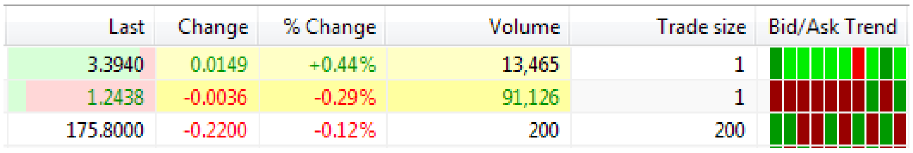


Ticker	Open	High	Low	Last	Change	% Ch...	Volume	Tra
$INDU	12,620.90	12,621...	12,46...	12,47...	-145.99	-1.16%	190,208,432	
AA	40.54	41.00	39.78	40.25	-0.55	-1.35%	12,594,454	
AXP	46.09	46.22	45.			-1.96%	8,456,140	1
BA	81.21	82.09	80.			+0.09%	4,476,576	
C	21.75	21.79	21.			-2.76%	65,579,320	
AAPL	180.77	181.99	177.			+2.33%	32,440,244	
CNET	11.42	11.45	11.			-0.26%	1,607,406	
QCOM	45.78	48.10	45.			+4.12%	28,220,481	
ADBE	40.88	41.53	40.			-0.36%	3,645,325	2
CNXT	0.46	0.47	0.			-3.72%	1,327,934	
LLTC	35.95	36.34	35.			-1.59%	10,008,163	
QLGC	15.52	15.74	15.			-0.70%	1,697,916	1
ADCT	15.03	15.37	14.			+0.00%	2,586,111	2
COMS	2.40	2.48	2.			+0.41%	2,773,398	34

The context menu allows you to access the following options:

- **Time & Sales**
  Opens Time & Sales window that provides information about every bid, ask and trade streaming from the market.
- **Easy Alerts**
  Opens Easy Alerts window that provides way to define realtime alerts executed when bid/ask/last and other fields hit user-defined levels
- **Add Symbol**
  Adds current symbol to Real-Time Quote list
- **Add watch list...**
  Adds entire watch list to real-time quote window
- **Remove Symbol**
  Removes highlighted line (symbol) from the Real-Time Quote list.
- **Remove All**
  Removes all symbols from real-time quote list
- **Hide**
  Hides Real-Time Quote list

Bid/ask trend indicator



Last	Change	% Change	Volume	Trade size	Bid/Ask Trend
3.3940	0.0149	+0.44%	13,465	1	🟩🟩🟩🟩🟩🟩🟥🟩🟩🟩
1.2438	-0.0036	-0.29%	91,126	1	🟥🟥🟥🟥🟥🟥🟥🟩🟥🟩
175.8000	-0.2200	-0.12%	200	200	🟩🟩🟥🟥🟩🟥🟩🟥🟩🟥

Version 5.90 adds Bid/Ask trend - a graphical indicator showing the direction of 10 most recent changes in real-time bid/ask prices. The right-most box is most recent and as new bid/ask quotes arrive they are shifted to the left side. Color coding is as follows:

- **Dark green:** bid > previous bid OR ask > previous ask
- **Bright green:** bid > previous bid AND ask > previous ask
- **Dark red:** bid < previous bid OR ask < previous ask
- **Bright red**: bid < previous bid AND ask < previous ask
- **Red / Green box:** ask < previous ask AND bid > previous bid
- **Green / Red box:** ask > previous ask AND bid < previous bid

If bid/ask prices don't change there is no new box. NOTE: This column works only if there are real-time quotes streaming (markets are open)

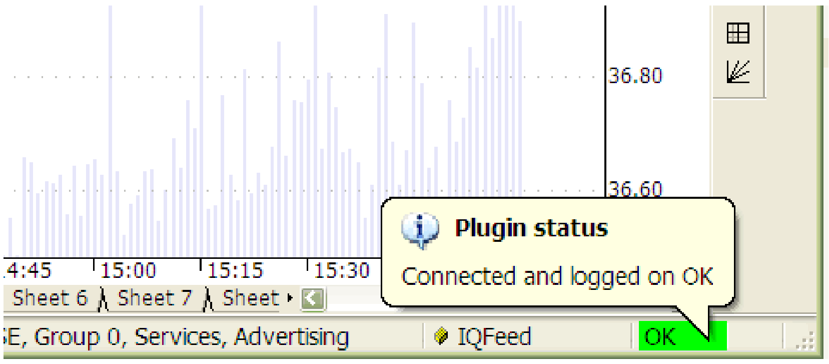
Working with intraday and daily charts

If your data source supports mixed EOD/Intraday mode (such as eSignal or IQFeed) you can use single database for both types of charts.

However if your data source does not support mixed EOD/Intraday mode and if you want to have long daily histories AND intraday charts you should consider running TWO instances of AmiBroker. One for EOD charts and second for intraday charting. Both may use the same real-time data source.

Connection status display

The data plug-in connection status is displayed in the plugin status display area located in the lower right part of the AmiBroker main window as shown in the picture below. When connection status changes AmiBroker plays a beep sound and pops up bubble tool tip to inform about status change.



The bubble tip provides more detailed information text and disappears automatically after 2 seconds.

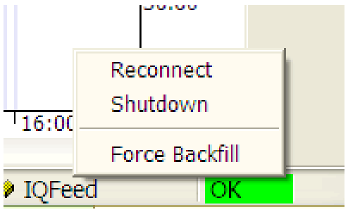
If you want to re-display it just hover your mouse over plugin status display area.

To enable quick examination of connection status AmiBroker displays color coded information:

- **OK (green light)** means that connection is OK and indicates correct operation of the plugin
- **WAIT (yellow light)** means that connection is being set-up right now or the plugin is connected only partially (to few of many servers). Usually this state is transient and within few seconds the status comes back to "OK".
- **ERR (red light)** means that connection is broken. It may mean invalid user name/password for your subscription, or the fact that some 3rd party component / program required is not running (for example if QuoteTracker is not running and you are using QuoteTracker plugin). This state usually requires some user intervention such as checking/fixing user/password in File->Database Settings->Configure or running required component. When you fix the reason the plugin will automatically attempt to reconnect (and if reconnect is successfull then "OK" will be displayed)
- **SHUT (purple light)** means that some serious problem occurred and the plugin will not attempt to reconnect automatically. In most cases you have to first fix the problem that caused this state and then reconnect manually using plugin context menu described below. Alternativelly you can just restart AmiBroker.

Using plugin context menu

Real time plugins provide some additional controls via plugin context menu. This context menu is available when you click with RIGHT mouse button over plugin status display area. If you do, the menu like this will be displayed:



Please note that various plugins offer various options in this menu, however most plugins provide at least 3 basic and useful options:

- **Reconnect** - this option allows you to reconnect manually. Most RT plugins attempt to reconnect automatically, but sometimes manual reconnect is necessary.
- **Shutdown** (Disconnect) - this allows to shutdown RT plugin. This is useful when you want to stop streaming of quotes.
- **Force backfill** - this option causes that plugin re-downloads entire (intraday) history from the server. Usually the plugin automatically handles all backfills so you don't need to trigger backfills by hand. If the plugin detects that you have some missing quotes from last available bar till current date/time it triggers backfills and it is all automatic. But... in at least two cases this option is useful:
    - ♦ backfilling more bars after settings change (when you enlarge 'number of bars to load' in File->Database Settings dialog you have to force backfill for symbols that were backfilled previously with smaller number of bars)
    - ♦ cleaning up bad ticks (when you see a bad tick you may try forcing backfill in hope that data vendor has cleaned up its database and you will get fixed data - works well for eSignal that really appears to fix bad ticks after they happen)

Things you should NOT do, or you should do very carefully

You should note the fact that when you are using data plugin then the plugin controls the quotation database (see Understanding database concepts article), therefore you should NOT import quotes from ASCII files (this includes AmiQuote) for symbols that are already present in the real-time database.

If you do, the plugin will eventually overwrite your imports with the real-time data or your database will become corrupted (if you import end-of-day data over intraday database).

So please do not import ASCII (especially EOD data) into real-time intraday database fed by the plugin.

You may ask: why this is not disabled at all. The answer is that sometimes it is useful and sometimes it will work (but these are rare cases). For example it will work if you import INTRADAY data into the intraday database fed by QuoteTracker plugin and both the database and imported data have exactly the same bar interval.

It also works if you import the data for symbols that are NOT present in the database. In this case newly imported symbols are marked by ASCII importer as "use only local database for this symbol" (See Information window for details), so they are EXCLUDED from the real-time update. This is useful if you want to import some other data (even not quote data) and access it via Foreign function while using your real-time database.

So ASCII import is not disabled in real-time database but you have to use it with extreme care and know what you are doing.

Second thing is using Quote Editor. Although data are controlled by the plugin it is in most cases possible to use Quote Editor. However please note that you will be able to edit only 1-minute data or higher interval, and you will be able only to edit symbols that are backfilled completely (there is no running backfill for the particular symbol) and you will NOT be able to edit last three bars. This is so because last three bars are cached in the plugin. So you will be able to edit them only when new bars arrive making them 'older' than last three.

'WAIT FOR BACKFILL' feature

The users of eSignal, myTrack and IQFeed real-time plugins may now check "wait for backfill" box in the Automatic analysis window and all scans, explorations and backfills will wait for completion of backfill process for given symbol. This flag has no effect on databases that do not use plugins (external data sources) or use end-of-day plugins (like FastTrack, QP2, TC2000/TCNet, etc). This flag has also no effect when using QT plugin due to the fact that QuoteTracker manages backfills by itself and does not provide any control of backfill process to 3rd party applications.

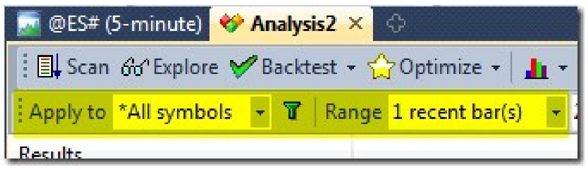
BACKFILLING ALL SYMBOLS AT ONCE

To backfill all symbols at once in the data source that supports "Wait for backfill" feature (IQFeed, eSignal), one can use Analysis window. The procedure is as follows:

1. Open Formula Editor and type a simple single-line rule like below and choose **Tools->Send to Analysis**

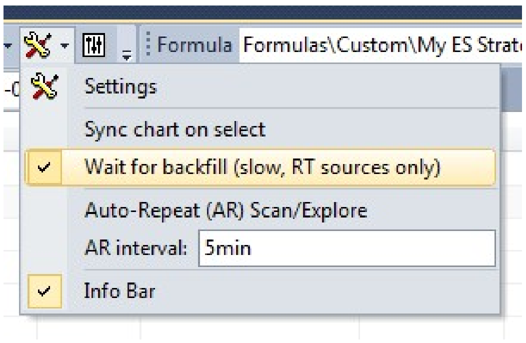
```
    Buy = 1;
```

2. In the Analysis window select **Apply to: *All symbols** and **Range: 1 recent bar**



Apply to *All symbols
Range 1 recent bar(s)

and turn on **Wait for backfill** option.



Formula Formulas\Custom\My ES Strat

✓ Wait for backfill (slow, RT sources only)

AR interval: 5min

✓ Info Bar

3. Press **Scan** button

The Analysis window will iterate through all symbols, requesting backfill for each symbol and waiting until the data arrive, so at the end of the scan all symbols will be backfilled.

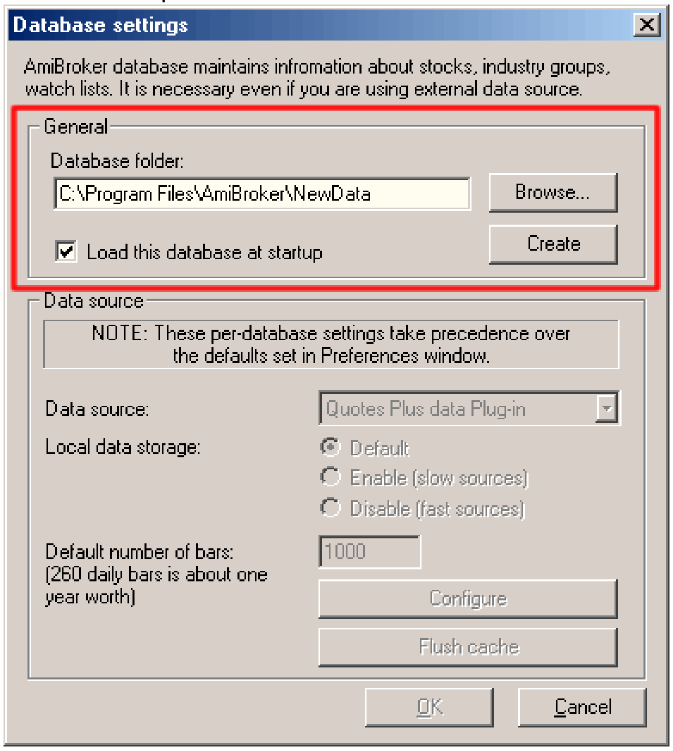
How to use AmiBroker with external data source (Quotes Plus, TC2000/TCNet/TC2005, FastTrack, Metastock)

One of the new features introduced in AmiBroker version 3.90 is the ability to read directly external databases. This is achieved by means of data plug-in DLLs that allow to link AmiBroker database with an external source. Please note that althrough you will be using external database, you will still need an AmiBroker database for storing additional information that is not supported by the external source like hand-drawn studies, assignments to groups, watch lists, composites and so on. You can find more information on AmiBroker database handling here.

One-time setup

To use an external data source with AmiBroker you will need to perform a one-time setup described below:

- Run AmiBroker
- Choose **File->New database**
- Type a new folder name (for example: C:\Program Files\AmiBroker\NewData) and click **Create** as shown in the picture below:



Database settings

AmiBroker database maintains infromation about stocks, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: C:\Program Files\AmiBroker\NewData

[✓] Load this database at startup

Data source

NOTE: These per-database settings take precedence over the defaults set in Preferences window.

Data source: Quotes Plus data Plug-in

Local data storage: [● Default] / Enable (slow sources) / Disable (fast sources)

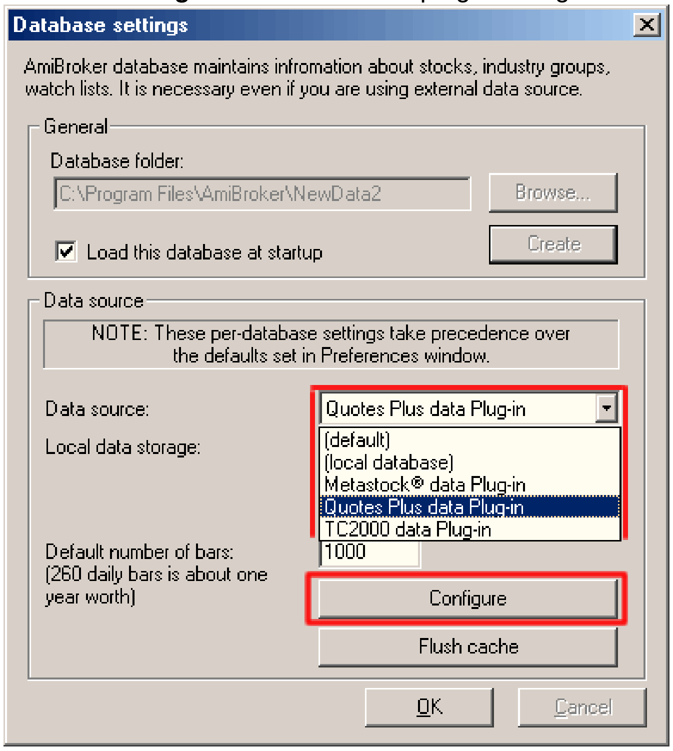
Default number of bars: (260 daily bars is about one year worth) 1000

- Choose appropriate entry from Data source combo:
    - **Quotes Plus users** select "**Quotes Plus plug-in**" as a **Data Source** and "**Disable**" from **Local data storage**
    - **TC2000/TCNet users** select "**TC2000/TCNet plug-in**" as a **Data Source** and "**Enable**" from **Local data storage**
    - **TC2000 for Mutual Funds users** select "**TC2000 Mutual Funds plug-in**" as a **Data Source** and "**Enable**" from **Local data storage**
    - **TC2005 users** select "**TC2000/TCNet plug-in**" as a **Data Source** and "**Enable**" from **Local data storage**

Note: TC2005 users may need to follow these instructions (click here) if TC2000 plugin does not show up.

- ♦ FastTrack users select "**FastTrack plug-in**" as a **Data Source** and "**Disabled**" from **Local data storage**
- ♦ Metastock users select "**Metastock plug-in**" as a **Data Source** and "**Disable**" from **Local data storage**

• Click on **Configure** button to show plugin configuration dialog as shown below



Database settings

AmiBroker database maintains infromation about stocks, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: C:\Program Files\AmiBroker\NewData2

Load this database at startup [✓]

Data source

NOTE: These per-database settings take precedence over the defaults set in Preferences window.

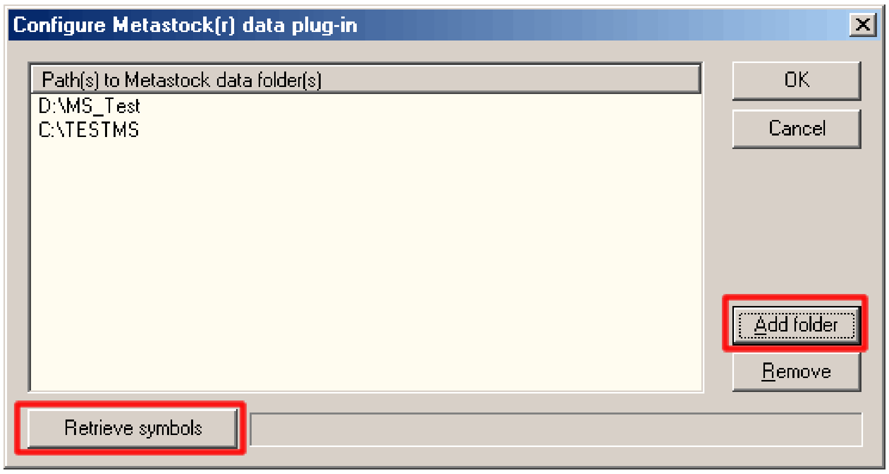
Data source: Quotes Plus data Plug-in

Local data storage:

(default)
(local database)
Metastock® data Plug-in
Quotes Plus data Plug-in
TC2000 data Plug-in

Default number of bars: (260 daily bars is about one year worth) 1000

• **<u>Metastock plug-in only</u>** *(skip this point in case of TC2000, Quotes Plus, FastTrack)*:
Click on the "Add folder" button to add Metastock database directory as your data source (browse for Metastock MASTER file and click OK) as shown below:

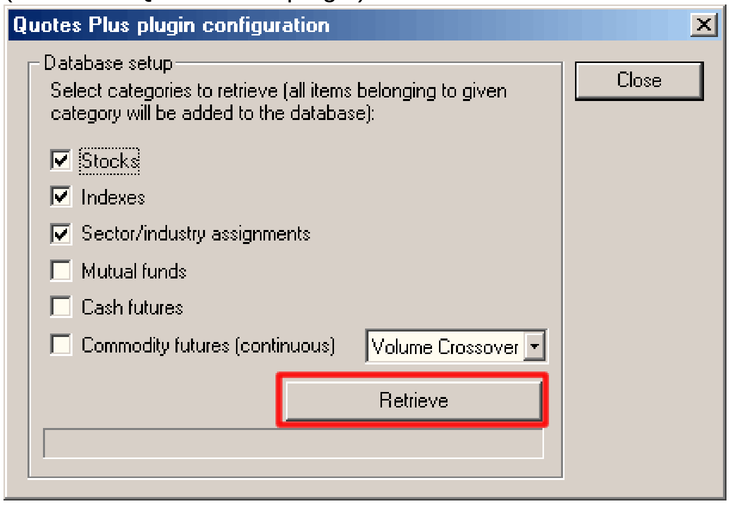


Configure Metastock(r) data plug-in

Path(s) to Metastock data folder(s)
D:\MS_Test
C:\TESTMS

- you can add unlimited number of Metastock directories effectivelly overcoming MS 4096 symbols limitation.

- Click **Retrieve** button - this will setup a new database with all symbols and full names. Quotes Plus and TC2000 plugins will also setup your sectors/industries names and assignments, as shown below (in case of Quotes Plus plugin):


Quotes Plus plugin configuration
Database setup
Select categories to retrieve (all items belonging to given category will be added to the database):
Close
Retrieve

[✓] Stocks
[✓] Indexes
[✓] Sector/industry assignments
[ ] Mutual funds
[ ] Cash futures
[ ] Commodity futures (continuous) Volume Crossover

From now on your AmiBroker reads quotes directly from the external data source. No need to import/update quotes anymore. **All new quotes will appear automatically without user intervention**.

IMPORTANT: If there are **new symbols** added or **old symbols** deleted to/from the external data source, you will need to go to File->Database Settings->Configure and click "RETRIEVE" again to get new symbols.

Plug in performance notes

Using AmiBroker native database gives absolutely the best performance (it takes less than 2 milliseconds to retrieve 1000 data bars).
Metastock plugin is also quite fast, as it can retireve 1000 bars in about 6-7 milliseconds (including looking up for symbol in 5 different directories). In fact AmiBroker can access Metastock data faster than Metastock itself :-)
Quotes Plus performance depends on various factors - first access can be much slower (0.1-0.2 sec for 1000 bars) but subsequent accesses are faster (downto 5 milliseconds). FastTrack plugin is as fast as Quotes Plus plugin.
TC2000 is not as fast, especially if you are using data only on CD. So it is advised to copy your database to hard disk for better performance. But still, even when using CD-only data, AmiBroker ca access 1000 bars from TC2000 in about 0.25 sec (first access) and 0.015 sec (subsequent accesses). Also it is advised to enable "**Local data storage**" when using TC2000 plugin because it gives tremendous (>10 times) speed-up (once you access the TC2000 data, AmiBroker caches them in its own native database for fast retrieval).

Times are approximate and do not include one-time plug-in initialization process. Measurements where done on fairly low-end Celeron 600 based computer with 196KB RAM and 24x CD-ROM

In-memory caching

By default AmiBroker holds only 10 the most recently accessed symbols' data in RAM. This takes up about 320 KB (yes, kilobytes) of memory for 1000 bars per symbol loaded. You can enlarge "**In memory cache**" (**Tools->Preferences** : "**Data**" tab) to 100 (approx. 3.2MB additional RAM consumption) or 1000 (approx. 32MB additional RAM consumption) or even more to get much better performance for subsequent data access (once data are in RAM AmiBroker does not need to ask plugin again and again)

How to update US quotes automatically using AmiQuote

QUICK START

Run AmiBroker

Choose **Tools->Auto-update quotes (US & Canada)**

HOW IT WORKS

AmiQuote loads (or retrieves from AmiBroker) a ticker list file (.TLS) which is simple ASCII file with ticker symbols, then parses it and generates URLs to the Yahoo! finance site based on ticker name, mode (current quotes or historical), country and From/To date. Then, when you start the download process, it requests the data from Yahoo and stores downloaded data in the separate .AQD (daily) or .AQH (historical) files for each ticker. After download, if AmiBroker is running, AmiQuote will import the quotes into AmiBroker automaticaly.

USAGE

Automatic update

The easiest method to work with AmiQuote and AmiBroker is to use the procedure given in **Quick Start** section of this document. Just run AmiBroker and AmiQuote and choose **Tools->Auto-update AmiBroker database**. This method updates historical quotes from the last date present in AmiBroker upto today. When performing automatic update, AmiQuote performs internally 4 steps a) retrieves the ticker list from AmiBroker (all symbols loaded currently in AmiBroker);
b) gets the last quotation date available in AmiBroker;
c) performs historical download from last date upto today;
d) instructs AmiBroker to import downloaded files.

Please note that this procedure works only for US & Canada markets, because Yahoo provides historical quotes only for that markets.

Note that AmiQuote currently supports a new command line parameter: **/autoupdate**. This option forces AmiQuote to perform automatic update procedure without user intervention.
By default AmiBroker's Tools menu is configured as follows:

```
C:\Program Files\AmiBroker\AmiQuote\Quote.EXE /autoupdate
```

So, you are able to update your US database with a single click on **Tools->Auto-update quotes (US & Canada)** in AmiBroker

Manual operation

Automatic mode is nice but there are cases when you have to perform some tasks manually. There is a good old document describing that mode of operation at: How to download quotes manually using AmiQuote . Everything written in this document remains valid with one exception - now importing to AmiBroker are performed automatically if you have **Automatic import** checkbox marked.

There are also several cases when you prefer to do things manualy, then please don't forget about some useful tools available at your fingertips:

File->Open, File->Save, File->Save As

These functions enable you to load and save your edited ticker lists for future repeated use.

Edit->Add tickers

This function allows you to add the tickers to the list. Just type space separated tickers into the field that will show up when you choose this function.

Edit->Delete tickers

This function allows you to delete tickers from the list. Just select the items you want to delete from the list view (multiple selection possible by holding SHIFT or CTRL key while clicking on items), and choose this function.

Edit->Mark all, Edit->Unmark all, Edit->Toggle, Edit->Mark selection, Edit->Unmark selection

These functions allow you to mark the tickers for download. Please note that AmiQuote puts a checkmark before ticker name in the list view. ONLY MARKED items will be downloaded. This allows you to perform selective downloads/updates.

View->Refresh

Basically AmiQuote handles refreshes by itself when needed. For example if you changed the date range, the list will be refreshed before starting download. But there are some cases when you may want to refresh the list by yourself. For example if you downloaded and imported quotes once and want to do this again you would need to choose this function. The Refresh function simply applies all date and type settings to the URLs listed, and MARKS all tickers for a new download.

Tools->Import into AmiBroker

This function is useful if you want to import just downloaded quotes into AmiBroker but you have **Automatic import** checkbox cleared.

Tools->Get tickers from AmiBroker

This function retrieves all symbols from currently loaded AmiBroker database and fills the AmiQuote ticker list with them.

Tools->Get last update date

This function retrieves the date of the most recent quotation of the first symbol present in currently loaded AmiBroker database and sets the **From** date to this date.

Tools->Settings

Displays the settings window where you can define the destination directory where all downloads are stored. Note that blank destination directory means that downloads will be stored in the current working directory (in most cases this is the folder from where current .TLS file was loaded).

In this window you can also change the mode of writing the files. By default historical files are overwritten while daily files get appended. This is recommended setup. Appending daily files simply allows you to create intra-day historical files when you do the updates daily. You may change this behaviour for your particular purpose.

How to download quotes manually using AmiQuote

Introduction

The purpose of this document is to explain how to use AmiQuote and AmiBroker in order to obtain quotes from Yahoo finance and Quote.com sites. AmiQuote is a companion program to AmiBroker charting/analysis software. The main purpose of AmiQuote is to simplify and automate downloading daily and historical quotation data from free Yahoo! Finance (USA, major European exchanges and some other countries), Quote.com (USA only) sites, MSN (USA and some European exchanges), Integratir (US stocks), Forex (Finam free site)

Yahoo provides data in "Historical" and "Current" modes of AmiQuote. Quote.com provides data in "Intraday" mode of AmiQuote.

Preparing ticker list

A ticker list is a simple text file which lists line by line the tickers you want to import. The AmiQuote ticker list file has .TLS extension. AmiQuote comes with pre-written ticker list for components of main NYSE and NASDAQ indices and a number of European indices/markets. Additional ticker lists are available on the starter page at: http://www.amibroker.com/starter/. You can use those pre-written ticker lists or you can customize them or write your own one. In order to edit existing .TLS file or write completely new one all you need is plain text editor such as Notepad or any other plain ASCII editor (not MS Word!). All you have to do is to write tickers you want to import line by line (single ticker in single line) and save the file. Please make sure that you are saving the file with .TLS extension. Otherwise AmiQuote will not load this file.

Please note that Yahoo uses suffixes for non-US stocks. So in order to get quotes for non-US symbol you would need to add appropriate suffix to the ticker symbol. The suffixes in alphabetical order are (you can click on link to get the symbol list for each exchange) : .AS - Amsterdam, .AX - Australia (ASX), .BC - Barcelona, .BE - Berlin, .BO - Bombay, .BM - Breman, .BR - Brussels, .BA - Buenos Aires, .CL - Calcuta, .CR - Caracas, .V - CDNX, .CO - Copenhagen, .D - Dusseldorf, .F - Frankfurt, .H - Hamburg, .HA - Hanover, .HK - Hong Kong, .I - Ireland, .JK - Jakarta, .KA - Karachi, .KQ - Kosdaq, .KS - KSE, .KL - Kuala Lumpur, .L - London, .LM - Lima, .LS - Lisbon, .MA - Madrid, .MX - Mexico, .MI - Milan, .MU - Munich, .NS - NSE, .NZ - New Zeland, .OL - Oslo, .PA - Paris , .SN - Santiago, .SS - Shanghai, .SZ Shenzhen, .ST - Stockholm, .SG - Stutgart, .TW - Taiwan, .TA - Tel Aviv, .TO - Toronto, .VA - Valencia, .VI - Viena, .DE - XETRA, .S - Zurich.

Please note that also Yahoo and Quote.com use different symbols for indices. The main difference is that Yahoo uses ^ (dash) prefix and Quote.com uses $ (dollar) prefix.

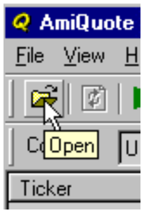
For list of indices provided by **Yahoo** please click here.

For list of indices provided by **Lycos/Quote.com** please click here. Please note that recently Lycos/Quote.com stopped delivering free quotes and you need to have Livecharts subcription ($9.95/month) in order to use it. For more details see this Knowledge Base article.

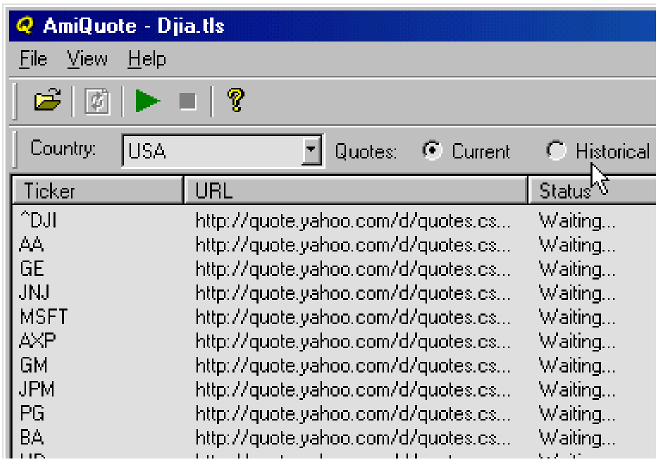
For list of symbols provided by **MSN** please click here.

Downloading data

In order to download the data please launch AmiQuote. Then please click on "Open" button in the toolbar (or choose ***File***->***Open*** menu) as shown in picture on the right.



From the file dialog please choose one .TLS file (for example DIJA.TLS) and click ***Open*** button. The you will see the main screen of AmiQuote filled with the list of tickers loaded, as show in picture below.



Ticker	URL	Status
^DJI	http://quote.yahoo.com/d/quotes.cs...	Waiting...
AA	http://quote.yahoo.com/d/quotes.cs...	Waiting...
GE	http://quote.yahoo.com/d/quotes.cs...	Waiting...
JNJ	http://quote.yahoo.com/d/quotes.cs...	Waiting...
MSFT	http://quote.yahoo.com/d/quotes.cs...	Waiting...
AXP	http://quote.yahoo.com/d/quotes.cs...	Waiting...
GM	http://quote.yahoo.com/d/quotes.cs...	Waiting...
JPM	http://quote.yahoo.com/d/quotes.cs...	Waiting...
PG	http://quote.yahoo.com/d/quotes.cs...	Waiting...
BA	http://quote.yahoo.com/d/quotes.cs...	Waiting...

Choose appropriate Data Source

- **Yahoo Historical** - allows you to download end-of-day histories upto current day (current day data appear few hours after session end)
- **Yahoo Current** - allows you to download current day quotes (15-min delayed) during the trading session
- **Lycos/Quote.com Intraday** - allows you to download intraday and daily historical data (1-min bars and up) - for US stocks/futures only. If you have choosen this mode you should also select the bar interval (see the limitations described below) - need Livecharts subscription ($9.95/month)
- **MSN Historical** - allows you to download end-of-day histories upto current day (current day data appear few hours after session end)
- **Forex** - allows you to download end-of-day and intraday (registered version) histories for the following currency pairs: EURCHF, EURGBP, EURJPY, EURUSD, GBPUSD, USDCHF, USDJPY

After choosing correct options please click on green arrow (or use ***File*** -> ***Start Download*** menu). The download process will begin. AmiQuote will display progress messages and status information including number of completed downloads and number of files left. At anytime you can stop download process with "Stop" button (red box). After finishing the download **AmiQuote will automatically update the quotes in AmiBroker** (if only AmiBroker is running in parallel and "automatic import" box in AmiQuote is checked.

Limitations

Intraday interval bar data (1-min, 5-min, 15-min, 60-min and 120-min) are available for US securities only. Historical data for international exchanges are usually much shorter than for US markets.

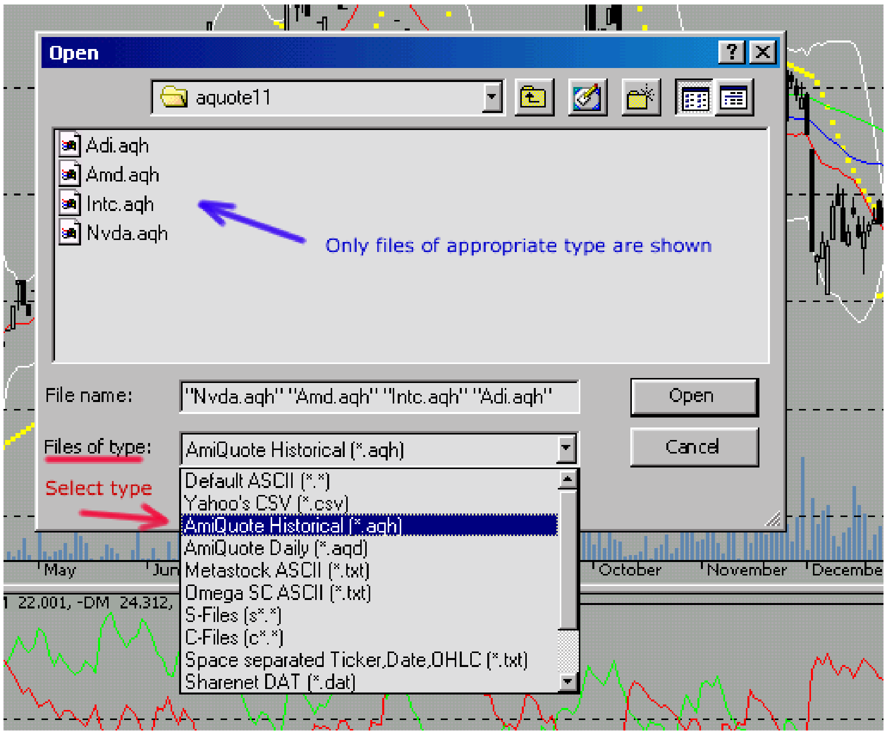
Because intraday bar data are downloaded from Quote.com servers the ticker symbols for indices are different than those used by Yahoo. For complete reference please check http://finance.lycos.com/home/misc/symbol_search.asp?options=i

Intraday bar data are limited to 500 bars regardless of bar interval. In other words you always get 500 bars data, whenever these are 1-min, 5-min, 15-min, 60-min or 120-min data - so by choosing bigger interval you get data from more days. This is the limitation imposed by Livecharts server.

Importing quotes into AmiBroker

NOTE: This step is no longer necessary if you are using "automatic import" feature of AmiQuote. The explanations are provided only for users wanting to import selectively or re-import files downloaded in the past.

First, please launch AmiBroker. From the ***File*** menu please select ***Import From ASCII*** option. You will see the following file dialog:



File name: "Nvda.aqh" "Amd.aqh" "Intc.aqh" "Adi.aqh"

Files of type: AmiQuote Historical (*.aqh)

In this picture I marked the most important items for easy identification. Marked with red is type selector combo-box ("***Files of type***"). In order to import AmiQuote files (those with .AQH and .AQD extensions you should choose ***AmiQuote Historical*** or ***AmiQuote Daily,*** or ***AmiQuote Intraday (.AQI)*** or ***AmiQuote MSN (.AQM)*** or ***AmiQuote eSignalCentral (.AQE)*** from the combo box (red arrow shows those options).

After choosing right type you will see only files of appropriate type in the file list (blue arrow shows that). Now you can select one or more files from the list. Multiple selection is possible by holding CTRL key depressed while selecting the items with a mouse (you can also press SHIFT for choosing a range of files with a single click). Now when you are done choosing the files you want to import just click "***Open***" button. The import process will start and you will see progress bar showing the AmiBroker is importing the data. After finishing the import AmiBroker will automatically refresh symbol list and you will see updated tickers and charts. If anything goes wrong with the import process AmiBroker writes a log file called "import.log" and located in AmiBroker's main directory. You can watch this log file if you want to find out what went wrong (since import.log is simple text file you can open it with any text editor)

Common questions

Question	**Answer**
How can I edit my own ticker list (.TLS) file?	You can create or edit .TLS using Windows Notepad. When saving a file simply give .TLS extension to the file (instead of the default. TXT)
What about ready-to-use complete ticker lists for NYSE, NASDAQ, AMEX?	There are following ready-to-use ticker lists available for download: • DJIA.TLS (30 stocks) • DJTA.TLS (20 stocks) • DJUA.TLS (15 stocks) • NASDAQ100.TLS (100 stocks) • NYSE.TLS (2612 stocks) • NASDAQ.TLS (4464 stocks) • AMEX.TLS (794 stocks)

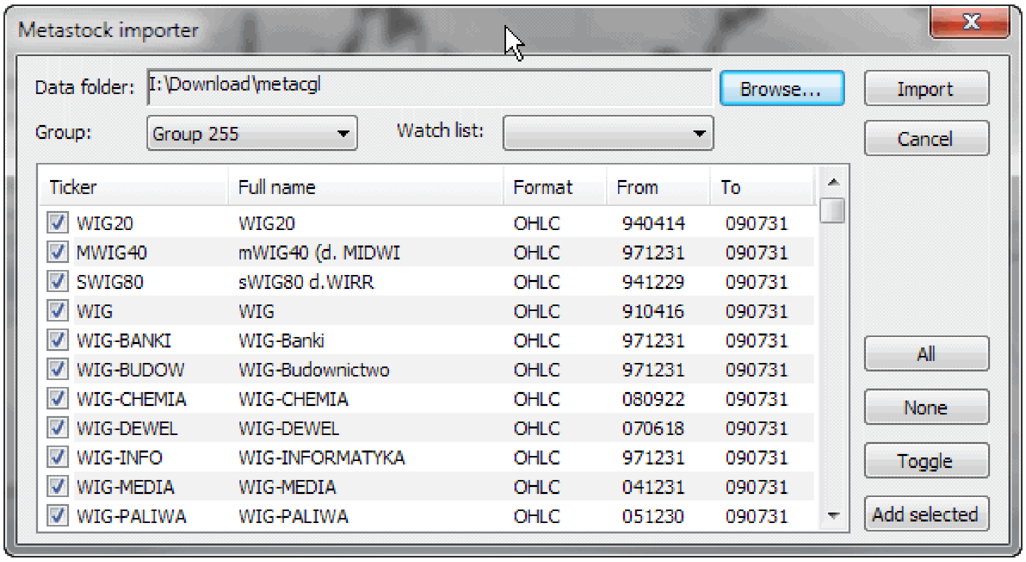
Further information

For further information please consult AmiBroker User's Guide section "*Data management - Importing data from ASCII file".* In case of any further questions, comments and suggestions please use customer support page at http://www.amibroker.com/support.html

Metastock importer window

IMPORTANT NOTE: Metastock importer should be used ONLY if you want to import MS data to native, local AmiBroker database once. If you want AmiBroker to just read Metastock database ***DIRECTLY*** *without need to import new data over and over please set up your database WITH METASTOCK PLUGIN as described in the Tutorial.*

NOTE 2: if you setup your database with the ***MS plugin*** *you should NOT use Metastock importer, because there is no point in using it when your data are already fed by the plugin.*



Data folder: I:\Download\metacgl

Group: Group 255

Watch list:

Ticker	Full name	Format	From	To
✓ WIG20	WIG20	OHLC	940414	090731
✓ MWIG40	mWIG40 (d. MIDWI	OHLC	971231	090731
✓ SWIG80	sWIG80 d.WIRR	OHLC	941229	090731
✓ WIG	WIG	OHLC	910416	090731
✓ WIG-BANKI	WIG-Banki	OHLC	971231	090731
✓ WIG-BUDOW	WIG-Budownictwo	OHLC	971231	090731
✓ WIG-CHEMIA	WIG-CHEMIA	OHLC	080922	090731
✓ WIG-DEWEL	WIG-DEWEL	OHLC	070618	090731
✓ WIG-INFO	WIG-INFORMATYKA	OHLC	971231	090731
✓ WIG-MEDIA	WIG-MEDIA	OHLC	041231	090731
✓ WIG-PALIWA	WIG-PALIWA	OHLC	051230	090731

Metastock importer opens AmiBroker to very rich source of historical data. The importer supports both old Metastock 6.5 and new 7.x (XMASTER) formats.

Basically Metastock data consist of:

- MASTER/EMASTER file which holds general information about the tickers, stock names, etc.
- F1.DAT....Fxx.DAT files which hold actual quotation data

The MASTER/EMASTER file is essential because it holds the references to Fxx.DAT files. Fxx.DAT files store only quotations in either 5 field (date/high/low/close/volume), 6 or 7 field (date/open/high/low/close/volume/openinterest) format. As you see MASTER/EMASTER and Fxx.DAT files are closely connected and you need them all to import the data.

Usage

To import Metastock data you should do the following:

- Choose *Metastock import* from the menu
- Using the directory requester (***Browse***...) select the location of data in Metastock format (the directory with MASTER/EMASTER and Fxx.DAT files)
- After choosing proper directory AmiBroker will display the list of available symbols and date ranges. By default all available symbol will be marked for importing (checkmark at the beginning of the list). Now you can exclude some symbol from the import list by clicking appropriate item in the list (checkmark will toggle when you click).

- You can decide to which group and watch list the new symbols are added using **Group** and **Watch List** combos.
- After making your selections push '**Import**' button to start the process of importation.
- During the process you can cancel the operation by clicking '**Abort**' button in the progress window

Understanding AmiBroker database concepts

Background

A typical Windows application, for example, Paint, works with a SINGLE file. You just open and save that single file (.BMP in Paint, or .DOC in MS WORD), and that file holds all the necessary information.

AmiBroker is a more complex piece of software. It uses huge amounts of data (all quotes from different tickers, hand drawn studies, assignments to groups, markets, watch lists, favorites, industries, sectors, etc.), so it must manage multiple files.

It would actually be possible to save all this information in a single file, but it would be (a) huge, and (b) slow to update selectively. So AmiBroker uses multiple files for storing all the data. There are a lot of files associated with any database. The files for a particular database are stored in a directory (and its subdirectories) specific to that database. In AmiBroker documentation, such a directory is referred to either as a "database directory" (versions 3.9 or later) or as a "workspace directory" (earlier AmiBroker versions).

When you install AmiBroker for the first time, a default database directory is created, called 'data', in the AmiBroker directory. This database directory contains a sample Dow Jones Industrial Average database.

In AmiBroker database menu and dialog selections, you are choosing or creating a database directory, not an individual file.

AmiBroker database structure

A database (or a workspace) is a directory that holds a set of binary files, which are stored in 0-9, a-z, '_' subdirectories. Those binary files hold quotes, symbol information, your studies (trend lines, Fibonacci stuff). Each symbol's information is stored separately in the file with the name of the ticker symbol located in the subdirectory corresponding to the first character of the symbol, so IBM quotation data/studies are stored in the 'IBM' file located in the 'I' subdirectory.

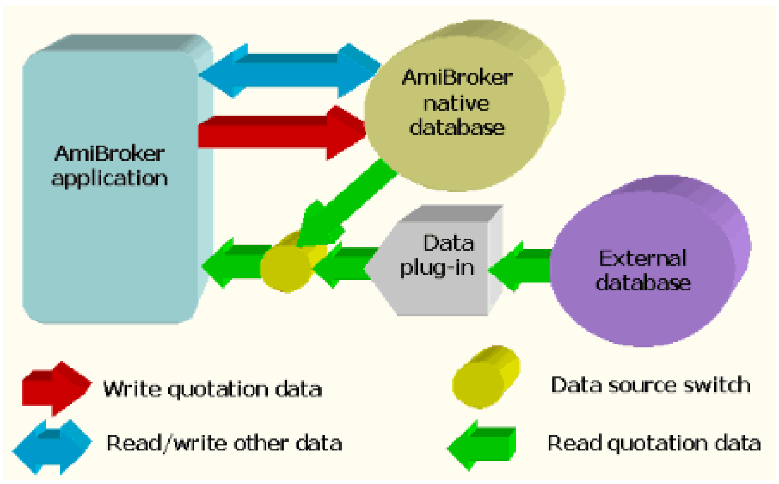
The default database for AB is the 'data' directory. It contains DJIA sample data. You may create additional databases in other directories via the File->'New database' menu.

In addition to these subdirectories and files, two additional files are also created by AmiBroker: broker.workspace and broker.master. The first is used to store category names and information about advancing/declining/unchanged issues. The latter stores the table of all symbols that is used for quick loading of the database. These two files are located in the root directory of each database, the 'data' directory, by default.

In almost all cases, you should NOT touch files in an AmiBroker database, as the program manages them automatically, and no user intervention is required.

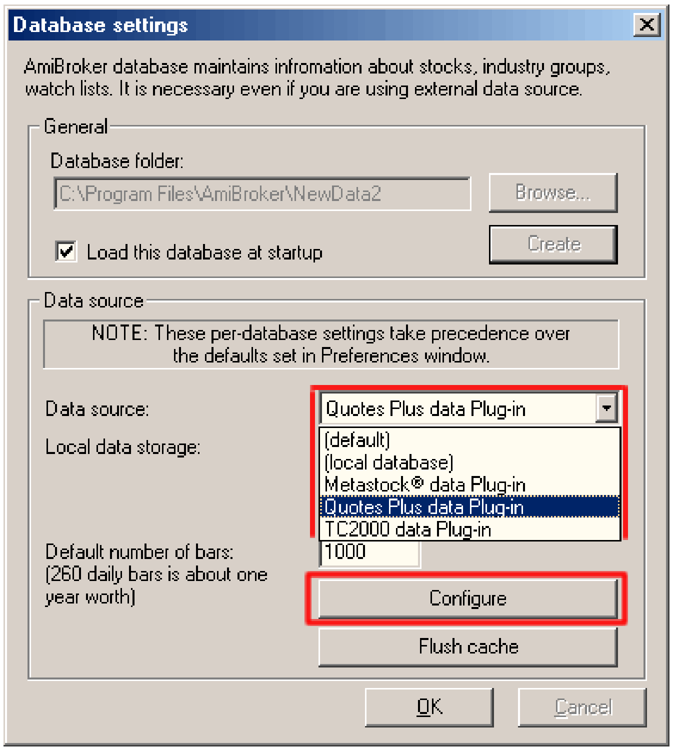
What about the external data?

AmiBroker 3.9 has the ability to read quotes DIRECTLY from an external data source. Currently, AmiBroker can read directly from Quotes Plus (QP2), TC2000 (TC2K) and Metastock (MS) databases. This is achieved by means of data plug-ins that AmiBroker uses to read the data from an external data source. When a user decides that she/he wants to use an external database - AmiBroker - instead of reading the quotes from its own database - just asks the plug-in for quotes for any given symbol. The plug-in reads the external database and feeds the data to AmiBroker. The whole process is shown in the picture below:



As you can see, data plug-ins provide **read-only** access to the quotes in the external database. This means that your external data sources are never modified by AmiBroker. Changes or additions that you make to data and charts (like hand drawn studies, assignments to categories, etc.) are always saved in AmiBroker's own database. **So AmiBroker still uses its own database (to save changes, as a cache to speed up access, and for other tasks), even when using an external data source.**

The Data source switch represented in the graphic above can be set by the user to access various external databases. External data sources are selected by going to the File->'Database settings' dialog, shown below:


Database settings

AmiBroker database maintains infromation about stocks, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: C:\Program Files\AmiBroker\NewData2

Load this database at startup [✓]

Data source

NOTE: These per-database settings take precedence over the defaults set in Preferences window.

Data source: Quotes Plus data Plug-in

Local data storage:

(default)
(local database)
Metastock® data Plug-in
Quotes Plus data Plug-in
TC2000 data Plug-in

Default number of bars: (260 daily bars is about one year worth) 1000

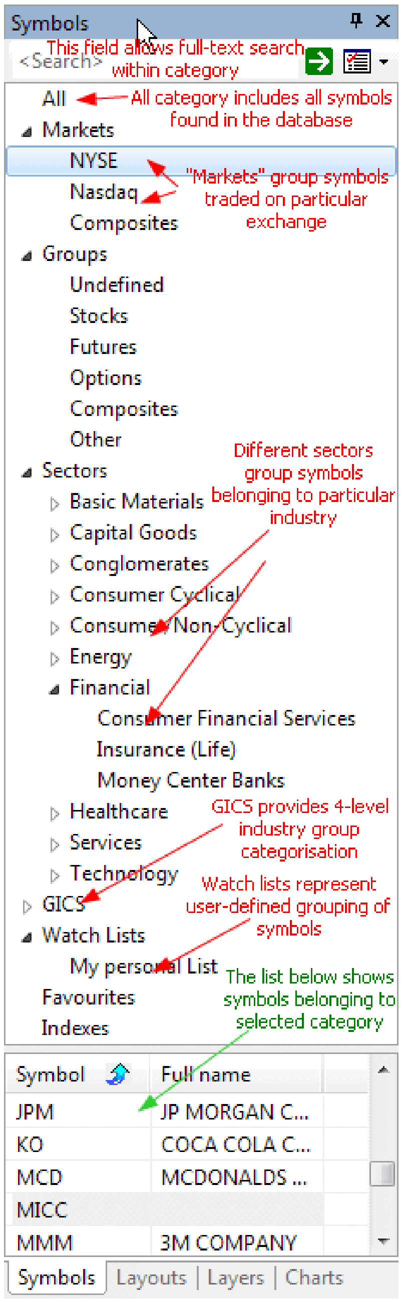
You may also choose to store the quotes retrieved from the external source to AmiBroker's own database for faster retrieval in subsequent accesses. If you want to do this, you should switch the 'Local data storage' setting to 'Enabled'.

Note: Similar settings can be found in the Tools->Preferences 'Data' tab, but these are only defaults used when creating new databases. **File->Database Settings configurations always take precedence over those done in Preferences -- EXCEPT in the following cases: If you choose the 'Default' entry in the Data Source drop down list (shown above), or the 'Default' radio button for Local Data Storage (also shown above), AmiBroker will use your Preferences settings for those items.**

Understanding categories

AmiBroker has an ability to assign symbols to different categories allowing you (when properly set up) to narrow your analysis searches to the symbols meeting certain selection criteria (thanks to Filter feature available in Quick Review and Automatic Analysis windows). The initial set up of categories may be a little bit complicated especially when you want to track several thousands symbols.

Categories show up in **Symbols** window. First and foremost thing to remember is categories do NOT work like folders and Workspace window does NOT work like Windows Explorer.



This field allows full-text search within category

All category includes all symbols found in the database

"Markets" group symbols traded on particular exchange

Different sectors group symbols belonging to particular industry

GICS provides 4-level industry group categorisation

Watch lists represent user-defined grouping of symbols

The list below shows symbols belonging to selected category

Symbol	Full name
JPM	JP MORGAN C...
KO	COCA COLA C...
MCD	MCDONALDS ...
MICC	
MMM	3M COMPANY

The difference is fundamental. In the Windows Explorer file appears (usually) only once in the given tree leaf.
In the symbol tree given symbol shows up multiple times because it appears in every category leaf to which it belongs to even if this is the same symbol and only it exists only as single entity.

Symbols window is divided into three parts:
a) search box
b) category tree
c) symbol list

The **search box** allows to perform full text searches (including wildcard matching) against symbol and full name within selected category. So for example if you select "Technology" sector and type A* (letter 'A' and wildcard character *) the symbol list will show all symbols belonging to Technology sector with symbol or full name beginning with letter 'A'. Another example would be tping *-A0-FX - this will return all forex symbols on eSignal database (those ending with -A0-FX substring).

The **category tree** (see the picture) shows different kind of categories.

The **symbol list** (bottom part) shows the list of symbols belonging to selected category. The symbol list can be sorted by symbol or by full name. To sort just click on the header row of the list. Once you choose desired sorting order it will be kept for all subsequent category choices and searches. Also the order of columns can be changed so Full name column appears as first one. To re-arrange column, click on the column header, hold down the moust button and drag the column to desired location. Then release mouse button.

Single symbol belongs to MANY categories at the same time. For example AAPL (Apple Inc.) will belong to:

- *Stocks* group category
- *Nasdaq* market category
- *Information* sector category
- *Comp-Computer Mfg* industry category

and may also belong to several watch lists and favorites category. All at the same time. That's why one symbol will appear in many leaves of the workspace symbol tree. Now if you delete the SYMBOL it will of course disappear from ALL categories because you have deleted the symbol itself, not its assignment to category.

There are two types of categories:

1. with mutualy exclusive membership: groups, markets, sectors/industries, GICS - it means that symbol must belong to single group, single market and single sector/industry at a time. You can move the symbol from one group/market/sector/industry to another but you can not remove this assignment - you should create "Unassigned" group/market/sector/industry instead and move 'unassigned' symbols there.
2. with free membership: watch lists/favorites/indexes- it means that a symbol may belong to ANY number (including zero) of watch lists (and to favorite/index category too). In this case you can remove this assignment by Watch List->Remove

   Watch lists are covered in detail in the User's Guide: Tutorial: Working with Watch Lists.

There is also one special category called "ALL" that shows up in the workspace symbol tree. It simply lists ALL symbols present in the database.

Working with sectors and industries

Basics - predefined sectors and industries

Now we will focus on setting up sectors and industries and assigning the symbols to them. First let me discuss some basic ideas.

AmiBroker comes with an example Dow Jones Industrials database holding all 30 components of this world's most famous market average. They are assigned to predefined sectors and industries. These sectors and industries are exactly the same as used on Yahoo finance site and here is a table which shows them all:

Sector	Industry
Basic Materials (0)	Chemical Manufacturing
	Chemicals - Plastics & Rubber
	Containers & Packaging
	Fabricated Plastic & Rubber
	Forestry & Wood Products
	Gold & Silver
	Iron & Steel
	Metal Mining
	Misc. Fabricated Products
	Non-Metallic Mining
	Paper & Paper Products

Capital Goods (1)	Aerospace & Defense
	Constr. - Supplies & Fixtures
	Constr. & Agric. Machinery
	Construction - Raw Materials
	Construction Services
	Misc. Capital Goods
	Mobile Homes & RVs
Conglomerates (2)	Conglomerates
Consumer Cyclical (3)	Apparel/Accessories
	Appliance & Tool
	Audio & Video Equipment
	Auto & Truck Manufacturers
	Auto & Truck Parts
	Footwear
	Furniture & Fixtures
	Jewelry & Silverware
	Photography
	Recreational Products
	Textiles - Non Apparel
	Tires
Consumer/Non-Cyclical (4)	Beverages (Alcoholic)
	Beverages (Non-Alcoholic)
	Crops
	Fish/Livestock
	Food Processing
	Office Supplies
	Personal & Household Prods.
	Tobacco
Energy (5)	Coal
	Oil & Gas - Integrated
	Oil & Gas Operations
	Oil Well Services & Equipment
Financial (6)	Consumer Financial Services
	Insurance (Accident & Health)
	Insurance (Life)
	Insurance (Miscellaneous)
	Insurance (Prop. & Casualty)

	Investment Services
	Misc. Financial Services
	Money Center Banks
	Regional Banks
	S&Ls/Savings Banks
Healthcare (7)	Biotechnology & Drugs
	Healthcare Facilities
	Major Drugs
	Medical Equipment & Supplies
Services (8)	Advertising
	Broadcasting & Cable TV
	Business Services
	Casinos & Gaming
	Communications Services
	Hotels & Motels
	Motion Pictures
	Personal Services
	Printing & Publishing
	Printing Services
	Real Estate Operations
	Recreational Activities
	Rental & Leasing
	Restaurants
	Retail (Apparel)
	Retail (Catalog & Mail Order)
	Retail (Department & Discount)
	Retail (Drugs)
	Retail (Grocery)
	Retail (Home Improvement)
	Retail (Specialty)
	Retail (Technology)
	Schools
	Security Systems & Services
	Waste Management Services
Technology (9)	Communications Equipment
	Computer Hardware
	Computer Networks

	Computer Peripherals
	Computer Services
	Computer Storage Devices
	Electronic Instruments & Controls
	Office Equipment
	Scientific & Technical Instr.
	Semiconductors
	Software & Programming
Transportation (10)	Air Courier
	Airline
	Misc. Transportation
	Railroads
	Trucking
	Water Transportation
Utilities (11)	Electric Utilities
	Natural Gas Utilities
	Water Utilities

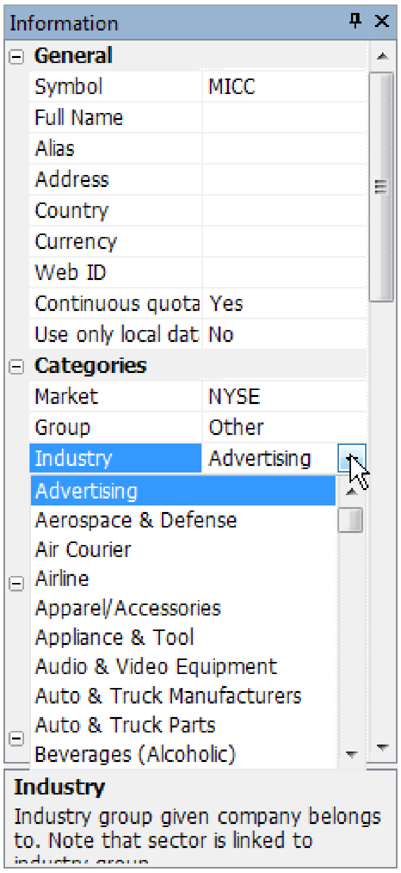
It is important to understand the difference between a sector and an industry: industries "belong" to sectors, for example: "Air Courier", "Airline", "Railroads", "Trucking" industries belong to "Transportation" sector. So if a symbol is assigned to given industry, it is "automatically" assigned also to the corresponding sector.

In the example DJIA database each stock is assigned to specific industry, for example GM (General Motors) is assigned to "Auto & Truck Manufacturers" industry, and this implicates that GM belongs to "Consumer/Cyclical" sector.

AmiBroker can handle up to 32 sectors and up to 256 industries.

How to assign symbol to the industry?

You can change the industry to which given symbol is assigned by using ***Window->Information*** dialog (Industry combo box)


Information

General

Symbol	MICC
Full Name	
Alias	
Address	
Country	
Currency	
Web ID	
Continuous quota	Yes
Use only local dat	No

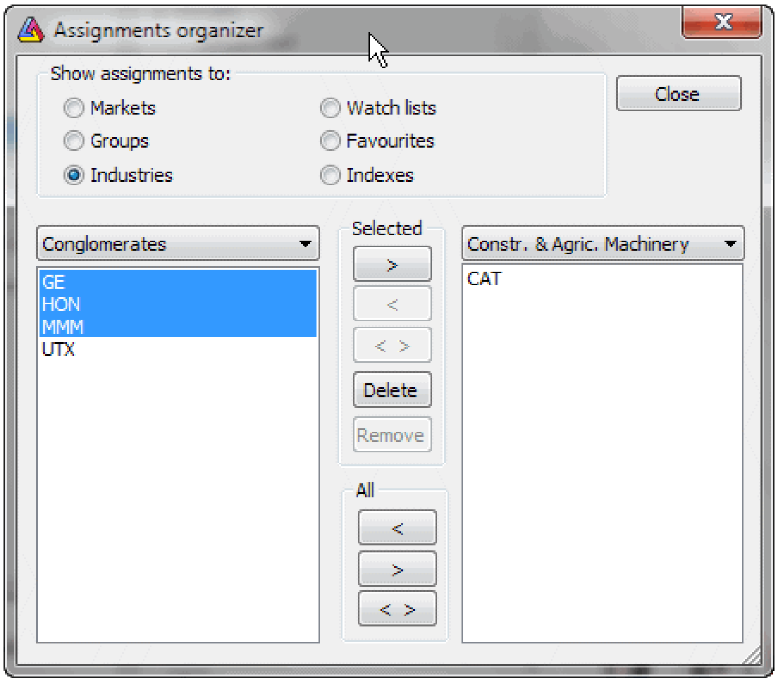
Categories

Market	NYSE
Group	Other
Industry	Advertising

Advertising
Aerospace & Defense
Air Courier
Airline
Apparel/Accessories
Appliance & Tool
Audio & Video Equipment
Auto & Truck Manufacturers
Auto & Truck Parts
Beverages (Alcoholic)

Industry
Industry group given company belongs to. Note that sector is linked to

or using ***Symbol->Organize Assignments***.


Assignments organizer

Show assignments to:
Markets Watch lists
Groups Favourites
[✓ Industries] Indexes

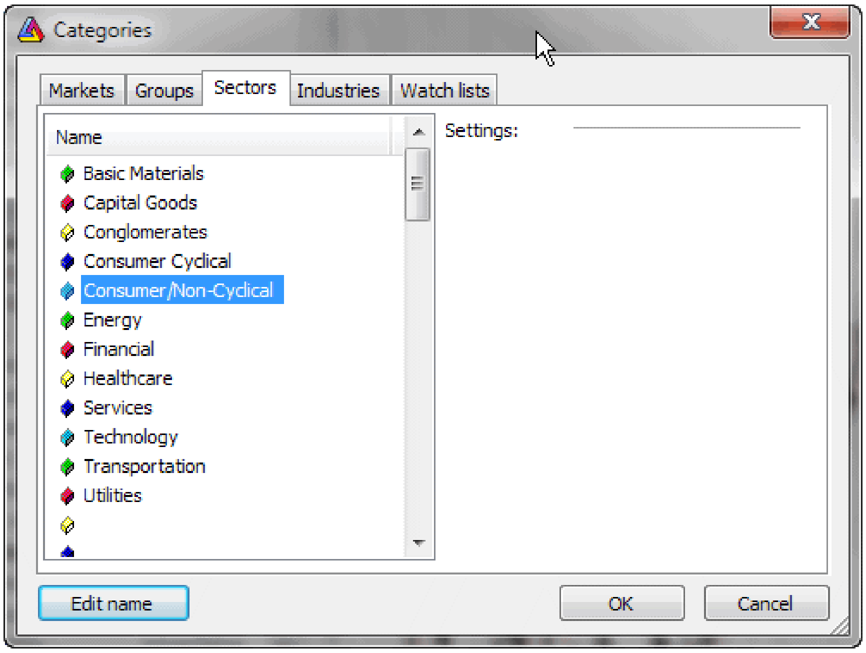
Conglomerates: GE, HON, MMM, UTX

Constr. & Agric. Machinery: CAT

The first method is fine if you want to change single symbol settings. The latter is better if you want to move multiple symbols from one category to another.

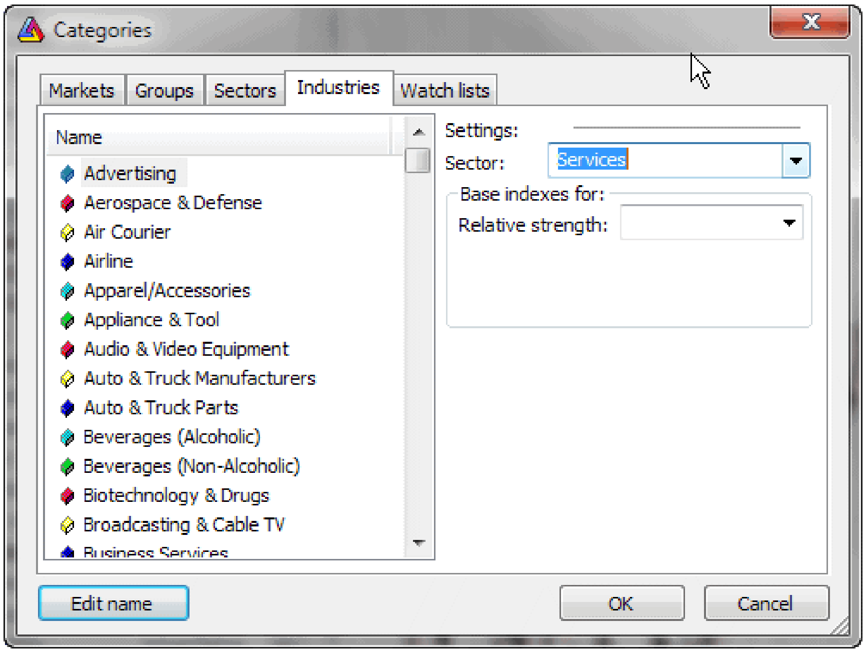
How to define your own sectors and industries

Please go to ***Symbol->Categories*** dialog, the last two tabs are "Sectors" and "Industries". First, switch to the "Sectors" tab and you will see the list of 32 sector names. You can now select the sector by clicking once on its name and edit the sector name by pressing ENTER or clicking "Edit name" button. Hit ENTER again to accept the name change.



Basic Materials
Capital Goods
Conglomerates
Consumer Cyclical
Consumer/Non-Cyclical
Energy
Financial
Healthcare
Services
Technology
Transportation
Utilities

After you renamed the sectors you can switch to the "Industries" tab. Similarly to the previous tab you can select the industry in the list and edit its name in the same manner. Here you can also assign the industry to the sector using "Sector" combo. Just select the sector to which you want to assign currently selected industry.



Sector: Services

Advertising
Aerospace & Defense
Air Courier
Airline
Apparel/Accessories
Appliance & Tool
Audio & Video Equipment
Auto & Truck Manufacturers
Auto & Truck Parts
Beverages (Alcoholic)
Beverages (Non-Alcoholic)
Biotechnology & Drugs
Broadcasting & Cable TV
Business Services

Where sector and industry information is stored?

Generally speaking this information is stored in AmiBroker database. The sector and industry names and settings are stored in the broker.workspace file (in the workspace folder), symbol data files hold only the information about the assignment of the symbol to given industry (IndustryID).

When you create a new workspace (a database) AmiBroker sets up your industries and sectors according to the templates stored in the "broker.sectors" and "broker.industries" files. These are simple text files that could be edited with plain text editor (such as Notepad). These files could be also used for quick, automatic setup of the sectors and industries. AmiBroker comes with predefined broker.sectors and broker.industries that follow described above convention (see the table). You can rewrite broker.sectors and broker.industries files to define your own default scheme. So, "broker.sectors" and "broker.industries" files are used as a template when creating new workspace. Once workspace is created these files are **not** taken into consideration. In this way you may have different categories in each workspace. If you want AmiBroker to load them into already existing workspace please delete broker.workspace file **before** opening the workspace. If you then open the workspace AmiBroker will read broker.sectors and broker.industries.

The layout of broker.sectors file is very simple: it is plain text file holding sector names written line by line as shown below:

```
Basic Materials
Capital Goods
Conglomerates
Consumer Cyclical
Consumer/Non-Cyclical
Energy
Financial
Healthcare
Services
Technology
Transportation
Utilities
```

The layout of broker.industries is similar, but in addition to industry names there is a number at the beginning of each line:

```
8 Advertising
1 Aerospace & Defense
10 Air Courier
10 Airline
3 Apparel/Accessories
3 Appliance & Tool
3 Audio & Video Equipment
3 Auto & Truck Manufacturers
3 Auto & Truck Parts
4 Beverages (Alcoholic)
4 Beverages (Non-Alcoholic)
7 Biotechnology & Drugs
8 Broadcasting & Cable TV
8 Business Services
8 Casinos & Gaming
0 Chemical Manufacturing
```

```
0 Chemicals - Plastics & Rubber
5 Coal
9 Communications Equipment
```

The numbers at front of industry names are "Sector IDs". Those numbers decide to which sector given industry belongs to. Because several industries may belong to one sector - you may need to put the same number for sector Id. Sector IDs are zero based, which means that 0 refers to the first line (sector name) of "broker.sectors" file, while 7 refers to the eighth line of this file. In the example above: "Advertising" industry belongs to "Services" sector, while "Aerospace & Defence" industry belongs to "Capital Goods" sector.

If you don't want to setup detailed industry information and want assign symbols only to sectors you can define one-to-one relationship between first 32 industries so they will be equivalent to sectors. Using the broker.sectors as show earlier in this article 1-1 broker.industries file would look like:

```
0 Basic Materials
1 Capital Goods
2 Conglomerates
3 Consumer Cyclical
4 Consumer/Non-Cyclical
5 Energy
6 Financial
7 Healthcare
8 Services
9 Technology
10 Transportation
11 Utilities
```

Note that this file is essentially the same as broker.sectors with the only difference that we have consecutive numbers prepended to each line. Using this kind of setup setting the industry will be equivalent to setting the sector.

Making it automatic

As described above symbol and industries names and relationship can be easily set up quickly using "broker.sectors" and "broker.industries" files. It will save some work needed otherwise to enter this information in ***Symbol->Categories*** window.

Unfortunately a lot more work is needed to assign all symbols to the industries even using ***Symbol->Organize Assignments*** dialog.

Fortunatelly there is a way to setup and update the database automatically.

In pre-5.60 version it still required scripting and lots of work (see 4th issue of AmiBroker Tips newsletter) but version 5.60 brings completely new ways to setup the database automatically.

The improved ASCII importer in v5.60 allows to import symbols, sectors and industry names and build complete database in just one step.

Let us say that we have CSV file that looks as follows:
"DDD","3D Systems Corporation","Technology","Computer Software: Prepackaged Software"
"MMM","3M Company","Health Care","Medical/Dental Instruments"
"SVN","7 Days Group Holdings Limited","Consumer Services","Hotels/Resorts"

```
"AHC","A.H. Belo Corporation","Consumer Services","Newspapers/Magazines"
"AIR","AAR Corp.","Capital Goods","Aerospace"
"AAN","Aaron's, Inc.","Technology","Diversified Commercial Services"
```

Now we can import it into AmiBroker and automatically setup all sectors and industries using this format definition

```
$FORMAT Ticker, FullName,SectorName,IndustryName
$SEPARATOR ,
$AUTOADD 1
$NOQUOTES 1
$OVERWRITE 1
$CLEANSECTORS 1
$SORTSECTORS 1
```

The last two commands ($CLEANSECTORS and $SORTSECTORS) instruct AmiBroker to clean (wipe) existing sector/industry names before importing and sort newly imported sectors after importing so they appear alphabetically

AmiBroker will read such ASCII file one-by one, then it will check whenever given sector name/industry name already exists, if not - it will create new sector/industry. Then it will assign given symbol to specified sector/industry.

The result will be a database with new sector/industry structure being set up and symbols assigned to proper sectors and industries.

Described functionality is used to implement Tools->Update US symbol list and categories tool.

One-click "Update US symbol list and categories"

Automatic setup and update of US stock database is available from **Tools->Update US symbol list and categories** menu. This is implemented using new #import command and new ASCII importer commands described above.

The command downloads symbol, sector and industry list from amibroker.com web site and create or update current database with stocks listed on NYSE, Nasdaq and AMEX. It also creates sector and industry structure and assigns stocks to proper industries.

CAVEAT: Be aware that using this tool will WIPE (delete) any existing sectors/industries and replace them with the ones imported automatically.

Note about GICS

GICS stands for Global Industry Classification Standard (http://en.wikipedia.org/wiki/Global_Industry_Classification_Standard).

AmiBroker allows also GICS 4-level classification system, but demo database does not have symbols classified according to that standard. You can find GICS classification codes in GICS.txt file inside AmiBroker folder.

Note about ICB

ICB stands for Industry Classification Benchmark
(http://en.wikipedia.org/wiki/Industry_Classification_Benchmark).

AmiBroker allows also ICB 4-level classification system, but demo database does not have symbols classified according to that standard. You can find ICB classification codes in ICB.txt file inside AmiBroker folder.

Working with watch lists

AmiBroker 5.00 uses now new watch list system. Watch lists differ from other kinds of categories (as groups, markets, industries, sectors) in that, that you can assign single symbol to more than one watch list.

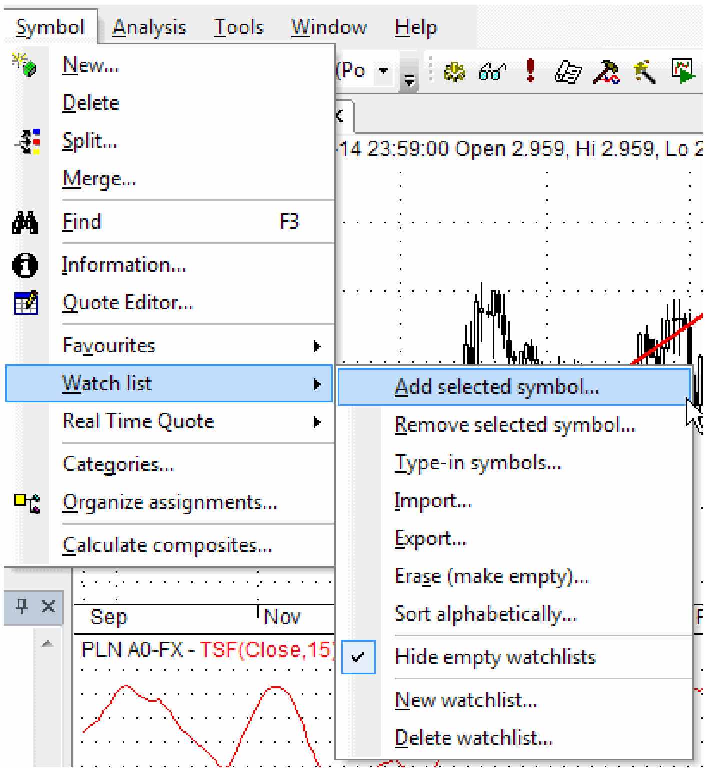
You can use UNLIMITED number of watch lists with their names definable in ***Symbol->Categories*** window. The members of each watch list is shown in the symbol tree under "Watch lists" leaf.

Watch lists are now stored as text files inside "Watchlists" folder inside database. The folder contains of any number of .TLS files with watch lists themselves and index.txt that defines the order of watch lists. You can add your own .tls file (one symbol per line) and AmiBroker will update index.txt automatically (adding any new watch lists at the end)The .TLS files can also be open in AmiQuote.

Watch lists remember the order in which symbols were added, so for example if you sort AA result list in some order and then you"add symbols to watch list" the order will be kept in the watch list.

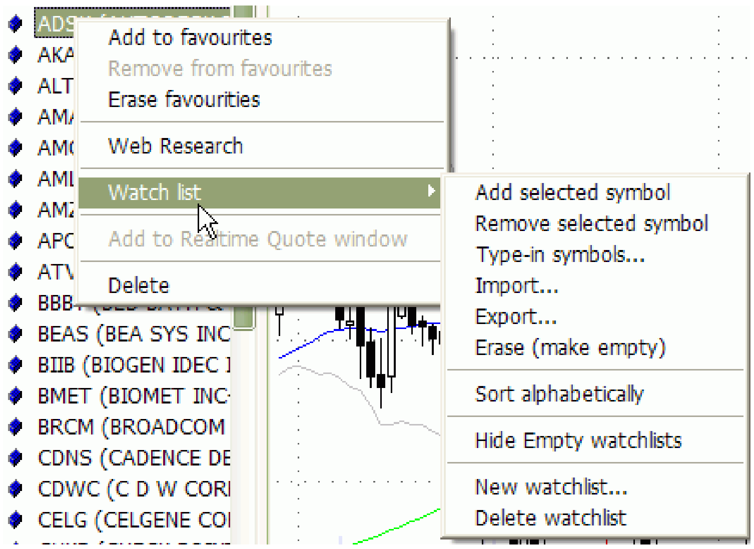
Adding / removing watch lists

You can now Add/Delete watch lists using **Symbol->Watch List->New Watchlist** and **Symbol->Watch List->Delete Watch list** menu or from watch list context menu. Note that if you have done any customization to the menu, you may need to go to Tools->Customize, select "Menu Bar" and press "Reset" button for this new menu items to appear.

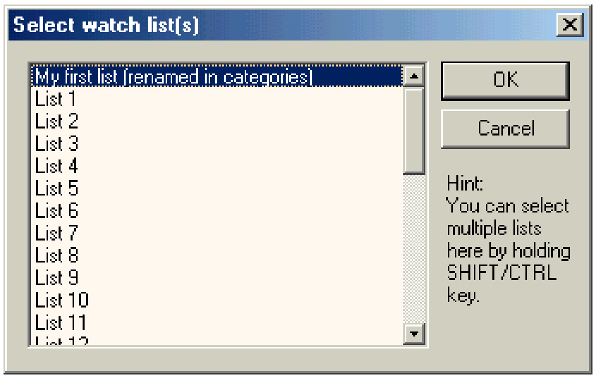


Adding tickers to watch lists

You can easily add a ticker to the watch list by simply clicking with a right mouse button over the item in the symbol tree and choosing ***Watch List->Add selected symbol*** option:

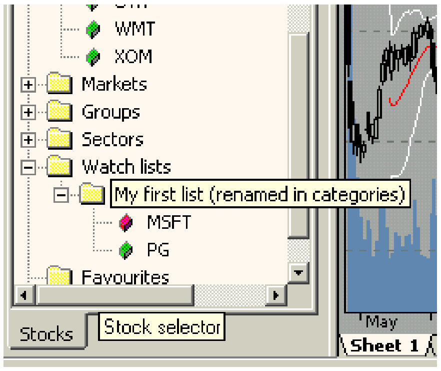


After choosing this option a watch list selector window will appear:



Hint:
You can select multiple lists here by holding SHIFT/CTRL key.

Here you should select the list you want to add the symbol to. Note that you can add one symbol to multiple lists at once, by holding CTRL key while clicking on the list items. After clicking OK selected symbol (MSFT) appears in the watch list of your choice:



You can also type-in symbols directly into the watch list using **Symbol->Watch list->Type-in option.** Symbols should be comma-separated. You can also right click over the watch list name in the workspace tree to type in symbols directly into the watch list.

Sorting tickers in a watch list

You can now alphabetically sort the symbols in the watch list - click on the watch list with RIGHT mouse button and select **"Sort Alphabetically"**

Removing tickers from watch lists

Removing symbols from the watch list is as easy as adding them. Just click on the list member with a right mouse button and select ***Remove from watch list(s)***. Then similar list selector window will appear showing only those lists that currently selected symbol belongs to. You can now select one or more lists and the symbol will be removed from the list(s).

Erasing watch lists

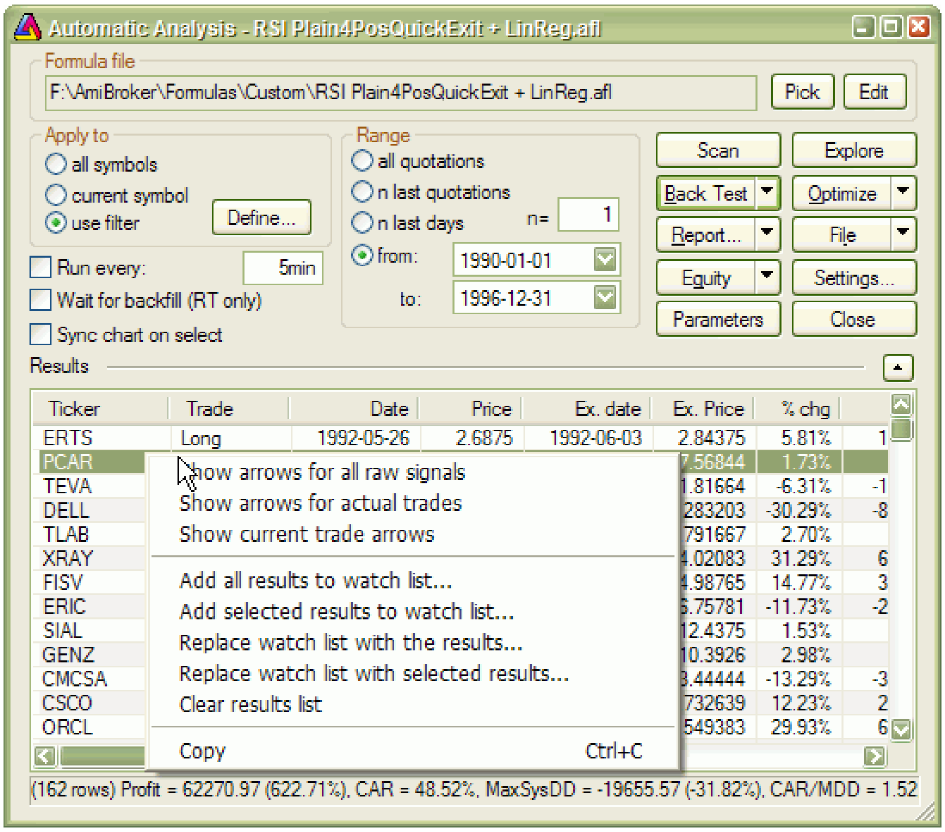
Sometimes you may want to clear (or erase) the whole watch list. Then just select ***Symbol->Watch list->Erase (empty)*** option. In the watch list selector window mark the list(s) you want to clear and click OK. This way selected watch list(s) become empty.

Hiding/Unhiding empty watch lists

By default empty watch lists are shown in the symbol tree but you can hide them by right-click on watch list in the symbol tree and select **"Hide Empty Watchlists"** menu. To un-hide, select this option again.

Using watch lists in Automatic analysis window

AmiBroker gives you a very easy way to store the results of scanning, backtesting and exploration into a watch list with a single mouse click - just run your favourite AFL formula over the whole database and click on the results list with a right mouse button to see the following menu:


Automatic Analysis - RSI Plain4PosQuickExit + LinReg.afl
Pick
Edit
Scan
Explore
Back Test
Optimize
Report...
File
Equity
Settings...
Parameters
Close
Define...
Results
Show arrows for all raw signals
Show arrows for actual trades
Show current trade arrows
Add all results to watch list...
Add selected results to watch list...
Replace watch list with the results...
Replace watch list with selected results...
Clear results list
Copy Ctrl+C

Formula file: F:\AmiBroker\Formulas\Custom\RSI Plain4PosQuickExit + LinReg.afl

Apply to: all symbols / current symbol / [✓ use filter]

Range: all quotations / n last quotations / n last days n= 1 / [✓ from:] 1990-01-01 to: 1996-12-31

Run every: 5min

Wait for backfill (RT only)

Sync chart on select

Ticker	Trade	Date	Price	Ex. date	Ex. Price	% chg	
ERTS	Long	1992-05-26	2.6875	1992-06-03	2.84375	5.81%	1
PCAR					7.56844	1.73%	
TEVA					1.81664	-6.31%	-1
DELL					283203	-30.29%	-8
TLAB					791667	2.70%	
XRAY					4.02083	31.29%	6
FISV					4.98765	14.77%	3
ERIC					5.75781	-11.73%	-2
SIAL					12.4375	1.53%	
GENZ					10.3926	2.98%	
CMCSA					3.44444	-13.29%	-3
CSCO					732639	12.23%	2
ORCL					549383	29.93%	6

(162 rows) Profit = 62270.97 (622.71%), CAR = 48.52%, MaxSysDD = -19655.57 (-31.82%), CAR/MDD = 1.52

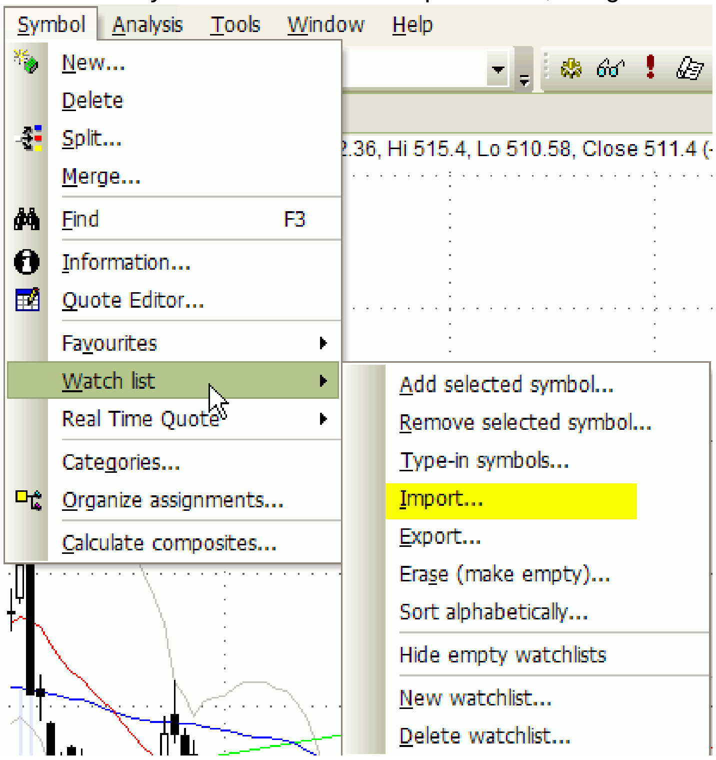
When you choose ***Add all/selected results to watch list*** a watch list selector will appear where you select to which list symbols should be added, then after clicking OK all symbols filtered by your trading rules will automatically appear in the watch list of your choice.

You can also use option **Replace watch list with the results/selected results**
This new option empties the watch list before adding results. The order of symbols in the result list is preserved in the watch list.

How to import/export watch list from/to file

IMPORT WATCH LIST FROM FILE

1. Choose Symbol->Watch List->Import menu, or right click over watch list in the tree and choose Import.



2. Choose destination watch list

3. In the file dialog that will appear pick .TLS, .LST, .TXT or .CSV file

.TLS, .CSV, .TXT files should have **one ticker symbol per line and no other fields**.
.LST files are Quotes-Plus standard, comma separated list files that have the ticker symbol in the first column and some additional data in remaining columns. AmiBroker reads just first column and ignores rest.

4. Click OK.

EXPORT WATCHLIST TO FILE

1. Choose Symbol->Watch List->Export menu.
or right click over watch list in the tree and choose Export.

2. Choose source watch list and switch to "External data source"

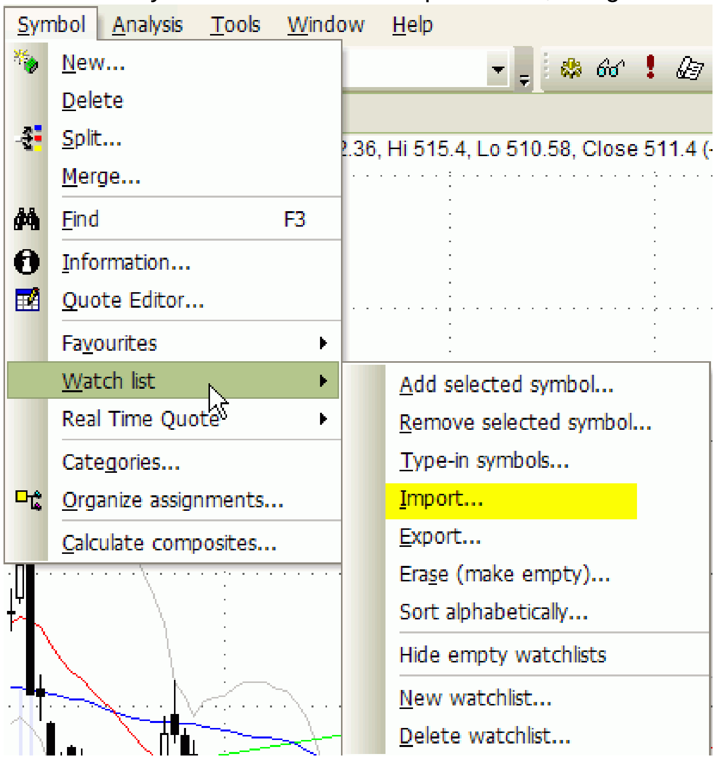
3. In the file dialog choose the file to export to. Generated file will be simple ASCII file witch one ticker symbol per line.

How to import/export watch list from/to external database

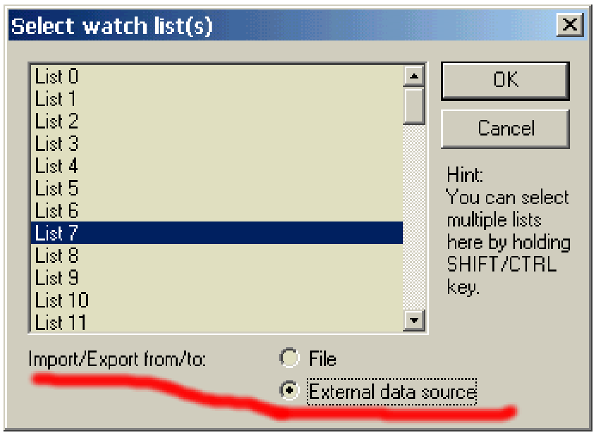
ATTENTION: It works ONLY if you have "Data source" set to "Fast Track" plugin in File->Database Settings (and if you have FastTrack database installed of course).

IMPORT FAMILY FROM FASTTRACK

1. Choose Symbol->Watch List->Import menu, or right click over watch list in the tree and choose Import.



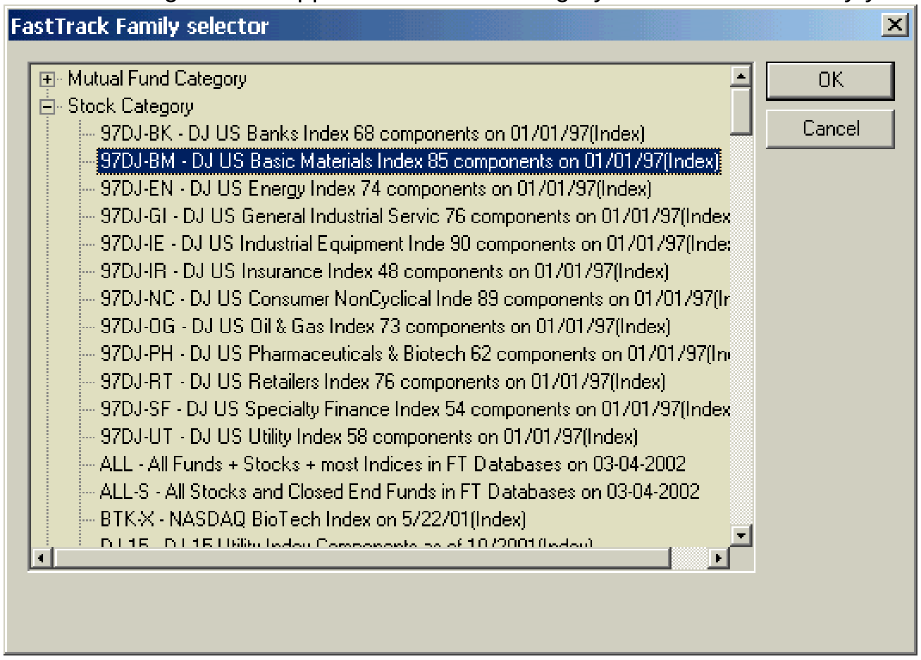
2. Choose destination watch list and switch to "External data source"



Hint:
You can select multiple lists here by holding SHIFT/CTRL key.

Import/Export from/to: File [✓ External data source]

3. In the dialog that will appear unfold one category and select the family you want to import symbols from:



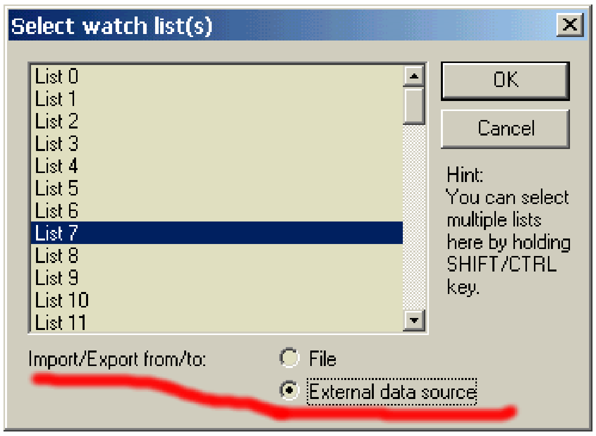
97DJ-BK - DJ US Banks Index 68 components on 01/01/97(Index)
97DJ-BM - DJ US Basic Materials Index 85 components on 01/01/97(Index)
97DJ-EN - DJ US Energy Index 74 components on 01/01/97(Index)
97DJ-GI - DJ US General Industrial Servic 76 components on 01/01/97(Index
97DJ-IE - DJ US Industrial Equipment Inde 90 components on 01/01/97(Inde
97DJ-IR - DJ US Insurance Index 48 components on 01/01/97(Index)
97DJ-NC - DJ US Consumer NonCyclical Inde 89 components on 01/01/97(Ir
97DJ-OG - DJ US Oil & Gas Index 73 components on 01/01/97(Index)
97DJ-PH - DJ US Pharmaceuticals & Biotech 62 components on 01/01/97(In
97DJ-RT - DJ US Retailers Index 76 components on 01/01/97(Index)
97DJ-SF - DJ US Specialty Finance Index 54 components on 01/01/97(Index
97DJ-UT - DJ US Utility Index 58 components on 01/01/97(Index)
ALL - All Funds + Stocks + most Indices in FT Databases on 03-04-2002
ALL-S - All Stocks and Closed End Funds in FT Databases on 03-04-2002
BTK-X - NASDAQ BioTech Index on 5/22/01(Index)

4. Click OK.

EXPORT WATCHLIST TO FASTTRACK FAMILY

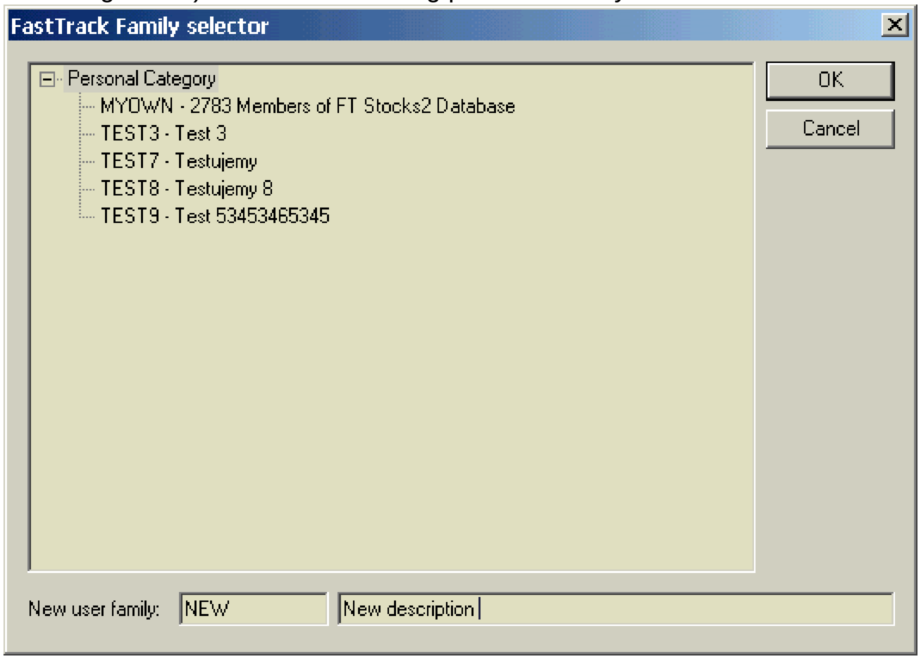
1. Choose Symbol->Watch List->Export menu.
or right click over watch list in the tree and choose Export.

2. Choose source watch list and switch to "External data source"



Import/Export from/to: File [✓ External data source]

3. Now either TYPE IN the new personal family name in "New user family" (and the description in the file next on the right side) OR choose existing personal family from the list.



New user family: NEW New description

Understanding how AFL works

Introduction

One of most important aspects of AFL is that it is an array processing language. It operates on arrays (or rows/vectors) of data. This way of operation is quite similar to the way how popular spreadsheets work (like Microsoft Excel). Anyone familiar with MS Excel should have no trouble quickly picking up AFL. - In fact all the examples in this article were all created using MS Excel.

What is an Array?

An array is simply a list (or row) of values. In some books it may be referred to as a vector. Each numbered row of values in the example represents an individual array. Amibroker has stored in its database 6 arrays for each symbol. One for opening price, one for the low price, one for the high price, one for the closing price and one for volume (see the rows labelled 1-5 below) and one for open interest. These can be referenced in AFL as open, low, high, close, volume, openint or o, l, h, c, v, oi.

All array indices in AFL are zero-based, i.e. counting bars starting from bar 0 (oldest). The latest (newest) bar has an index of (BarCount-1). The 10-element array will have BarCount = 10 and indices going from 0 to 9 as shown below

		(oldest)	----> time stamps increasing ---->								**(newest)**
	Bar	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	**Open**	1.23	1.24	1.21	1.26	1.24	1.29	1.33	1.32	1.35	1.37

Fig 1. Open price array

Any other array is calculated from these 6 arrays using formulae built into AFL. These arrays are not stored in the database but calculated where necessary.

Each individual value in an array has a date associated with it. If you have the tool tip option turned on (Preferences -> Miscellaneous Tab - > Price data tool tips), when you move your cursor over candle on a daily candle chart, a small yellow rectangle appears. AFL then looks up the open, low, high, close, volume values in the appropriate array and displays them inside the tool tip.

Processing arrays - why is AFL so fast?

Lets see how the following statement is processed:

MyVariable = (High + Low)/2;

When AFL is evaluating statement like this (High + Low)/2 it does not need to re-interpret this code for each bar. Instead it takes the High ARRAY and Low ARRAY and adds corresponding array elements in single stage. In other words + operator (and other operators too) work on arrays at once and it is executed at full compiled-code speed, then the resulting array (each element of it) is divided by 2 also in single stage.

Let's look into the details - see fig 2.. When AFL engine looks at the (High + Low)/2 it first takes High (1) and Low (2) arrays and produces (in single compiled step) the temporary array (3). Then it creates the final array (4) by dividing each element of temporary array by two. This result is assigned to myVariable

	Bar	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

1	**High** *(built-in array)*	1.24	1.27	1.25	1.29	1.25	1.29	1.35	1.35	1.37	1.29
2	**Low** *(built-in array)*	1.20	1.21	1.19	1.20	1.21	1.24	1.30	1.28	1.31	1.27
3	**High+Low** *(temporary array created during evaluation)*	2.44	2.48	2.44	2.49	2.46	2.53	2.65	2.63	2.68	2.46
4	**(High+Low) /2** *(gets assigned to MyVariable)*	1.22	1.24	1.22	1.245	1.23	1.265	1.325	1.315	1.34	1.23

Fig 2. AFL steps when processing (High + Low) /2

Moving averages, conditional statements

Let us now consider the following code:

```
Cond1 = Close > MA( Close, 3 );
Cond2 = Volume > Ref( Volume, -1 );
Buy = Cond1 AND Cond2;
Sell = High > 1.30;
```

This code generates a buy signal when todays close is higher than 3 day moving average of close AND todays volume is higher than yesterday's volume. It also generates a sell signal when today's high is higher than 1.30.

If in your AFL code you need to see if the closing price is greater than say a 3 day simple moving average AFL will first run through the close array creating a new array called MA(close,3) for the symbol being analysed. Each cell in the new array can then be compared one for one in the close array. In the example an array called Cond1 is created this way. For each cell where the closing price is greater than the corresponding cell value in MA(close,3) the cell value for new array 'Cond1' is set to '1'. If the closing price is not greater than the corresponding price in the close array the value in 'Cond1' is set to '0'.

AFL can also look forwards or backwards a number of cells in an array using the **Ref** function (see row 6 where temporary array is created holding previous day volume)

In row 9 a new array called Cond2 has been created by comparing the value of each cell in the volume array with its previous cell setting the Cond2 cell value to '1' if true and '0' if false.

Row 10 shows an array called 'Buy' created by comparing the cell values in Cond1 with the cell values in Cond2. If the cell in Cond1 has a '1' AND so does the corresponding cell in Cond2 then a '1' is placed in the 'Buy' array cell.

Row 11 shows an array called 'Sell' created whenever the cell value in the close array is greater than $1.30.

	Bar	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	**Open**	1.23	1.24	1.21	1.26	1.24	1.29	1.33	1.32	1.35	1.37
2	**High**	1.24	1.27	1.25	1.29	1.25	1.29	1.35	1.35	1.37	1.29
3	**Low**	1.20	1.21	1.19	1.20	1.21	1.24	1.30	1.28	1.31	1.27
4	**Close**	1.23	1.26	1.24	1.28	1.25	1.25	1.31	1.30	1.32	1.28
5	**Volume**	8310	3021	5325	2834	1432	5666	7847	555	6749	3456
6	**Ref(Volume, -1)** *(temporary array created during eval)*	Null	8310	3021	5325	2834	1432	5666	7847	555	6749
7		Null	Null	1.243	1.260	1.257	1.260	1.270	1.287	1.310	1.300

	MA(Close, 3) *(temporary array created during eval)*										
8	**Cond1 = Close < MA(close,3)** *(gives 1 (or true) if condition met, zero otherwise)*	Null	Null	1	0	1	1	0	0	0	1
9	**Cond2 = Volume > Ref(volume,-1)**	Null	0	1	0	0	1	1	0	1	0
10	**Buy = Cond1 AND Cond2**	Null	Null	1	0	0	1	0	0	0	0
11	**Sell = High > 1.30**	0	0	0	0	0	0	1	1	1	0

Obviously Buy and Sell are special arrays whose results can be displayed in the Analyser window or on screen using a red or green value as needed.

Getting little bit more complex

The examples above were very simple. Now I will just explain 3 things that seem to generate some confusion among the users:

- referencing selected values (SelectedValue, BeginValue, EndValue, LastValue)
- IIF function
- AMA function

As written in the Tutorial: Basic charting guide you can select any quote from the chart and you can mark From-To range. The bar selected by verticall line is called "selected" bar while start and end bars of the range are called "begin" and "end" bars. AFL has special functions that allow to reference value of the array at selected, begin and end bar respectively. These functions are called SelectedValue, BeginValue and EndValue. There is one more function called LastValue that allows to get the value of the array at the very last bar. These four functions take the array element at given bar and return SINGLE NUMBER representing the value of the array at given point. This allows to calculate some statistics regarding selected points. For example:

```
    EndValue( Close ) - BeginValue( Close )
```

Will give you dollar change between close prices in selected from-to range.

When number retrieved by any of these functions is compared to an array or any other arithmetic operation involving number and the array is performed it works like the number spanned all array elements. This is illustrated in the table below (rows 2, 6, 7). Green color marks "begin" bar and red color marks "end" bar. Selected bar is marked with blue.

	Bar	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	**Open**	1.23	1.24	1.21	1.26	1.24	1.29	1.33	1.32	1.35	1.37
2	**BeginValue(Open)**	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24	1.24
3	**EndValue(Open)**	1.32	1.32	1.32	1.32	1.32	1.32	1.32	1.32	1.32	1.32
4	**SelectedValue(Open)**	1.21	1.21	1.21	1.21	1.21	1.21	1.21	1.21	1.21	1.21
5	**LastValue(Open)**	1.37	1.37	1.37	1.37	1.37	1.37	1.37	1.37	1.37	1.37
6	**Close**	1.22	1.26	**1.23**	1.28	1.25	1.25	1.31	1.30	1.32	1.28
7	**Close <= BeginValue(Open)**	1	0	1	0	0	0	0	0	0	0

8	**result = IIF(Close <= BeginValue(Open), Close, Open);**	1.22	1.24	1.23	1.26	1.24	1.29	1.33	1.32	1.35	1.37
9	**Period**	2	3	4	2	3	5	2	3	4	2
10	**Factor = 2/(Period+1)**	0.667	0.500	**0.400**	0.667	0.500	0.333	0.667	0.500	0.400	0.667
11	**1 - Factor**	0.333	0.500	0.600	0.333	0.500	0.667	0.333	0.500	0.600	0.333
12	**AMA(Close, Factor)**	0.8125	1.0363	**1.1138**	1.2234	1.2367	1.2399	1.2853	1.2927	1.3036	1.2866

Now the IIF(condition, truepart, falsepart) function. It works that it returns the value of second (*truepart*) or third (*falsepart*) argument depending on *condition*. As you can see in the table above in row 8 the values come from Close array (*truepart*) for bars when condition is true (1) and come from Open array (*falsepart*) for the remaining bars. In that case the array returned by IIF function consists of some values from Close and some values from Open array. Note that both *truepart* and *falsepart* are arrays and they are evaluated regardless of the condition (so this is not a regular IF-THEN-ELSE statement but **function** that returns array)

The AMA(array, factor) function seems to cause the most problems with understanding it. But in fact it is very simple. It works in recursive way. It means that it uses its previous value for the calculation of current value. It processes array bar by bar, with each step it multiplies given cell of first argument (array) by given cell of second argument (factor) and adds it to the previous value of AMA multiplied by (1-factor). Lets consider bar number 2 (marked with blue). The value of AMA on that bar is given by multipling close price from that bar (1.23) by factor (0.4). Than we add the previous value of AMA (1.0363) multiplied by (1-factor = 0.6). The result (rounded to 4 places) is 1.23 * 0.4 + 1.0363 * 0.6 = **1.1138**.

If you look at the figures in the row 12 you may notice that these values look like a moving average of close. And that's true. We actually presented how to calculate variable-period exponential moving average using AMA function.

New looping

With version 4.40 AmiBroker brings ability to iterate through quotes using *for* and *while* loops and adds *if-else* flow control statement. These enhancements make it possible to work BOTH ways: either use ARRAY processing (described above) for speed and simplicity or use LOOPS for doing complex things. As an example how to implement variable period exponential averaging (described above) using looping see the following code:

```
Period = ... some calculation

vaexp[ 0 ] = Close[ 0 ]; // initialize first value

for( i = 1; i < BarCount; i++ )
{
  // calculate the value of smoothing factor
  Factor = 2/(Period[ i ] + 1 );

  // calculate the value of i-th element of array
  // using this bar close ( close[ i ] ) and previous average value ( vaexp[ i -
1 ] )
  vaexp[ i ] = Factor * Close[ i ] + ( 1 - Factor ) * vaexp[ i - 1 ];
}
```

As you can see the code is longer but on the other hand it is very similar to any other programming language as C/Pascal/Basic. So people with some experience with programming may find it easier to grasp.

If you are beginner I suggest to learn array processing first before digging into more complex looping stuff.

If you're having trouble coding AFL I suggest you generate the arrays in the example in Excel for yourself. If that's a problem get some help from a friend - especially if that friend is an accountant.

Once you've got the hang of it you can code any system from a book on trading - or build one yourself.

--- *Special thanks to Geoff Mulhall for original article in the newsletter that was the basis of this tutorial* ---

Creating your own indicators

There are two ways to create your own indicators:

1) using drag-and-drop interface

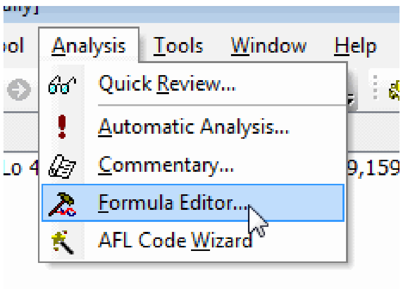
2) by writing your own formula

First method, using drag-and-drop interface is very simple and does not require writing single line of code. To learn more about drag-and-drop indicator creation please check Tutorial: How to use drag-and-drop charting interface

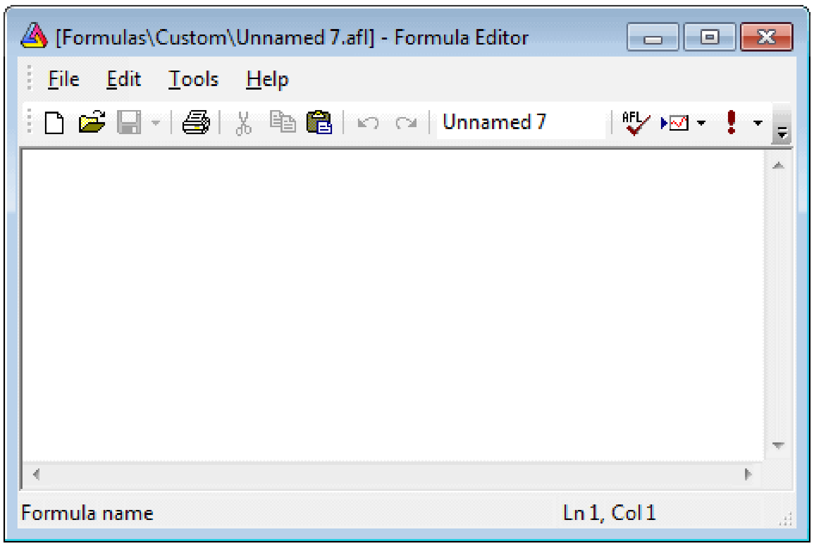
Second method involves writing an indicator formula in flexible AFL (AmiBroker Formula Language). You can find the description of this language in AFL Reference Guide section of user's guide. Here we will present basic steps needed to define and display your own custom indicator. In this example we will define an "indicator" that will show line volume graph (opposite to built-in bar volume graph).

Just follow these steps

1. Select *Analysis->Formula Editor* option from the menu as shown below:

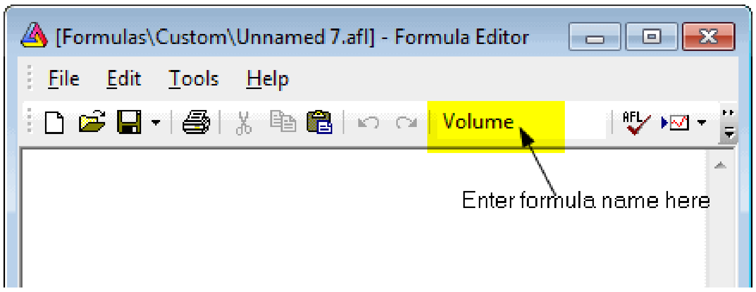


2. You will see the following dialog displayed on the screen:

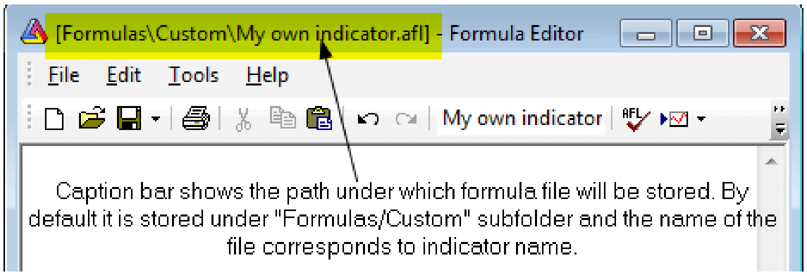


It presents an empty Formula Editor window.

3. Now single-click in the edit field located in the editor toolbar to change the name of the indicator:



Now you can edit the name of the custom indicator. Give it the name "My own indicator". After you press ENTER key the caption will be updated with the new file name as shown below:

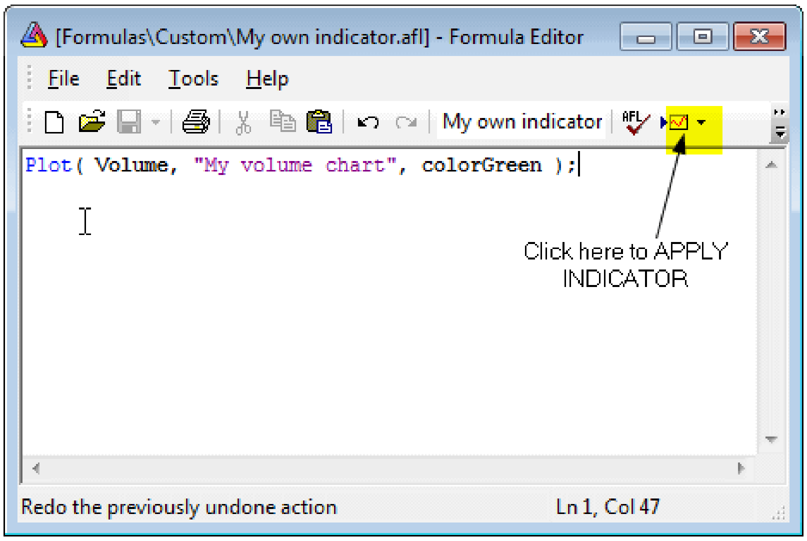


Caption bar shows the path under which formula file will be stored. By default it is stored under "Formulas/Custom" subfolder and the name of the file corresponds to indicator name.

4. Now type the formula:

```
Plot( Volume, "My volume chart", colorGreen );
```

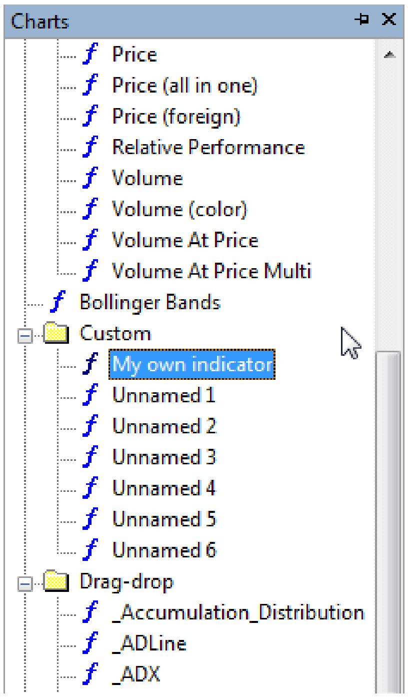
This formula instructs AmiBroker to plot built-in Volume array. Second parameter specifies the title of the plot and third parameter defines the color. The picture below shows formula editor after entering the code:


[Formulas\Custom\My own indicator.afl] - Formula Editor
File Edit Tools Help
My own indicator
Click here to APPLY INDICATOR
Redo the previously undone action
Ln 1, Col 47

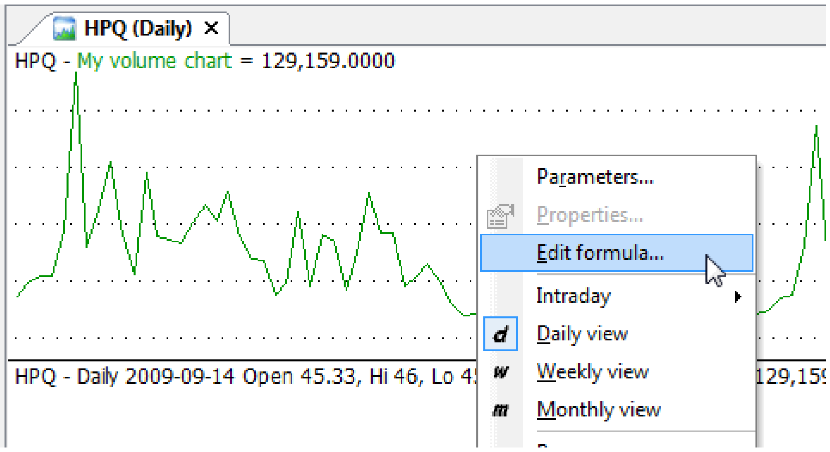
```
Plot( Volume, "My volume chart", colorGreen );
```

5. Now click **Apply indicator** toolbar button (or choose **Tools->Apply indicator** menu) as shown in the picture and close editor by pressing **X** button in the upper right corner of the editor window.

Now the indicator you have just written is displayed as a chart. You can also find it stored as a formula in Chart tree:



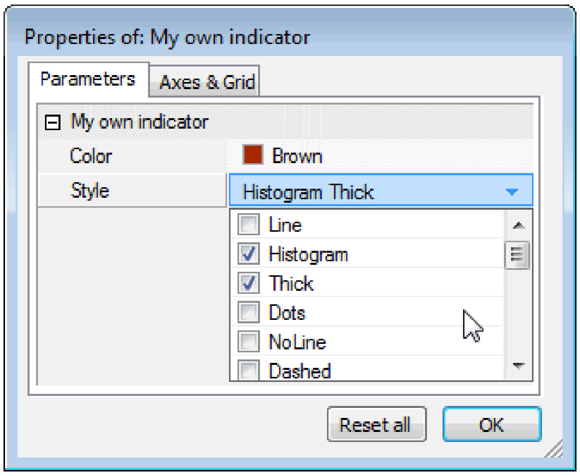
Now you can improve your indicator by adding Param functions so both color and style of the plot can be modified using Parameters dialog. To do so, click with RIGHT mouse button over chart pane and select **Edit Formula** (or press Ctrl+E)



And modify the formula to:

```
Plot( Volume, "My volume chart", ParamColor("Color", colorGreen ),
ParamStyle("Style", 0, maskAll ) );
```

Then press **Apply indicator** to apply the changes. Now click with RIGHT mouse button over chart pane again and select **Parameters** (or press Ctrl+R) and you will see parameters dialog allowing to modify colors and styles used to plot a chart:

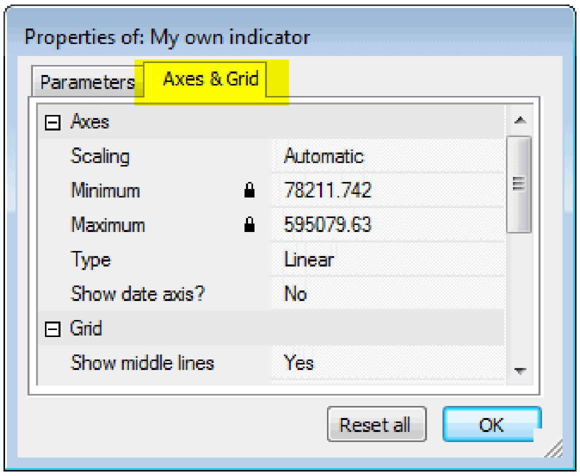


My own indicator

Color	🟥 Brown
Style	Histogram Thick

- [ ] Line
- [✓] Histogram
- [✓] Thick
- [ ] Dots
- [ ] NoLine
- [ ] Dashed

Also in the "Axes & Grid" tab you will be able to change settings for axes, grids and other charting options referring to this particular chart:



Axes

Scaling	Automatic
Minimum	78211.742
Maximum	595079.63
Type	Linear
Show date axis?	No

Grid

Show middle lines	Yes

For further information on creating your indicators please check Using graph styles and colors tutorial section

For further reference on using Formula Editor please consult *Environment - Formula Editor* and *AmiBroker Formula Language - AFL Tools* sections of AmiBroker User's guide and using AFL editor.

Using graph styles, colors, titles and parameters in Indicators

AmiBroker provides customizable styles and colors of graphs in custom indicators. These features allow more flexibility in designing your indicators. This article will explain how to use styles and colors. It will also explain how to define chart title that appears at the top of the chart.

Plot() function

Plot is the function used to plot a chart. It takes 9 parameters, out of which first 3 are required.

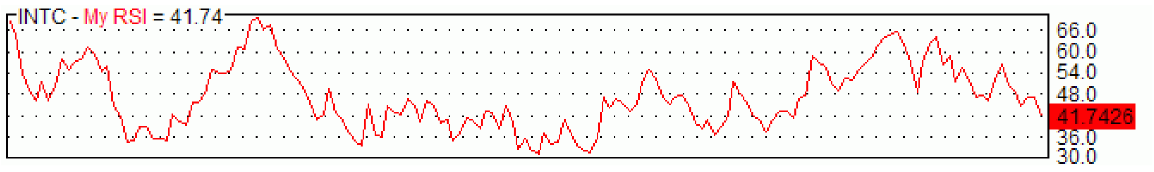
Plot(*array*, *name*, *color*, *style* = styleLine, *minvalue* = Null, *maxvalue* = Null, *XShift* = 0, *ZOrder* = 0, *width* = 1)

- *array* parameter represents data to be plotted,
- *name* parameter defines the name of the graph (used in title string to show the values of the indicator),
- *color* parameter defines the color of plot,
- *style* defines "the look" of the chart (i.e. line/histogram/candlestick/bar, etc). Default style is line.
- *minvalue* and *maxvalue* are rarely used paremeters that define hard-coded minimum and maximum values used when graph uses "independent" scaling, i.e. styleOwnScale is specified in *style* parameter. Usually you don't need to specify them at all.
- *XShift* allows shifting chart past the last bar (for example displaced moving averages or projections into the future)
- *ZOrder* - defines the Z-axis position of given plot. The default is zero. Zorder = 0 means also where the "grid" is located. So if you want to plot BEHIND the grid you need to specify negative zorder parameter.Plots are drawn in the following order:
  zorder parameter takes precedence over the order of calling Plot() functions, so if z-order is set, it determines plotting order. See this picture. If multiple plots use the same z-order parameter they are plotted in reverse call order (ones that appear last in the code are plotted first). This rule can be changed by already existing switch graphzorder = 1 which, when specified, reverses this behaviour (so plots are drawn in call order). Please note the above applies to each zorder "layer" separately (so within same zorder "layer" reverse call rule applies)
- *width* - (new in 5.60) defines pixel or percent width of given plot. The default is 1 pixel. Positive values specify PIXEL width, negative values specify width in percent of current bar width. So for example -20 will give you dynamic width that is 20% of bar width.

An example, the following single function call plots a RSI indicator with red color line:

```
Plot( RSI(14), "My RSI", colorRed );
```

As you can see we have provided only first three (required) parameters. First parameter is the array we need to plot. In our example it is RSI(14) indicator. Second parameter is just the name. It can be any name you want. It will be displayed in the title line along with indicator value as shown in the picture below:



Third parameter is the color. To specify plot color you can use one of the following pre-defined constants:

Color constants

Custom colors refer to color user-defined palette editable using Tools->Preferences->Colors, the numerical values that appear after = (equation) mark are for reference only and you don't need to use them. Use just the name such as colorDarkGreen.

colorCustom1 = 0
colorCustom2 = 1
colorCustom3 = 2
colorCustom4 = 3
colorCustom5 = 4
colorCustom6 = 5
colorCustom7 = 6
colorCustom8 = 7
colorCustom9 = 8
colorCustom10 = 9
colorCustom11 = 10
colorCustom12 = 11
colorCustom13 = 12
colorCustom14 = 13
colorCustom15 = 14
colorCustom16 = 15

colorBlack = 16
colorBrown = 17
colorDarkOliveGreen = 18
colorDarkGreen = 19
colorDarkTeal = 20
colorDarkBlue = 21
colorIndigo = 22
colorDarkGrey = 23

colorDarkRed = 24
colorOrange = 25
colorDarkYellow = 26
colorGreen = 27
colorTeal = 28
colorBlue = 29
colorBlueGrey = 30
colorGrey40 = 31

colorRed = 32
colorLightOrange = 33
colorLime = 34
colorSeaGreen = 35
colorAqua = 35
colorLightBlue = 37
colorViolet = 38
colorGrey50 = 39

colorPink = 40
colorGold = 41

colorYellow = 42
colorBrightGreen = 43
colorTurquoise = 44
colorSkyblue = 45
colorPlum = 46
colorLightGrey = 47

colorRose = 48
colorTan = 49
colorLightYellow = 50
colorPaleGreen = 51
colorPaleTurquoise = 52
colorPaleBlue = 53
colorLavender = 54
colorWhite = 55

You can also use new 24-bit (full color palette) functions ColorRGB and ColorHSB

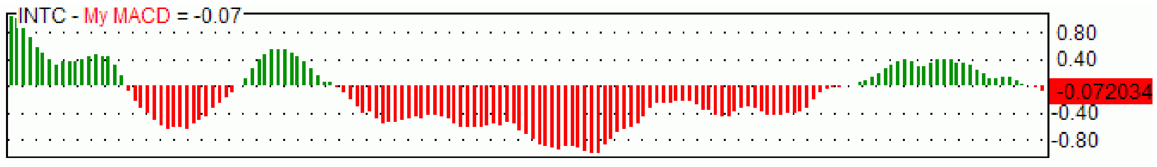
You can easily plot multi colored charts using both Plot functions. All you need to do is to define array of color indexes.
In the following example MACD is plotted with green color when it is above zero and with red color when it is below zero.

```
dynamic_color = IIf( MACD() > 0, colorGreen, colorRed );
Plot( MACD(), "My MACD", dynamic_color  );
```

In addition to defining the color we can supply 4th parameter that defines style of plot. For example we can change previous MACD plot to thick histogram instead of line:

```
dynamic_color = IIf( MACD() > 0, colorGreen, colorRed );
Plot( MACD(), "My MACD", dynamic_color, styleHistogram |
styleThick  );
```

As you can see, multiple styles can be combined together using | (binary-or) operator. (Note: the | character can be typed by pressing backslash key '\' while holding down SHIFT key). Resulting chart looks like this:



To plot candlestick chart we are using styleCandle constant, as in this example:

```
Plot( Close, "Price", colorBlack, styleCandle );
```

To plot traditional bars with color (green up bars and red down bars) we just specify color depending on relationship between open and close price and styleBar in *style* argument:

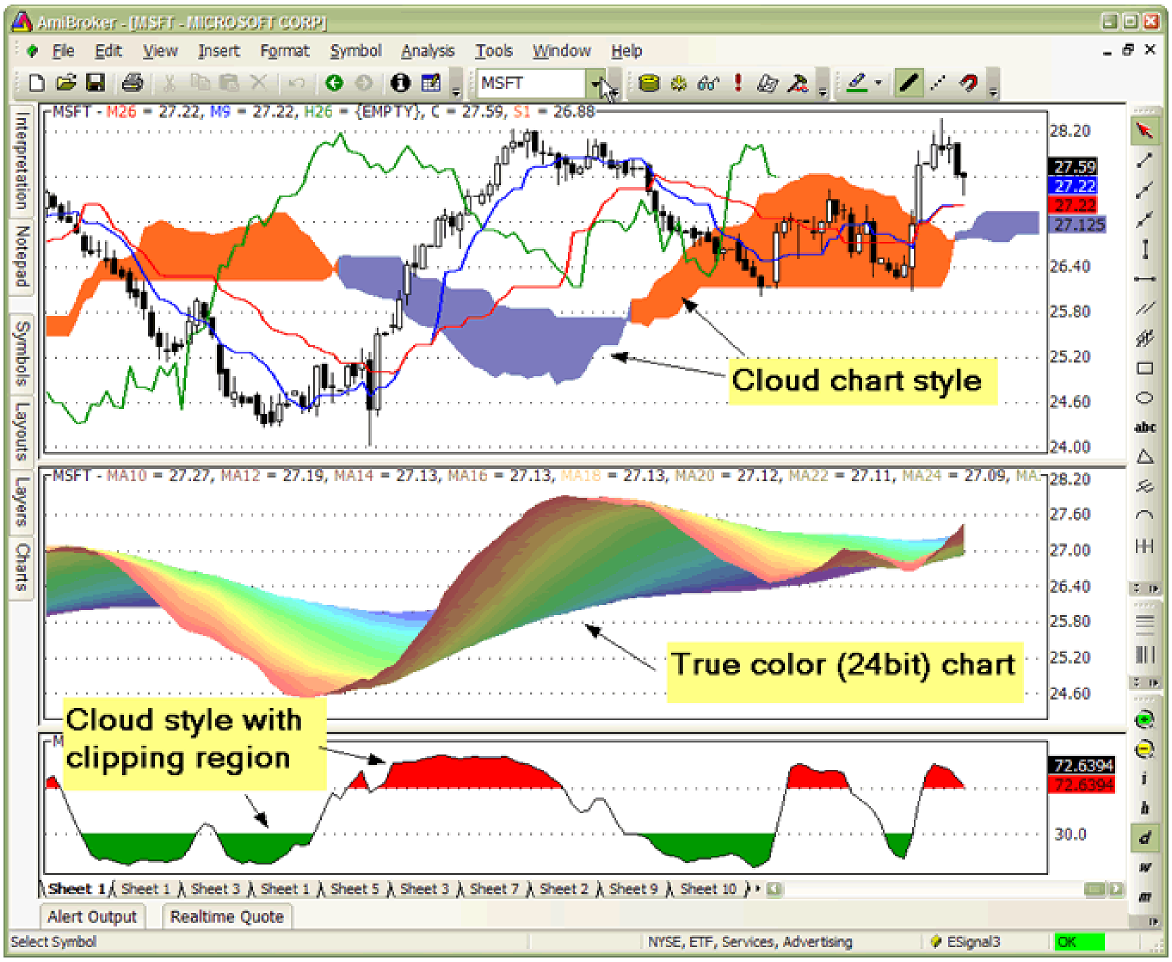
```
Plot( Close, "Price", IIf( Close > Open, colorGreen, colorRed ),
styleBar | styleThick );
```

All available style constants are summarized in the table below.

Style constants

Style is defined as a combination (using either addition (+) or binary-or (|) operator) of one or more following flags (you can use predefined style__ constants instead of numbers)
 	
styleLine = 1 - normal (line) chart (default)
styleHistogram = 2 - histogram chart
styleThick =4 - fat (thick)
styleDots = 8 - include dots
styleNoLine = 16 - no line
styleDashed = 32 - dashed line style
styleCandle = 64 - candlestick chart
styleBar = 128 - traditional bar chart
styleNoDraw = 256 - no draw (perform axis scaling only)
styleStaircase = 512 - staircase (square) chart
styleSwingDots = 1024 - middle dots for staircase chart
styleNoRescale = 2048 - no rescale
styleNoLabel = 4096 - no value label
stylePointAndFigure = 8192 - point and figure
styleArea = 16384 - area chart (extra wide histogram)
styleOwnScale = 32768 - plot is using independent scaling
styleLeftAxisScale = 65536 - plot is using left axis scale (independent from right axis)
styleNoTitle = 131072 - do not include this plot value in title string
styleCloud = 262144 - paint a "cloud" (filled area) chart (see examples below)
styleClipMinMax = 524288 - clip area between Min and Max levels defined in Plot statement. (Note: this style is not compatible with most printers)
styleGradient - (new in 5.60) - gradient area chart. Upper gradient color is specified by color parameter in Plot() function, bottom gradient color is either background color or can be defined using SetGradientFill function. styleGradient can be combined with styleLine
 	
Not all flag combinations make sense, for example (64+1) (candlestick + line) will result in candlestick chart (style=64)
 	
Note on candlestick/bar charts: if these styles are applied to Plot() function then they use indirectly O, H, L arrays.
If you want to specify your own OHL values you need to use PlotOHLC() function.

New styleCloud and styleClipMinMax styles bring new interesting possibilities shown in the sample image below:


AmiBroker - [MSFT - MICROSOFT CORP]
File Edit View Insert Format Symbol Analysis Tools Window Help
MSFT
MSFT - M26 = 27.22, M9 = 27.22, H26 = {EMPTY}, C = 27.59, S1 = 26.88
Cloud chart style
MSFT - MA10 = 27.27, MA12 = 27.19, MA14 = 27.13, MA16 = 27.13, MA18 = 27.13, MA20 = 27.12, MA22 = 27.11, MA24 = 27.09
True color (24bit) chart
Cloud style with clipping region
Sheet 1 Sheet 1 Sheet 3 Sheet 1 Sheet 5 Sheet 3 Sheet 7 Sheet 2 Sheet 9 Sheet 10
Alert Output
Realtime Quote
Select Symbol
NYSE, ETF, Services, Advertising
ESignal3
OK
Interpretation
Notepad
Symbols
Layouts
Layers
Charts

The formula for chart in the middle pane (rainbow 24-bit multiple moving averages) looks as follows:

```
side = 1;
increment = Param("Increment",2, 1, 10, 1 );
for( i = 10; i < 80; i = i + increment )
{
 up = MA( C, i );
 down = MA( C, i + increment );

 if( ParamToggle("3D effect?", "No|Yes", 1 ) )
   side =  IIf(up<=down AND Ref( up<=down, 1 ), 1, 0.6 );

 PlotOHLC( up,up,down,down, "MA"+i, ColorHSB( 3*(i - 10),
           Param("Saturation", 128, 0, 255 ),
           side * Param("Brightness", 255, 0, 255 ) ), styleCloud | styleNoLabel
);
}
```

The formula for the chart in the lower pane (slow stochastic %K with colored tops and bottoms) looks as follows. It uses styleClipMinMax to achieve clipping of the cloud region between min and max levels specified in the plot statement. Without this style area between min/max would be filled. Please note that due to Windows GDI limitation clipping region (styleClipMinMax) is supported only on raster (bitmap) devices so it is not compatible with printers or WMF (windows metafile) output.

```
SetChartOptions(0,0,ChartGrid30 | ChartGrid70 );
r = StochK(14);
Plot( r, "StochK", colorBlack );
 PlotOHLC( r,r,50,r, "", IIf( r > 50, colorRed, colorGreen ), styleCloud |
styleClipMinMax, 30, 70 );
```

X-shift feature

The XShift parameter allows to displace (shift) the plot in horizontal direction by specified number of bars. This allows to plot displaced moving averages and projections into the future. See the following sample code of displaced moving average:

```
Periods = Param("Periods", 30, 2, 100 );
Displacement = Param("Displacement", 15, -50, 50 );

Plot( MA( C, Periods ), _DEFAULT_NAME(), ColorCycle, styleLine, 0, 0,
Displacement );
```

PlotForeign() function

It is now easy to overlay price plots of multiple symbols using PlotForeign function:

PlotForeign(*tickersymbol*, *name*, *color/barcolor*, *style* = *styleCandle* | *styleOwnScale*, *minvalue* = {empty}, *maxvalue* = {empty}, *xshift* = 0)

Plots the foreign-symbol price chart (symbol is defined by *tickersymbol* parameter). Second argument *name* defines graph name used for displaying values in a title bar. Graph color could be static (if third argument is a number) or dynamic (when third argument is an array). Color indexes are related to the current palette (see Preferences/Color)
style defines chart plot style (see Plot() function for possible values)

```
      PlotForeign( "^DJI", "Dow Jones", colorRed );
      PlotForeign( "^NDX", "Nasdaq 100", colorBlue );
      PlotForeign( "^IXIC", "Nasdaq Composite", colorGreen );
```

Multiple plots using different scaling

Two new styles can be used to plot multiple graphs using different Y-scale: styleOwnScale and styleLeftAxisScale.

It also makes it easy to plot 2 or more "own scale" plots with the same scaling:

```
      minimum = LastValue( Lowest( Volume ) );
      maximum = LastValue( Highest( Volume ) );

      Plot( Close, "Price", colorBlue, styleCandle );

      /* two plots below use OwnScale but the scale is common because we
      set min and max values of Y axis */
      Plot( Volume, "Volume", colorGreen, styleHistogram | styleThick |
      styleOwnScale, minimum, maximum );
      Plot( MA( Volume, 15 ), "MA volume", colorRed, styleLine |
styleOwnScale, minimum, maximum );
```

New style: styleLeftAxisScale = 65536 - allows to plot more than one graph using common scaling but different from regular (right axis) scale.
Example: price plot plus volume and moving average plot:

```
// Plot price plot and its moving average
Plot( Close, "Price", colorWhite, styleCandle );
Plot( MA( Close, 20 ), "MAC", colorRed );

// Now plot volume and its moving average using left-hand axis
scaling
Plot( Volume , "Volume", colorBlue, styleLeftAxisScale |
styleHistogram | styleThick );
Plot( MA( Volume,15), "MAV", colorLightBlue, styleLeftAxisScale );
```

New parameters make it also easy to plot ribbons, for example:

```
Plot( Close, "Price", colorBlue, styleCandle );
Plot( 2, /* defines the height of the ribbon in percent of pane width
*/
"Ribbon",
IIf( up, colorGreen, IIf( down, colorRed, 0 )), /* choose color */
styleOwnScale|styleArea|styleNoLabel, -0.5, 100 );
```

Using custom defined parameters

AmiBroker allows to create user-defined parameters. Such parameters are then available via Parameters dialog for quick and fast adjustment of indicator.

Most often used parameter functions are (click on the links to get more detailed description):

- Param("name", default, min, max, steps, incr = 0)
- ParamStr("name", "default");
- ParamColor("name", defaultcolor);
- ParamStyle("name", defaultval = styleLine, mask = maskDefault)

They make it possible to define your own parameters in your indicators. Once Param functions are included in the formula you can right click over chart pane and select "Parameters" or press Ctrl+R, and change them via Parameters dialog and get immediate response.

The simplest case looks like this:

```
period = Param("RSI period", 12, 2, 50, 1 );
Plot( RSI( period ), "RSI( " + period + ") ", colorRed );
```

Right click over the chart and choose "Parameters" and move the slider and you will see RSI plotted with different periods immediatelly as you move the slider.

Sample code below shows how to use ParamStr to get the ticker symbol and ParamColor to get colors.

```
ticker = ParamStr( "Ticker", "MSFT" );
sp = Param( "MA Period", 12, 2, 100 );
PlotForeign( ticker, "Chart of "+ticker,
             ParamColor( "Price Color", colorBlack ), styleCandle );
Plot( MA( Foreign( ticker, "C" ), sp ), "MA", ParamColor( "MA Color",
colorRed ) );
```

The following sample formula (from AmiBroker mailing list) that allows to visually align price peak/troughs with sine curve on the chart:

```
Cycle = Param("Cycle Months", 12, 1, 12, 1 )*22;//264==12mth,22==1mth
xfactor = Param("Stretch",1,0.1,2,0.1);//1==1yr,2==2yr
xshift = Param("slide",0,-22,22,2)/3.1416^2;//slide curve 1==5days

x = 2*3.1416/Cycle/xfactor;
y = sin(Cum(x)-xshift);

Plot(C,"Daily Chart", colorBlack, styleCandle | styleNoLabel);
Plot(y,
 "cycle =" + WriteVal(Cycle*xfactor/22,1.0)+"months",
 colorBlue,styleLine|styleNoLabel|styleOwnScale);
```

Right click over the chart and choose "Parameters" and move the sliders and you will see chart immediatelly reflecting your changes.

For more information on user-definable parameters please check also Tutorial: Using drag-and-drop interface

Plotting texts at arbitrary positions on the chart

AmiBroker now allows annotation of the chart with text placed on any x, y position specified on the formula level using new PlotText function.

PlotText("text", x, y, color, bkcolor = colorDefault)

where
x - is x-coordinate in bars (like in LineArray)
y - is y-coordinate in dollars

color is text color, bkcolor is background color. If bkcolor is NOT specified (or equal to colorDefault) text is written with TRANSPARENT background, any other value causes solid background with specified background color

Example:

```
Plot(C,"Price", colorBlack, styleLine );
Plot(MA(C,20),"MA20", colorRed );

Buy=Cross( C, MA(C,20 ) );
Sell= Cross( MA( C, 20 ), C );

dist = 1.5*ATR(10);
```

```
for( i = 0; i < BarCount; i++ )
{
if( Buy[i] ) PlotText( "Buy\n@" + C[ i ], i, L[ i ]-dist[i], colorGreen );
if( Sell[i] ) PlotText( "Sell\n@" + C[ i ], i, H[ i ]+dist[i], colorRed,
colorYellow );
}

PlotShapes( Buy * shapeUpArrow + Sell * shapeDownArrow, IIf( Buy, colorGreen,
colorRed ) );
```

Gradient fill of the background

AmiBroker 4.90 allows to fill indicator background with gradually changing color. To achieve this you need to use new function SetChartBkGradientFill(topcolor, bottomcolor, titlebkcolor = default)

The function enables background gradient color fill in indicators.
Please note that this is independent from chart background color (background color fills entire pane, gradient fill is only for actual chart interior, so axes area is not affected by gradient fill). Parameters are as follows:

topcolor - specifies top color of the gradient fill
bottomcolor - specifies bottom color of the gradient fill
titlebkcolor - (optional) the background color of title text. If not specified then top color is automatically used for title background.

Example:

```
SetChartBkGradientFill( ParamColor("BgTop", colorWhite),ParamColor("BgBottom",
colorLightYellow));
```

Gradient fill area charts

Version 5.60 brings native gradient area charts. To display a simple gradient chart it is enough to use styleGradient in the Plot() function call. By default upper gradient color is specified by color parameter in Plot() function, bottom gradient color is either background color. styleGradient can be combined with styleLine.

A simple gradient area chart can be displayed using:

```
Plot( C, "C", colorDefault, styleGradient | styleLine );
```

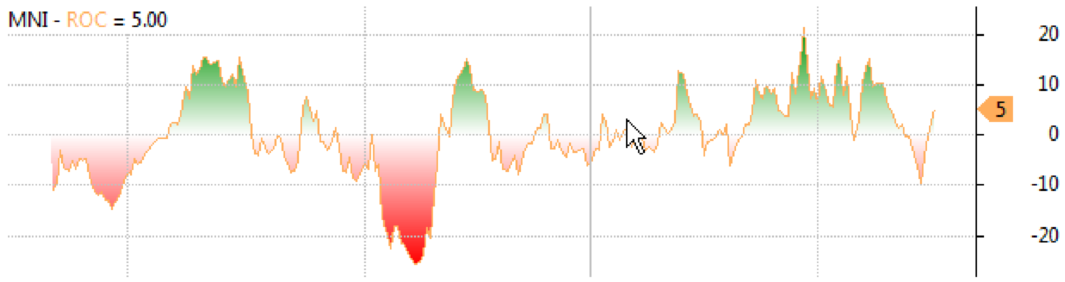
For detailed control over gradient colors and baseline there is an extra function SetGradientFill(topcolor, bottomcolor, baseline, baselinecolor) that should be called before Plot().

When you use SetGradientFill function, the upper gradient color is specified by topcolor argument, bottom gradient color is specified by botttomcolor. Optional parameters (baseline/baselinecolor) allow reverse gradient chart (such as underwater equity) and 3 color gradients top->baseline->bottom. See code for Underwater Equity for example usage of reverse gradient chart (with baseline at the top). Baseline parameter specifies the Y-axis position of chart baseline. The baselinecolor parameter specifies the color of gradient that is to be used at that level. If baselinecolor is not specified, then only 2-color gradient is plotted (topcolor->bottomcolor).

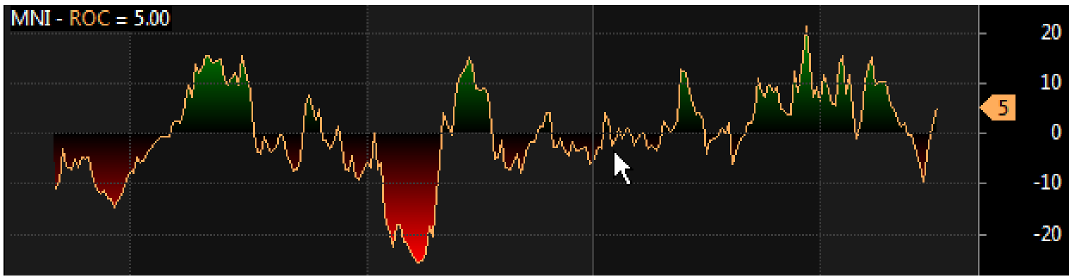
For example to display three-color gradient Rate Of Change that will use green as "top" color for positive values, background color as "baseline" color and red as "bottom" color for negative values it is enough to write:

```
SetGradientFill( colorGreen /*top*/, colorRed /*bottom*/, 0 /*baseline level*/,
GetChartBkColor() /*baseline color */);
Plot( ROC( C, 14), "ROC", colorLightOrange, styleLine | styleGradient, Null,
Null, 0, -1  );
```

The resulting chart will look as follows (using Basic chart theme):



.. or this way (using Black chart theme):



Super thick charts

Version 5.60 allows to define the line width beyond styleThick that was the only option before.

Now 9th parameter of Plot() defines pixel or percent width of given plot. The default is 1 pixel. Positive values specify pixel width, negative values specify width in percent of current bar width. So for example -20 will give you dynamic width that is 20% of bar width. Example:

```
Plot( C, "Close", colorDefault, styleBar, Null, Null, 0, 1, -20 /* line width as
percent of bar */ );
```

As you zoom-in the bars will become thicker and thicker.

Now you can get super thick lines as shown in the example below (10-pixel thick line chart):

```
Plot( C, "Close", colorRed, styleLine, Null, Null, 0, 1, 10 /* 10 pixel wide */
);
```

Miscellaneous

As you already know each plot has its own name that is used to create a title string which displays names and values of indicators. AmiBroker however allows you to override this automatic mechanism and define your own title string from the scratch. The **Title** reserved variable is used for that. You just assign a string to it and it will be displayed in the chart instead of automatically generated one.

Also there two more reserved variables (GraphXSpace and GraphZOrder) that allow to fine-tune indicator look.

They are all described in the table below.

Variable	Usage	Applies to
Title	Defines title text If you use Title variable you can specify colors in the string. It is advised to use AFL **EncodeColor** function that makes it easier than coding escape sequences. EncodeColor(colornumber). And you can write the above example like this: Title = "This is written in " + EncodeColor(colorViolet) + "violet color " + EncodeColor(colorGreen) + "and this in green"; Multi-line caption is possible by simply embedding line break \n, for example: Title = "This is 1st line\nThis is second line"; *For sake of completeness: colors can also be specified using espace sequences but it is NOT recommended because is hard to write and hard to read. \\cXX sequence where XX is 2 digit number specifying color index \\c38 - defines violet, there is a special sequence \\c-1 that resets to default axis color.* *For example* *Title = "This is written in \\c38violet color \\c27and this in green";*	Indicators
Tooltip	**Obsolete in 5.40**. Use Data window instead or use Plot() with styleHidden if you want to add your custom values to data tooltip. For example: Plot(my_value, "MyValueForTooltip", colorBlack, styleHidden);	Indicators
GraphXSpace	defines how much extra space should be added above and below graph line (in percent). For example: GraphXSpace = 5;	Indicators

	adds 5% extra space above and below the graph line. When GraphXSpace is not defined in the formula then default 2% is used.	
GraphLabelDecimals	(new in 5.90) controls number of decimals in chart value labes (example, adding GraphLabelDecimals = 2; to the formula would give you value lables with 2 decimal places)	Indicators
GraphZOrder	GraphZOrder variable allows to change the order of plotting indicator lines. When GraphZOrder is not defined or is zero (false) - old ordering (last to first) is used, when GraphZOrder is 1 (true) - reverse ordering is applied.	Indicators

Obsolete graph variables

This table shows obsolete reserved variables. They are still functional for backward-compatibility but new code should use Plot() functions only. What's more, when using new Plot() functions you should NOT use obsolete variables below.

Variable	Usage	Applies to
maxgraph	specifies maximum number of graphs to be drawn in custom indicator window (default=3)	Indicators
graph*N*	defines the formula for the graph number *N* (where *N* is a number 0,1,2,..., maxgraph-1)	Indicators
graph*N*open, graph*N*high, graph*N*low,	define additional O, H, L price arrays for candlestick and traditional bar charts	Indicators
graph*N*color	defines the color index of *N*th graph line. Color indexes are related to the current palette - see Preferences/Color.	Indicators
graph*N*barcolor	defines the array that holds palette indexes for each bar drawn	Indicators
graph*N*style	defines the style of *N*th graph. Style is defined as a combination (sum) of one or more following flags (you can use predefined style__ constants instead of numbers)	Indicators

How to create your own exploration

One of the most useful features of the Analysis window is called "Exploration". Basically, an exploration works in a similar way to scan but instead of looking for and reporting just buy/sell signals it allows you to generate customizable screening (or exploration) report that can give you much more information than simple scan.

The idea behind an exploration is simple - one variable called **filter** controls which symbols/quotes are accepted. If "true" (or 1) is assigned to that variable for given symbol/quote it will be displayed in the report.

So, for example, the following formula will accept all symbols with closing prices greater than 50 :

```
filter = close > 50;
```

*(NOTE: To create new formula please open Formula Editor using **Analysis->Formula Editor** menu, type the formula and choose **Tools->Send to Analysis** menu in Formula editor)*

Note that exploration uses all range and filter settings that are also used by back-tester and scanning modes so you can get multiple signals (report lines) if you select "All quotations" range. To check just the most recent quote you should choose "**1 recent bar(s)**"

Now, what about customizable reports?

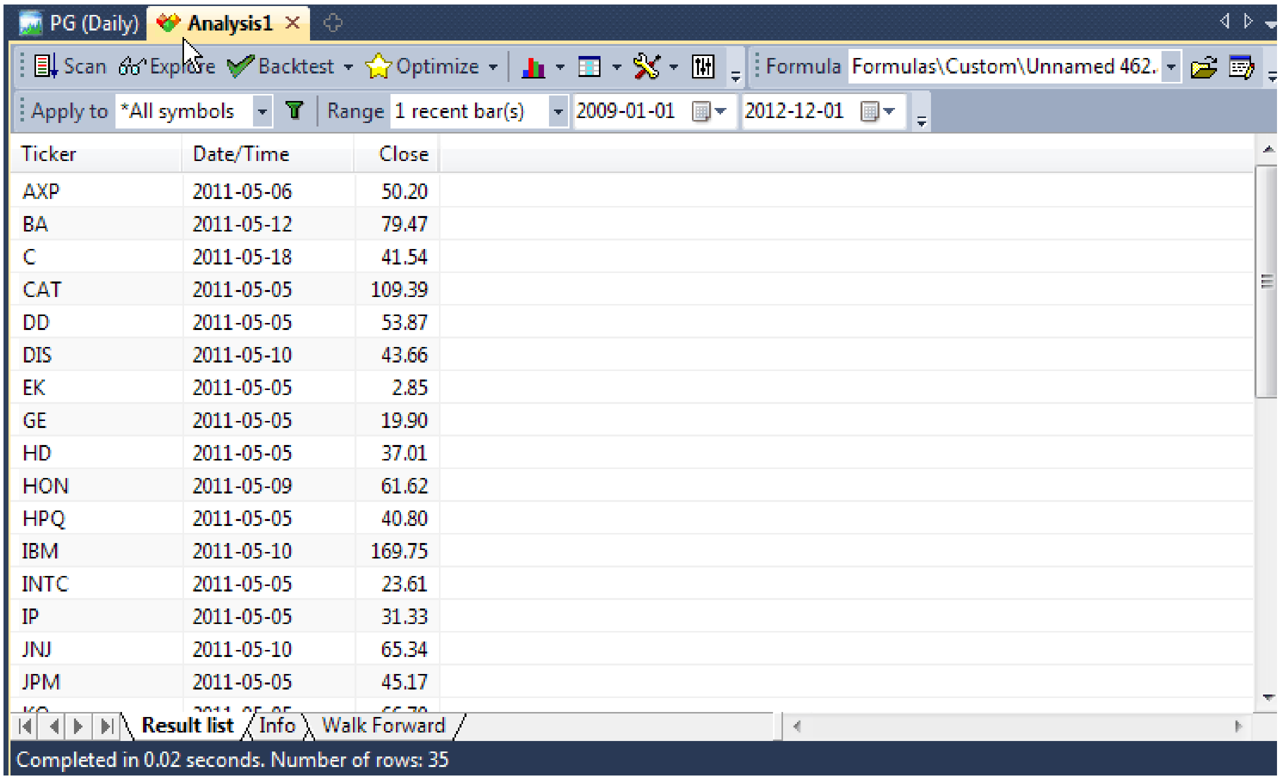
Yes, exploration mode allows you to create and then export a report with completely customizable columns and it is quite simple to do.

All you have to do is to tell AmiBroker what columns do you want. This can be done by calling AddColumn function in your exploration formula:

```
AddColumn( Close, "Close" );
```

The first argument of AddColumn function is the data ARRAY you want to display, the second argument defines the column caption

If you now press "**Explore**" button in the Analysis window you will get the result similar to this:



Ticker	Date/Time	Close
AXP	2011-05-06	50.20
BA	2011-05-12	79.47
C	2011-05-18	41.54
CAT	2011-05-05	109.39
DD	2011-05-05	53.87
DIS	2011-05-10	43.66
EK	2011-05-05	2.85
GE	2011-05-05	19.90
HD	2011-05-05	37.01
HON	2011-05-09	61.62
HPQ	2011-05-05	40.80
IBM	2011-05-10	169.75
INTC	2011-05-05	23.61
IP	2011-05-05	31.33
JNJ	2011-05-10	65.34
JPM	2011-05-05	45.17

Completed in 0.02 seconds. Number of rows: 35

Note that there are actually 3 columns: predefined Ticker and Date/Time column and one custom columnholding close price. Note that only tickers with close price greater than 50 are reported.

Now you can click "**Export**" and your exploration will be saved to CSV (comma separated values) file that could be easily loaded to any other program including Excel for further analysis.

Actually AddColumn function accepts more arguments to allow you to customize the output even more. The full syntax is:

AddColumn(array, name, format = 1.2, textColor = colorDefault, bkgndColor = colorDefault)

format parameter allows you to define the formatting applied to numbers. By default all variables are displayed with 2 decimal digits, but you can change this by assigning a different value to this variable: 1.5 gives 5 decimal digits, 1.0 gives no decimal digits. So, in our example, typing:

```
AddColumn( Close, "Close", 1.4 );
```

will give closing prices displayed with 4 decimal digits.
(Note for advanced users: the integer part of this number can be used to pad formatted number with spaces - 6.0 will give no decimal digits but a number space-padded upto 6 characters.)

There are also special format pre-defined constants that allow to display date/time and single character codes:

- formatDateTime - produces date time formated according to your system settings
  ```
  AddColumn( DateTime(), "Date / Time", formatDateTime );
  ```
- formatChar - allows outputting single ASCII character codes:

Example (produces signal file accepted by various other programs):

```
Buy=Cross(MACD(),Signal());
Sell=Cross(Signal(), MACD());
Filter=Buy OR Sell;
SetOption("NoDefaultColumns", True );
AddColumn( DateTime(), "Date", formatDateTime );
AddColumn( IIf( Buy, 66, 83 ), "Signal", formatChar );
```

textColor and **bkgndColor** arguments allow you to produce colorful reports. By default result list is displayed using system color but you can override this behaviour providing your own colors.

For example, the code that displays close price in green color when 1 day rate of change is positive and otherwise uses red color:

```
AddColumn( Close, "Close", 1.4, IIF( ROC(C, 1 ) > 0, colorGreen, colorRed ) );
```

Examples

The exploration mode is extermely flexible: you can, for example, export the whole database to CSV file using the following formula:

```
filter = 1; /* all symbols and quotes accepted */

AddColumn(Open,"Open",1.4);
AddColumn(High,"High",1.4);
AddColumn(Low,"Low",1.4);
AddColumn(Close,"Close",1.4);
AddColumn(Volume,"Volume",1.0);
```

This one will show you only heavily traded securities:

```
filter = volume > 5000000; /* adjust this threshold for your own
needs */
AddColumn(Close,"Close",1.4);
AddColumn(Volume,"Volume",1.0);
```

or...just show securities with volume being 30% above its 40-day exponential average

```
filter = volume > 1.3 * ema( volume, 40 );
AddColumn(Close,"Close",1.4);
AddColumn(Volume,"Volume",1.0);
```

With this one, you can export multiple indicator values for further analysis:

```
filter = close > ma( close, 20 ); /* only stocks trading above its 20
day MA*/
AddColumn( macd(), "MACD", 1.4 );
AddColumn( signal(), "Signal", 1.4 );
AddColumn( adx(), "ADX", 1.4 );
AddColumn( rsi(), "RSI", 1.4 );
AddColumn( roc( close, 15 ), "ROC(15)", 1.4 );
AddColumn( mfi(), "MFI", 1.4 );
```

```
      AddColumn( obv(), "OBV", 1.4 );
      AddColumn( cci(), "CCI", 1.4 );
      AddColumn( ultimate(), "Ultimate", 1.4 );
```

One more example of color output:

```
      Filter =1;

      AddColumn( Close, "Close", 1.2 );
      AddColumn( MACD(), "MACD", 1.4 , IIf( MACD() > 0, colorGreen,
      colorRed ) );
      AddTextColumn( FullName(), "Full name", 77 , colorDefault, IIf( Close
      < 10, colorLightBlue, colorDefault ) );
```

Scatter (X-Y) charts in Exploration

Version 5.60 brings a new feature to the exploration - scatter X/Y charts. Scatter charts are useful to display relationships between many symbols such as correlation, risk, etc. They can be seen as replacement and upgrade to "Risk/yield" map that was hard coded to just one function. Now you can code your own X-Y charts that are not limited to just risk/yield maps.

All you need to do to display your own scatter plot is to add XYChartAddPoint to your formula for each X-Y point you want to have on your chart.

For example you can get scatter plot of MFE/Profit and MAE/Profit relationships as shown in the description of XYChartAddPoint AFL function.

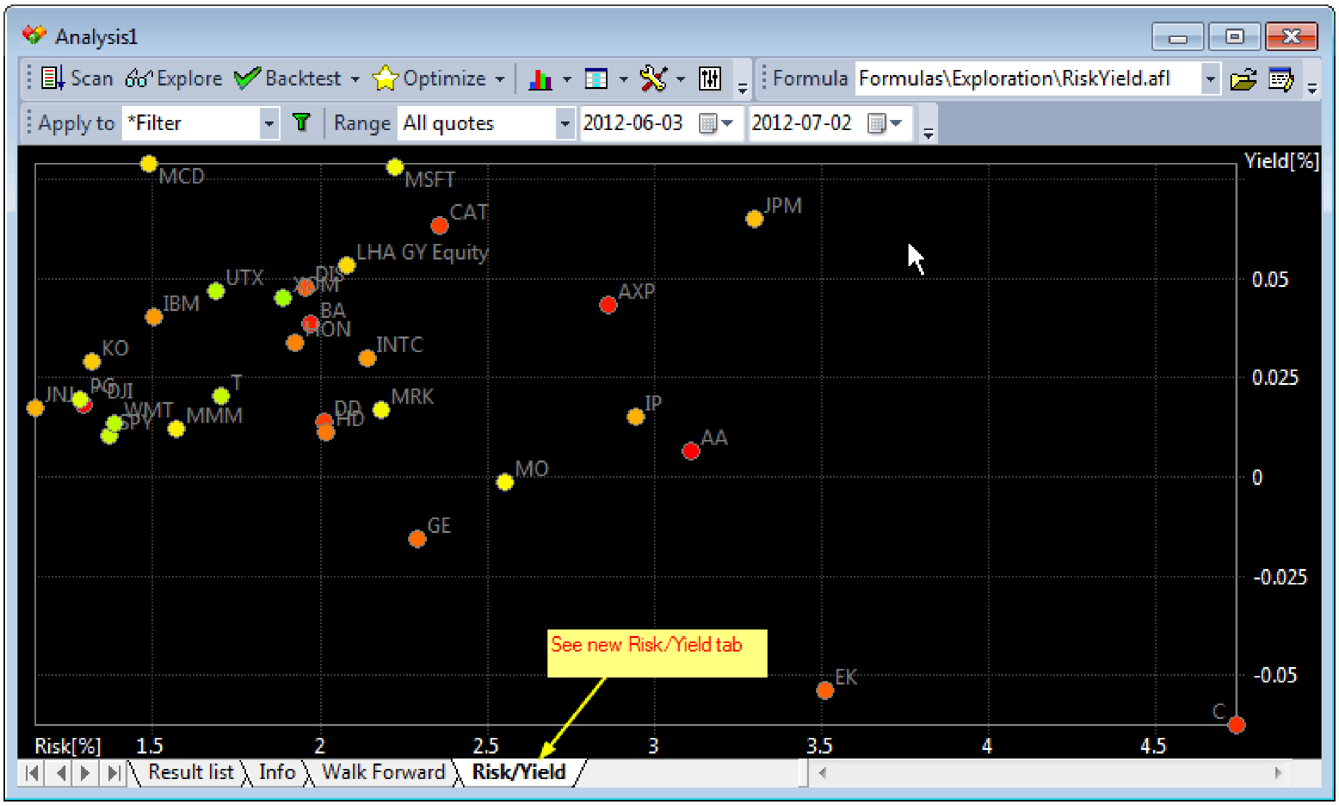
To display risk/yield scatter chart using new functions follow the steps below.

1. Click **File**->**New**->**Analysis**

2. Pick "Formulas\Exploration\RiskYield.afl" file (listed below)

3. Click on **Explore** button in the new Analysis window

4. In the bottom row of tabs you will see new "Risk/Yield" tab, click on it and you will see XY chart generated during exploration:



You can hover the mouse over that X-Y chart to read the values and you can click, drag to mark rectangle to zoom in. Click without marking rectangle restores full view.

```
// XY scatter chart example
// This is AFL equivalent of Risk-Yield map
// Note that this exploration should be run on
// WEEKLY data
// it calculates average weekly gain (yield)
// and standard deviation of gains (risk)

Filter=Status("lastbarinrange");
Length = SelectedValue( BarIndex() );
Chg = ROC( C, 1 ); //one bar yield
yield = MA( Chg, Length - 1);
risk =  StDev( Chg, Length - 1);
AddColumn(yield,"yield");
AddColumn(risk,"risk");

Clr = ColorHSB( 2 * Status("stocknum") % 255, 255, 255 );


XYChartAddPoint( "Risk/Yield", Name(), risk[ Length ], yield[ Length ] , Clr );
XYChartSetAxis("Risk/Yield", "Risk[%]", "Yield[%]");
```

Final tip

Please don't forget that you can sort the results of the exploration by any column by simply clicking on its header.

How to write your own chart commentary

One of the interesting aspects of using AmiBroker Formula Language is writing automatic chart commentaries. The idea behind this technique is as follows:

1. You write the commentary formula that consists of two basic elements: static texts and AFL expressions
2. AmiBroker evaluates expressions using currently selected symbol data and generates dynamic content
3. The mixture of static text and evaluated formulas are displayed in commentary output window
4. Additionally buy/sell arrows are plotted on the chart

Commentaries are available from *Analysis->Commentary* menu. When you open commentary window you will see two tabs: *Commentary* and *Formula.* In the *Formula* tab you can type the AFL statements which will be evaluated by AmiBroker resulting in dynamic commentary that appears in *Commentary* tab. The following sections will guide you through the steps needed to write your very own commentary formulas.

Writing static texts

Static text elements written in the formula should be enclosed in the quotation marks and terminated by semicolon sign as shown below:

```
"This is sample static text statement";
```

You can write several statements and each statement will be placed in a new line in the commentary output window:

```
"This is first line of text";
"This is second line of text";
```

Please type these examples into edit field in the *Formula* tab and switch to *Commentary* tab. You will see the texts displayed in the output area but without any quotation marks or semicolons. This is because AmiBroker has evaluated this simple text statements into strings and it displayed the strings in the output window.

Instead of just typing the text, it is advised that any new code should use **printf** function instead.

```
printf( "This is sample static text statement" );
```

To write several lines of text you can use a couple of statements as shown above or you can do this using single statement and line break sequence ('\n'):

```
printf( "This is first line of text\nThis is second line of
text\nThis is third line of text" );
```

You can also concatenate the string constants which will result in single line text:

```
printf( "This" +
" is" +
" single"+
" line" + " of text" );
```

Colors and styles

Since version 5.90 commentary and interpretation windows support colors and bold/italic styles. To specify beginning and end of bold section use **<b>** and **</b>** tags. To specify beginning and end of italic section use **<i>** and **</i>** tags. To change text color use EncodeColor as shown in the example below:

```
printf("<b>Bold text</b>\n");
printf("<i>Italic text</i>\n");
printf("Now " + EncodeColor( colorRed ) + "red text\n");
printf("and finally " + EncodeColor( colorGreen ) + "green <b>AND bold <i>AND
italic</i></b>\n");
printf(EncodeColor( colorBlack ) + "going back to black");
```

Dynamic content

I guess that you are quite bored with these simple examples, let's start with some dynamic content.

To enable dynamic commentaries AFL has a couple of special functions available, but two of them are the most important: NumToStr() and WriteIF(). WriteIF() function is used for conditional text display and will be described later in this article, now let us see what we can do using NumToStr() function.

The AFL reference manual says:

SYNTAX	NumToStr(NUMBER);
	NumToStr(ARRAY);

RETURNS	STRING
FUNCTION	This function can only be used within an Guru commentary. It is used to display the numeric value of NUMBER or ARRAY.

So, if you want to display a value of a number or currently selected bar of the array you should use NumToStr() function. But... wait a minute - what does it mean "currently selected bar of the array"? Let me explain this using simple formula (please type it in the *Formula* tab):

```
      printf( NumToStr( close ) );
```

When you switch to *Commentary* tab you will see the value of closing price (the same one which is displayed at the top of main price chart). But when you click on the chart in another place, selecting different date and then you click "Refresh" button you will see different value - the closing price at day you have selected. So NumToStr(close) function displays the value of currently selected bar of close array. And it works exactly the same way with other arrays. If you write

```
      printf( NumToStr( macd() ) );
```

you will see the exact value of MACD indicator at the day you have selected in the main chart. Having our current know-how we are able to write some statistics:

```
      printf( "Closing price = " + NumToStr( close ) + "\n" );
      printf( "Change since yesterday = " + NumToStr( close - ref( close,
      -1 ) ) + "\n" );
      printf( "Percent chg. since yesterday = " + NumToStr( roc( close, 1 )
      ) + " %%\n" );
      printf( "MACD =" + NumToStr( macd() ) + " , Signal line =" +
        NumToStr( signal() ) + "\n" );
```

When you switch to *Commentary* tab you will see output similiar to this one:

```
        Closing price = 17.940
        Change since yesterday = -0.180
        Percent chg. since yesterday = -0.993 %
        MACD = -0.001 , Signal line = 0.063
```

Quite nice, isn't it? You can also write current symbol ticker and selected date using name() and date() functions as shown below:

```
        printf( "Statistics of " + name() + " as of " + date() );
```

Instead of using NumToStr to convert number to string, we can format numbers directly using printf flexible % format specifiers. For example using %.2f means write a number with 2 decimal places, %.3f will mean write a number with 3 decimal places, %g will mean write a number with minimum required number of digits (auto-format). So we could write our previous example as follows:

```
printf( "Closing price = %.3f\n", close );
printf( "Change since yesterday = %.3f\n", close - ref( close, -1 ) );
printf( "Percent chg. since yesterday = %.2f%%\n", roc( close, 1 ) );
printf( "MACD = %.4f, Signal line = %.4f\n", macd(), signal() );
```

As we can see this code is shorter and clearer. The first argument of printf function is a string (strictly speaking so called formatting string that contains text and number placeholders/format specifiers marked with %. Subsequent arguments of printf function are actual values (numbers) we want to write (without need to convert to string anymore). As you may have noted, if we want to specify just percent sign, not a formatting sequence, we need to write %% (two percent signs).

But what we miss here is an ability to write something if some condition is met and write something different otherwise...

Conditional text output

AFL is equipped with very nice function called WriteIF() that can output different texts depending on the condition. Let us look what documentation says:

SYNTAX	writeif(EXPRESSION, "TRUE TEXT", "FALSE TEXT")
RETURNS	STRING
FUNCTION	This function can only be used within an Guru commentary. If EXPRESSION evaluates to "true", then the TRUE TEXT string is displayed within the commentary. If EXPRESSION evaluates to "false", then the FALSE TEXT string is displayed.

So we can easily output different text depending on expession, for example:

```
        writeif( macd() > signal(), "The MACD is bullish because is is above
        it's signal line", "The MACD is bearish because it is below its
        signal line" );
```

You can also combine several WriteIf() function calls in order to handle more possibilities:

```
"The current market condition for "+ name() + " is: ";

avgcond1 = ( c > ema( close, 200) ) + 0.1 * ( close > ema( close, 90)
) + 0.1 * ( close > ema( close , 30 ) );
avgcond2 = -( c < ema( close, 200) ) - 0.1 * ( close < ema( close,
90) ) - 0.1 * ( close < ema( close , 30 ) );

WriteIf( avgcond1 == 1.2,
"Very Bullish",
WriteIf( avgcond1 == 1.1,
"Bullish",
WriteIf( avgcond1 == 1.0,
"Mildly Bullish", "") ) ) +

WriteIf( avgcond2 == -1.2,
"Very Bearish",
WriteIf( avgcond2 == -1.1,
"Bearish",
WriteIf( avgcond2 == -1.0,
"Mildly Bearish", "") ) );
```

The formula above will return the text "The current market condition for {your ticker here} is: Very Bullish" if close price is above 30 day average and close is above 90 day average and close is above 200 day average. In other cases the formula will give you Bullish, Mildly Bullish, Mildly Bearish, Bearish or Very Bearish ratings.

For more examples on AFL commentaries please check AFL formula library especially MACD commentary formula which demonstrates all techniques presented here.

Now you are ready to start with your own commentaries... Good luck!

Using studies in AFL formulas

AmiBroker 3.52 introduces ability to reference hand-drawn studies from AFL formulas. This feature is quite unique among trading software and as you will find out using this feature is quite easy.

I will show you an example how to check if the trend line is broken from AFL code. All we need to do is three simple steps:

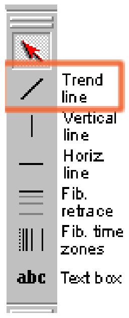
1. Draw a trend line
2. Define study ID
3. Write the formula that checks trend line break

Drawing trend line

A trend line is a sloping line drawn between two prominent points on a chart.

In this example we will draw the rising trend line that defines the uptrend. This kind of trend line is usually drawn between two (or more) troughs (low points) to illustrate price support.

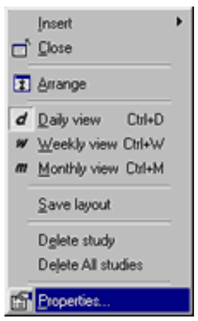
For sure you know how to draw a trend line in AmiBroker - just select a "Trend line" tool from "Draw" toolbar, find at least two recent troughs and just draw the line.



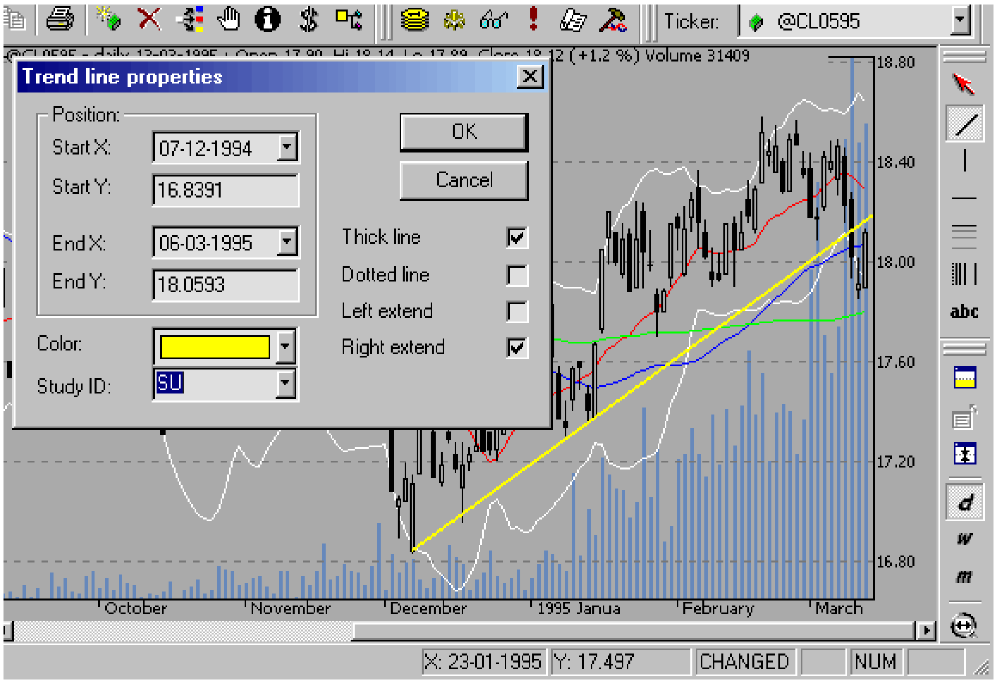
Define study ID

As you probably know, you can modify the properties of each line drawn in AmiBroker by clicking with the right mouse button over the study and selecting "Properties" from the menu. The properties dialog that shows up allows you to define exact start/end points and choose line colour, style and left and/or right extension mode.

For further analysis we will use the right-extended trend line (click on appropriate checkbox) to make sure that the trend line is automaticaly extended when new data are added.



Since version 3.52 the properties dialog allows also to define "Study ID" (the combo below colour box). "Study ID" is a two-letter code of the study that can be assigned to any study within a chart that allows AmiBroker to reference it from AFL. Predefined identifiers are: "UP" - uptrend, "DN" - downtrend, "SU" - support, "RE" - resistance, "ST" - stop loss, however you can use ANY identifiers (there are no limitations except that AmiBroker accepts only 2 letter codes). This way if you draw the support lines in many symbols and give them all "SU" identifier then you will be able to reference the support line from AFL code.


Trend line properties

Position:

Start X: 07-12-1994

Start Y: 16.8391

End X: 06-03-1995

End Y: 18.0593

Color: 🟨

Study ID: SU

Thick line ✓

Dotted line

Left extend

Right extend ✓

X: 23-01-1995 Y: 17.497 CHANGED NUM

So we will assign the "SU" study ID to the rising support trend line we have just drawn.

Write the formula that checks trend line break

In this example we will detect if the closing price drops BELOW support trend line. This is actually very simple:

```
        Sell = Cross( Study( "SU", GetChartID() ), Close );
```

Note that study() function accepts two arguments: the first is StudyID two letter code that corresponds to one given in properites dialog; the second argument is chart ID - it should be taken either via GetChartID() function (then it refers to current indicator) or read from Parameter dialog, Axes & Grid: Miscellaneous: Chart ID.

Back-testing your trading ideas

Introduction

One of the most useful things that you can do in the analysis window is to back-test your trading strategy on historical data. This can give you valuable insight into strengths and weak points of your system **before** investing real money. This single AmiBroker feature is can save lots of money for you.

Writing your trading rules

First you need to have objective (or mechanical) rules to enter and exit the market. This step is the base of your strategy and you need to think about it yourself since the system must match your risk tolerance, portfolio size, money management techniques, and many other individual factors.

Once you have your own rules for trading you should write them as buy and sell rules in AmiBroker Formula Lanugage (plus short and cover if you want to test also short trading).

In this chapter we will consider very basic moving average cross over system. The system would buy stocks/contracts when close price rises above 45-day exponential moving average and will sell stocks/contracts when close price falls below 45-day exponential moving average.

The exponential moving average can be calculated in AFL using its built-in function EMA. All you need to do is to specify the input array and averaging period, so the 45-day exponential moving average of closing prices can be obtained by the following statement:

```
      ema( close, 45 );
```

The **close** identifier refers to built-in array holding closing prices of currently analysed symbol.

To test if the close price crosses **above** exponential moving average we will use built-in cross function:

```
      buy = cross( close, ema( close, 45 ) );
```

The above statement defines a buy trading rule. It gives "1" or "true" when close price crosses above ema(close, 45). Then we can write the sell rule which would give "1" when opposite situation happens - close price crosses **below** ema(close, 45):

```
      sell = cross( ema( close, 45 ), close );
```

Please note that we are using **the same** cross function but the **opposite** order of arguments.

So complete formula for long trades will look like this:

```
      buy = cross( close, ema( close, 45 ) );
      sell = cross( ema( close, 45 ), close );
```

*NOTE: To create new formula please open Formula Editor using **Analysis->Formula Editor** menu, type the formula and choose **Tools->Send to Analysis** menu in Formula editor*

Back testing

To back-test your system just click on the **Back test** button in the Automatic analysis window. Make sure you have typed in the formula that contains at least buy and sell trading rules (as shown above). When the formula is correct AmiBroker starts analysing your symbols according to your trading rules and generates a list of simulated trades. The whole process is very fast - you can back test thousands of symbols in a matter of minutes. The progress window will show you estimated completion time. If you want to stop the process you can just click Cancel button in the progress window.

Analysing results

When the process is finished the list of simulated trades is shown in the bottom part of Automatic analysis window. (the **Results** pane). You can examine when the buy and sell signals occurred just by double clicking on the trade in **Results** pane. This will give you raw or unfiltered signals for every bar when buy and sell conditions are met. If you want to see only single trade arrows (opening and closing currently selected trade) you should double click the line while holding SHIFT key pressed down. Alternatively you can choose the type of display by selecting appropriate item from the context menu that appears when you click on the results pane with a right mouse button.

In addition to the results list you can get very detailed statistics on the performance of your system by clicking on the **Report** button. To find out more about report statistics please check out report window description.

Changing your back testing settings

Back testing engine in AmiBroker uses some predefined values for performing its task including the portfolio size, periodicity (daily/weekly/monthly), amount of commission, interest rate, maximum loss and profit target stops, type of trades, price fields and so on. All these settings could be changed by the user using settings window. After changing settings please remember to run your back testing again if you want the results to be in-sync with the settings.

For example, to back test on weekly bars instead of daily just click on the **Settings** button select **Weekly** from **Periodicity** combo box and click **OK**, then run your analysis by clicking **Back test**.

Reserved variable names

The following table shows the names of reserved variables used by Automatic Analyser. The meaning and examples on using them are given later in this chapter.

Variable	Usage	Applies to
buy	defines "buy" (enter long position) trading rule	Automatic Analysis, Commentary
sell	defines "sell" (close long position) trading rule	Automatic Analysis, Commentary
short	defines "short" (enter short position - short sell) trading rule	Automatic Analysis
cover	defines "cover" (close short position - buy to cover) trading rule	Automatic Analysis
buyprice	defines buying price array (this array is filled in with the default values according to the Automatic Analyser settings)	Automatic Analysis

sellprice	defines selling price array (this array is filled in with the default values according to the Automatic Analyser settings)	Automatic Analysis
shortprice	defines short selling price array (this array is filled in with the default values according to the Automatic Analyser settings)	Automatic Analysis
coverprice	defines buy to cover price array (this array is filled in with the default values according to the Automatic Analyser settings)	Automatic Analysis
exclude	If defined, a true (or 1) value of this variable excludes current symbol from scan/exploration/back test. They are also not considered in buy and hold calculations. Useful when you want to narrow your analysis to certain set of symbols.	Automatic Analysis
roundlotsize	defines round lot sizes used by backtester (see explanations below)	Automatic Analysis (new in 4.10)
ticksize	defines tick size used to align prices generated by **built-in stops** (see explanations below) (note: it does not affect entry/exit prices specified by buyprice/sellprice/shortprice/coverprice)	Automatic Analysis (new in 4.10)
pointvalue	allows to read and modify future contract point value (see backtesting futures) CAVEAT: this AFL variable is by default set to 1 (one) regardless of contents of Information window UNLESS you turn ON futures mode (SetOption("FuturesMode", True))	Automatic Analysis (new in 4.10)
margindeposit	allows to read and modify future contract margin (see backtesting futures)	Automatic Analysis (new in 4.10)
positionsize	Allows control dollar amount or percentage of portfolio that is invested into the trade (see explanations below)	Automatic Analysis (new in 3.9)

Advanced concepts

Until now we discussed fairly simple use of the back tester. AmiBroker, however supports much more sophisticated methods and concepts that will be discussed later on in this chapter. Please note that the beginner user should first play a little bit with the easier topics described above before proceeding.

So, when you are ready, please take a look at the following recently introduced features of the back-tester:

a) AFL scripting host for advanced formula writers
b) enhanced support for short trades
c) the way to control order execution price from the script
d) various kinds of stops in back tester
e) position sizing
f) round lot size and tick size
g) margin account
h) backtesting futures

AFL scripting host is an advanced topic that is covered in a separate document available here and I won't discuss it in this document. Remaining features are much more easy to understand.

Short trade support

In the previous versions of AmiBroker, if you wanted to back-test system using both long and short trades, you could only simulate stop-and-reverse strategy. When long position was closed a new short position was opened immediatelly. It was because buy and sell reserved variables were used for both types of trades.

Now (with version 3.59 or higher) there are separate reserved variables for opening and closing long and short trades:

buy - "true" or 1 value opens long trade
sell - "true" or 1 value closes long trade
short - "true" or 1 value opens short trade
cover - "true" or 1 value closes short trade

Som in order to back-test short trades you need to assign short and cover variables.
If you use stop-and-reverse system (always on the market) simply assign sell to short and buy to cover

```
        short = sell;
        cover = buy;
```

This simulates the way pre-3.59 versions worked.

But now AmiBroker enables you to have separate trading rules for going long and for going short as shown in this simple example:

```
        // long trades entry and exit rules:
        buy = cross( cci(), 100 );
        sell = cross( 100, cci() );

        // short trades entry and exit rules:
        short = cross( -100, cci() );
        cover = cross( cci(), -100 );
```

Note that in this example if CCI is between -100 and 100 you are out of the market.

Controlling trade price

AmiBroker now provides 4 new reserved variables for specifying the price at which buy, sell, short and cover orders are executed. These arrays have the following names: buyprice, sellprice, shortprice and coverprice.

The main application of these variables is controlling trade price:

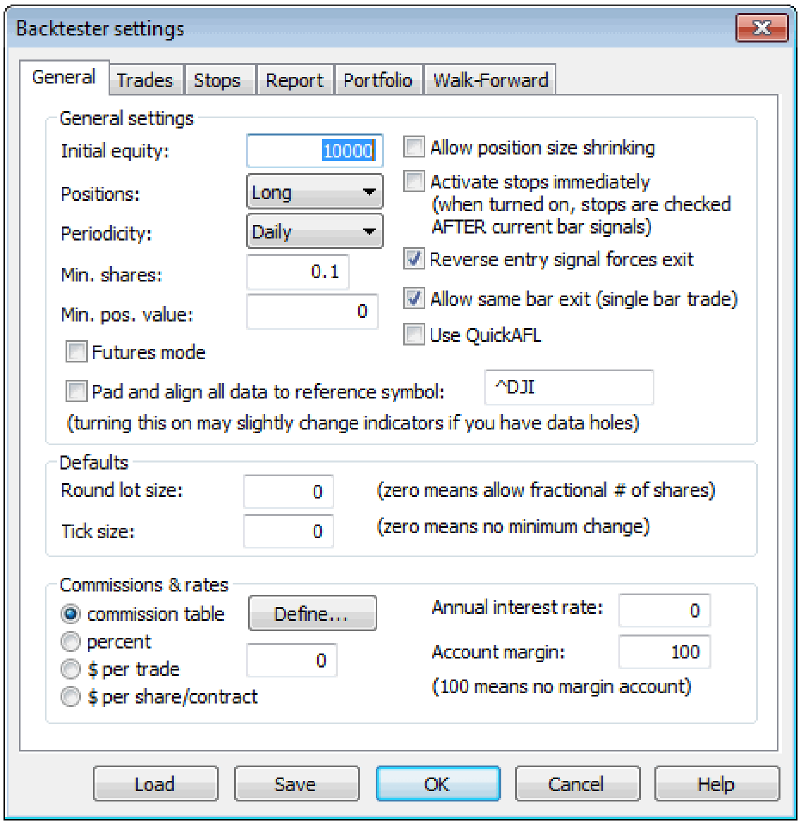
```
        BuyPrice = IIF( dayofweek() == 1, HIGH, CLOSE );
        // on monday buy at high, otherwise buy on close
```

So you can write the following to simulate real stop-orders:

```
        BuyStop = ... the formula for buy stop level;
        SellStop = ... the formula for sell stop level;

        // if anytime during the day prices rise above buystop level
        (high>buystop)
        // the buy order takes place (at buystop or low whichever is higher)
        Buy = Cross( High, BuyStop );
        // if anytime during the day prices fall below sellprice level ( low
        < sellstop )
        // the sell order takes place (at sellstop or high whichever is
        lower)
        Sell = Cross( SellPrice, SellStop);

        BuyPrice = max( BuyStop, Low ); // make sure buy price not less than
        Low
        SellPrice = min( SellStop, High ); // make sure sell price not
        greater than High
```

Please note that AmiBroker presets buyprice, sellprice, shortprice and coverprice array variables with the values defined in system test settings window (shown below), so you can but don't need to define them in your formula. If you don't define them AmiBroker works as in the old versions.

During back-testing AmiBroker will check if the values you assigned to buyprice, sellprice, shortprice, coverprice fit into high-low range of given bar. If not, AmiBroker will adjust it to high price (if price array value is higher than high) or to the low price (if price array value is lower than low)


Backtester settings

General settings

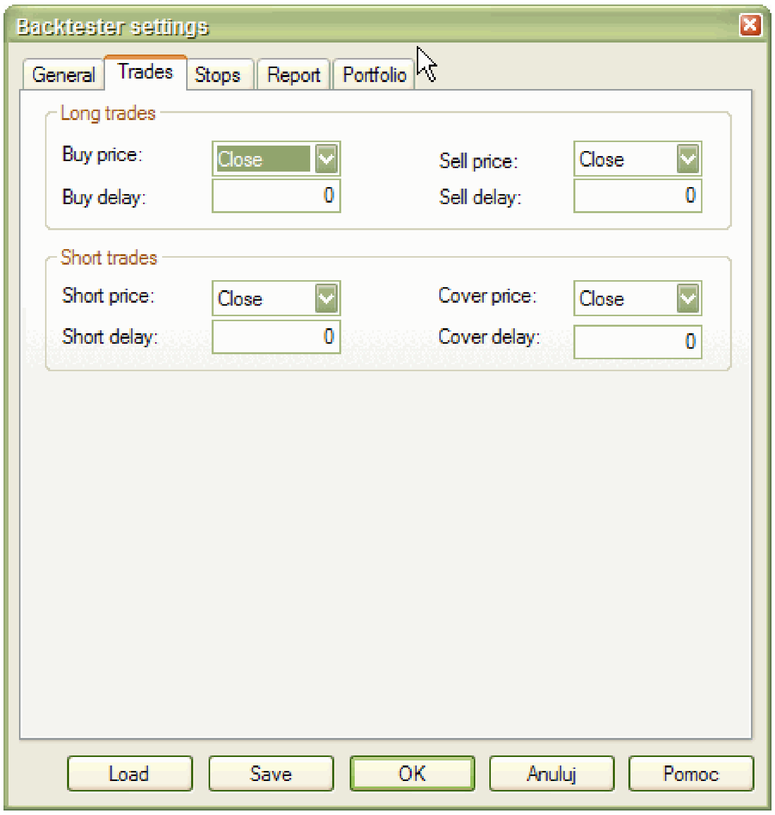
Initial equity: 10000
Positions: Long
Periodicity: Daily
Min. shares: 0.1
Min. pos. value: 0
[ ] Futures mode
[ ] Allow position size shrinking
[ ] Activate stops immediately (when turned on, stops are checked AFTER current bar signals)
[✓] Reverse entry signal forces exit
[✓] Allow same bar exit (single bar trade)
[ ] Use QuickAFL
[ ] Pad and align all data to reference symbol: ^DJI
(turning this on may slightly change indicators if you have data holes)

Defaults

Round lot size: 0 (zero means allow fractional # of shares)
Tick size: 0 (zero means no minimum change)

Commissions & rates

[● commission table] Define...
○ percent
○ $ per trade 0
○ $ per share/contract
Annual interest rate: 0
Account margin: 100
(100 means no margin account)

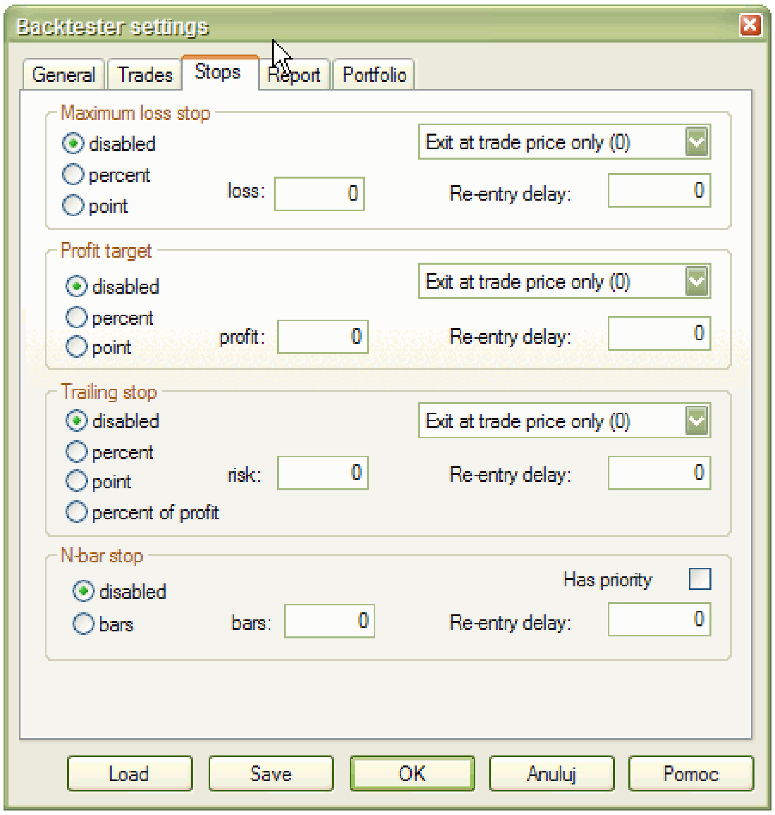


Long trades

Buy price:	Close	Sell price:	Close
Buy delay:	0	Sell delay:	0

Short trades

Short price:	Close	Cover price:	Close
Short delay:	0	Cover delay:	0



Maximum loss stop
[✓ disabled] percent point
loss: 0
Exit at trade price only (0)
Re-entry delay: 0

Profit target
[✓ disabled] percent point
profit: 0
Exit at trade price only (0)
Re-entry delay: 0

Trailing stop
[✓ disabled] percent point percent of profit
risk: 0
Exit at trade price only (0)
Re-entry delay: 0

N-bar stop
[✓ disabled] bars
bars: 0
Has priority ☐
Re-entry delay: 0

Profit target stops

As you can see in the picture above, new settings for profit target stops are available in the system test settings window. Profit target stops are executed when the **high** price for a given day exceedes the stop level that can be given as a percentage or point increase from the buying price. By default stops are executed at price that you define as sell price array (for long trades) or cover price array (for short trades). This behaviour can be changed by using "Exit at stop" feature.

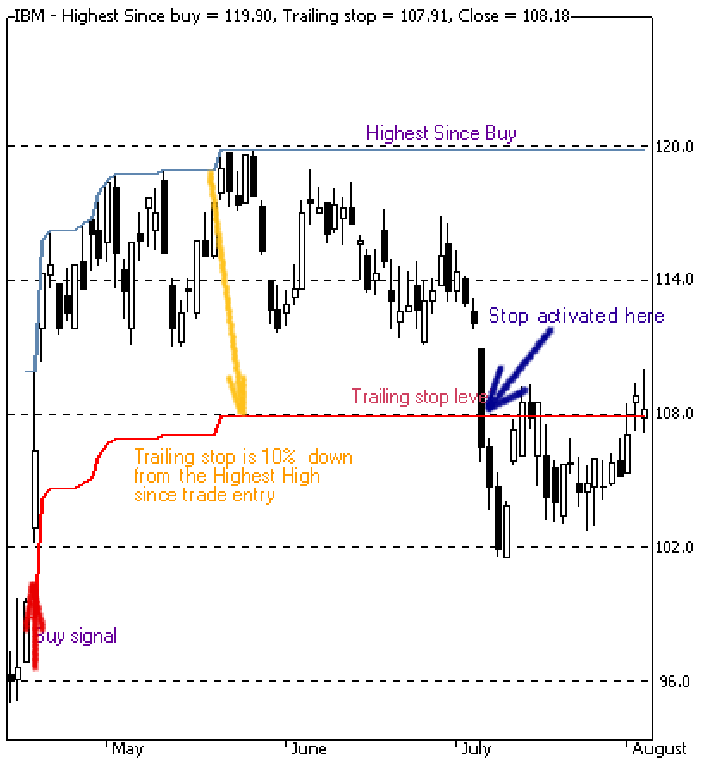
"Exit at stop" feature

If you mark "Exit at stop" box in the settings the stops will be executed at exact stop level, i.e. if you define profit target stop at +10% your stop and the buy price was 50 stop order will be executed at 55 even if your sell price array contains different value (for example closing price of 56).

Maximum loss stops work in a similar manner - they are executed when the **low** price for a given day drops below the stop level that can be given as a percentage or point increase from the buying price

Trailing stops

This kind of stop is used to protect profits as it tracks your trade so each time a position value reaches a new high, the trailing stop is placed at a higher level. When the profit drops below the trailing stop level the position is closed. This mechanism is illustrated in the picture below (10% trailing stop is shown):



<

The trailing stop, as well as two other kind of stops could be enabled from user interface (Automatic analysis' Settings window) or from the formula level - using ApplyStop function:

To reproduce the example above you would need to add the following code to your automatic analysis formula:

ApplyStop(2, 1, 10, 1); // 10% trailing stop, percent mode, exit at stop ON or you can write it using predefined constants that are more descriptive

ApplyStop(stopTypeTrail, stopModePercent, 10, True);

Trailing stops could be also defined in points (dollars) and percent of profit (risk). In the latter case the amount parameter defines the percentage of profits that could be lost without activating the stop. So 20% percent of profit (risk) stop will exit your trade that has maximum profit of $100 when the profit decreases below $80.

Dynamic stops

The ApplyStop() function allows now to change the stop level from trade to trade. This enables you to implement for example volatility-based stops very easily.

For example to apply maximum loss stop that will adapt the maximum acceptable loss based on 10 day average true range you would need to write:

ApplyStop(0, 2, 2 * ATR(10), 1);

or you can write it using predefined constants that are more descriptive

ApplyStop(stopTypeLoss, stopModePoint, 2 * ATR(10), True);

The function above will place the stop 2 times 10 day ATR below entry price.

As ATR changes from trade to trade - this will result in dynamic, volatility based stop level. Please note that 3rd parameter of ApplyStop function (the amount) is sampled at the trade entry and held troughout the trade. So in the example above it uses ATR(10) value from the date of the entry. Further changes of ATR do not affect the stop level.

See complete APPLYSTOP function documentation for more details.

Coding your own custom stop types

ApplyStop function is intended to cover most "popular" kinds of stops. You can however code your own kind of stops and exits using looping code. For example the following re-implements profit target stop and shows how to refer to the trade entry price in your formulas:

```
/* a sample low-level implementation of Profit-target stop in AFL: */

Buy = Cross( MACD(), Signal() );

priceatbuy=0;

for( i = 0; i < BarCount; i++ )
{
     if( priceatbuy == 0 && Buy[ i ] )
     priceatbuy = BuyPrice[ i ];

     if( priceatbuy > 0 && SellPrice[ i ] > 1.1 * priceatbuy )
     {
       Sell[ i ] = 1;
       SellPrice[ i ] = 1.1 * priceatbuy;
```

```
        priceatbuy = 0;
      }
      else
        Sell[ i ] = 0;
}
```

Position sizing

This is a new feature in version 3.9. Position sizing in backtester is implemented by means of new reserved variable

PositionSize = <size array>

Now you can control dollar amount or percentage of portfolio that is invested into the trade

- positive number define (dollar) amount that is invested into the trade for example:

  PositionSize = 1000; // invest $1000 in every trade

- negative numbers -100..-1 define percentage:
  -100 gives 100% of current portfolio size,
  -33 gives 33% of available equity for example:

  PositionSize = -50; /* always invest only half of the current equity */

- dynamic sizing example:

  PositionSize = - 100 + RSI();

  as RSI varies from 0..100 this will result in position depending on RSI values -> low values of RSI will result in higher percentage invested

If less than 100% of available cash is invested then the remaining amount earns interest rate as defined in the settings.

There is also a new checkbox in the AA settings window: "Allow position size shrinking" - this controls how backtester handles the situation when requested position size (via PositionSize variable) exceeds available cash: when this flag is checked the position is entered with size shinked to available cash if it is unchecked the position is not entered.

To see actual position sizes please use a new report mode in AA settings window: "Trade list with prices and pos. size"

For the end, here is an example of Tharp's ATR-based position sizing technique coded in AFL:

```
Buy = <your buy formula here>
Sell = 0; // selling only by stop

TrailStopAmount = 2 * ATR( 20 );
Capital = 100000; /* IMPORTANT: Set it also in the Settings: Initial Equity */
```

```
Risk = 0.01*Capital;
PositionSize = (Risk/TrailStopAmount)*BuyPrice;
ApplyStop( 2, 2, TrailStopAmount, 1 );
```

The technique could be summarized as follows:

The total equity per symbol is $100,000, we set the risk level at 1% of total equity. Risk level is defined as follows: if a trailing stop on a $50 stock is at, say, $45 (the value of two ATR's against the position), the $5 loss is divided into the $1000 risk to give 200 shares to buy. So, the loss risk is $1000 but the allocation risk is 200 shares x $50/share or $10,000. So, we are
allocating 10% of the equity to the purchase but only risking $1000. *(Edited excerpt from the AmiBroker mailing list)*

Round lot size and tick size

Round lot size

Various instruments are traded with various "trading units" or "blocks". For example you can purchase fractional number of units of mutual fund, but you can not purchase fractional number of shares. Sometimes you have to buy in 10s or 100s lots. AmiBroker now allows you to specify the block size on global and per-symbol level.

You can define per-symbol round lot size in the Symbol->Information page (pic. 3). The value of zero means that the symbol has no special round lot size and will use "Default round lot size" (global setting) from the Automatic Analysis settings page (pic. 1). If default size is set also to zero it means that fractional number of shares/contracts are allowed.

You can also control round lot size directly from your AFL formula using RoundLotSize reserved variable, for example:

```
      RoundLotSize = 10;
```

Tick size

This setting controls the minimum price move of given symbol. You can define it on global and per-symbol level. As with round lot size, you can define per-symbol tick size in the Symbol->Information page (pic. 3). The value of zero instructs AmiBroker to use "default tick size" defined in the Settings page (pic. 1) of Automatic Analysis window. If default tick size is also set to zero it means that there is no minimum price move.

You can set and retrieve the tick size also from AFL formula using TickSize reserved variable, for example:

```
      TickSize = 0.01;
```

Note that the tick size setting affects ONLY trades exited by built-in stops and/or ApplyStop(). The backtester assumes that price data follow tick size requirements and it does not change price arrays supplied by the user.

So specifying tick size makes sense only if you are using built-in stops so exit points are generated at "allowed" price levels instead of calculated ones. For example in Japan - you can not have fractional parts of yen so you should define global ticksize to 1, so built-in stops exit trades at integer levels.

Margin account

Account margin setting defines percentage margin requirement for **entire account**. The default value of *Account margin* is 100. This means that you have to provide 100% funds to enter the trade, and this is the way how backtester worked in previous versions. But now you can simulate a margin account. When you buy on margin you are simply borrowing money from your broker to buy stock. With current regulations you can put up 50% of the purchase price of the stock you wish to buy and borrow the other half from your broker. To simulate this just enter 50 in the *Account margin* field (see pic. 1) . If your intial equity is set to 10000 your buying power will be then 20000 and you will be able to enter bigger positions. Please note that this settings sets the margin for entire account and it is NOT related to futures trading at all. In other words you can trade stocks on margin account.

Additional settings

- "Reverse entry signal forces exit" check box to the Backtester settings.
  When it is ON (the default setting) - backtester works as in previous versions and closes already open positon if new entry signal in reverse direction is encountered. If this switch is OFF - even if reverse signal occurs backtester maintains currently open trade and does not close positon until regular exit (sell or cover) signal is generated.
  In other words when this switch is OFF backtester ignores Short signals during long trades and ignores Buy signals during short trades.

- "Allow same bar exit (single bar trade)" option to the Settings
  When it is ON (the default settings) - entry and exit at the very same bar is allowed (as in previous versions)
  if it is OFF - exit can happen starting from next bar only (this applies to regular signals,there is a separate setting for ApplyStop-generated exits). Switching it to OFF allows to reproduce the behaviour of MS backtester that is not able to handle same day exits.
- "Activate stops immediately"

  This setting solves the problem of testing systems that enter trades on market open. In versions prior to 4.09 backtester assumed that you were entering trades on market close so built-in stops were activated from the next day. The problem was when you in fact defined open price as the trade entry price - then same day price fluctuations did not trigger the stops. There were some published workarounds based on AFL code but now you don't need to use them. Simply if you trade on open you should mark "Activate stops immediately" (pic. 1).

  You may ask why do not simply check the buyprice or shortprice array if it is equal to open price. Unfortunatelly this won't work. Why? Simply because there are doji days when open price equals close and then backtester will never know if trade was entered at market open or close. So we really need a separate setting.
- "Use QuickAFL"

  QuickAFL(tm) is a feature that allows faster AFL calculation under certain conditions. Initially (since 2003) it was available for indicators only, as of version 5.14+ it is available in Automatic Analysis too.

  Initially the idea was to allow faster chart redraws through calculating AFL formula only for that part which is visible on the chart. In a similar manner, automatic analysis window can use subset of available quotations to calculate AFL, if selected “range” parameter is less than “All quotations".

  Detailed explanation on how QuickAFL works and how to control it, is provided in this Knowledge Base article: http://www.amibroker.com/kb/2008/07/03/quickafl/

Note that this option works not only in the backtester, but also in optimizations, explorations and scans.

See Also:

Portfolio-level backtesting article.

Backtesting systems for futures contracts article.

APPLYSTOP function description

Using AFL editor section of the guide.

Insider guide to backtester (newsletter 1/2002)

Portfolio-level backtesting

IMPORTANT: Please read first Tutorial: Backtesting your trading ideas article

New backtester **works on PORTFOLIO LEVEL**, it means that there is single portfolio equity and position sizing refers to portfolio equity. Portfolio equity is equal to available cash plus sum of all simultaneously open positions at given time.

AmiBroker's **portfolio backtester** lets you combine trading signals and trade sizing strategies into simulations which exactly mimic the way you would trade in real time. A core feature is its ability to perform dynamic money management and risk control at the portfolio level. Position sizes are determined with full knowledge of what's going on at the portfolio level at the moment the sizing decision is made. Just like you do in reality.

HOW TO SET IT UP ?

There are only two things that need to be done to perform portfolio backtest

1. You need to have first the formula that generates buy / sell / short /cover signals as described in "Backtesting your trading ideas" article

2. You should define how many simultaneous trades you want to test and what position sizing algorithm you want to use.

SETTING UP MAXIMUM NUMBER OF SIMULTANEOUSLY OPEN TRADES

There are two ways to set the maximum number of simultaneously open trades:

1. Go to the **Settings** dialog**,** switch to **Portfolio** tab and enter the number to **Max. Open Positions** field

2. Define the maximum in the formula itself (this overrides any setting in the Settings window) using SetOption function:

```
SetOption("MaxOpenPositions", 5 ); // This sets maximum number of open positions
to 5
```

SETTING UP POSITION SIZE

IMPORTANT: to enable more than one symbol to be traded you have to add PositionSize variable to your formula, so less than 100% of funds are invested in single security:

```
PositionSize = -25; // invest 25% of portfolio equity in single trade
```

or

```
PositionSize = 5000; // invest $5000 into single trade
```

There is a quite common way of setting both position size and maximum number of open positions so equity is spread equally among trades:

```
PosQty = 5; // You can define here how many open positions you want
SetOption("MaxOpenPositions", PosQty );
```

```
PositionSize = -100/PosQty; // invest 100% of portfolio equity divided by max.
position count
```

You can also use more sophisticated position sizing methods. For example volatility-based position sizing (Van Tharp-style):

```
PositionSize = -2 * BuyPrice/(2*ATR(10));
```

That way you are investing investing 2% of PORTFOLIO equity in the trade adjusted by BuyPrice/2*ATR factor.

USING POSITION SCORE

You can use new PositionScore variable to decide which trades should be entered if there are more entry signals on different securities than maximum allowable number of open positions or available funds. In such case AmiBroker will use the absolute value of PositionScore variable to decide which trades are preferred. See the code below. It implements simple MA crossover system, but with additional flavour of preferring entering trades on symbols that have low RSI value. If more buy signals occur than available cash/max. positions then the stock with lower RSI will be preferred. You can watch selection process if you backtest with "Detailed log" report mode turned on.

The code below includes also the example how to find optimum number of simultaneously open positions using new Optimization in Porfolio mode.

```
/*****
** REGULAR PORTFOLIO mode
** This sample optimization
** finds what is optimum number of positions open simultaneously
**
****/

SetOption("InitialEquity", 20000 );
SetTradeDelays(1,1,1,1);
RoundLotSize = 1;

posqty = Optimize("PosQty", 4, 1, 20, 1 );
SetOption("MaxOpenPositions", posqty);

// desired position size is 100% portfolio equity
// divided by PosQty positions

PositionSize = -100/posqty;

// The system is very simple...
// MA parameters could be optimized too...
p1 = 10;
p2 = 22;
// simple MA crossover
Short=Cross( MA(C,p1) , MA(C,p2) );
Buy=Cross( MA(C,p2) , MA(C,p1) );
// always in the market
Sell=Short;
```

```
Cover=Buy;

// now additional score
// that is used to rank equities
// when there are more ENTRY signals that available
// positions/cash
PositionScore = 100-RSI(); // prefer stocks that have low RSI;
```

BACKTEST MODES

AmiBroker 5.0 offers 6 different backtest modes:

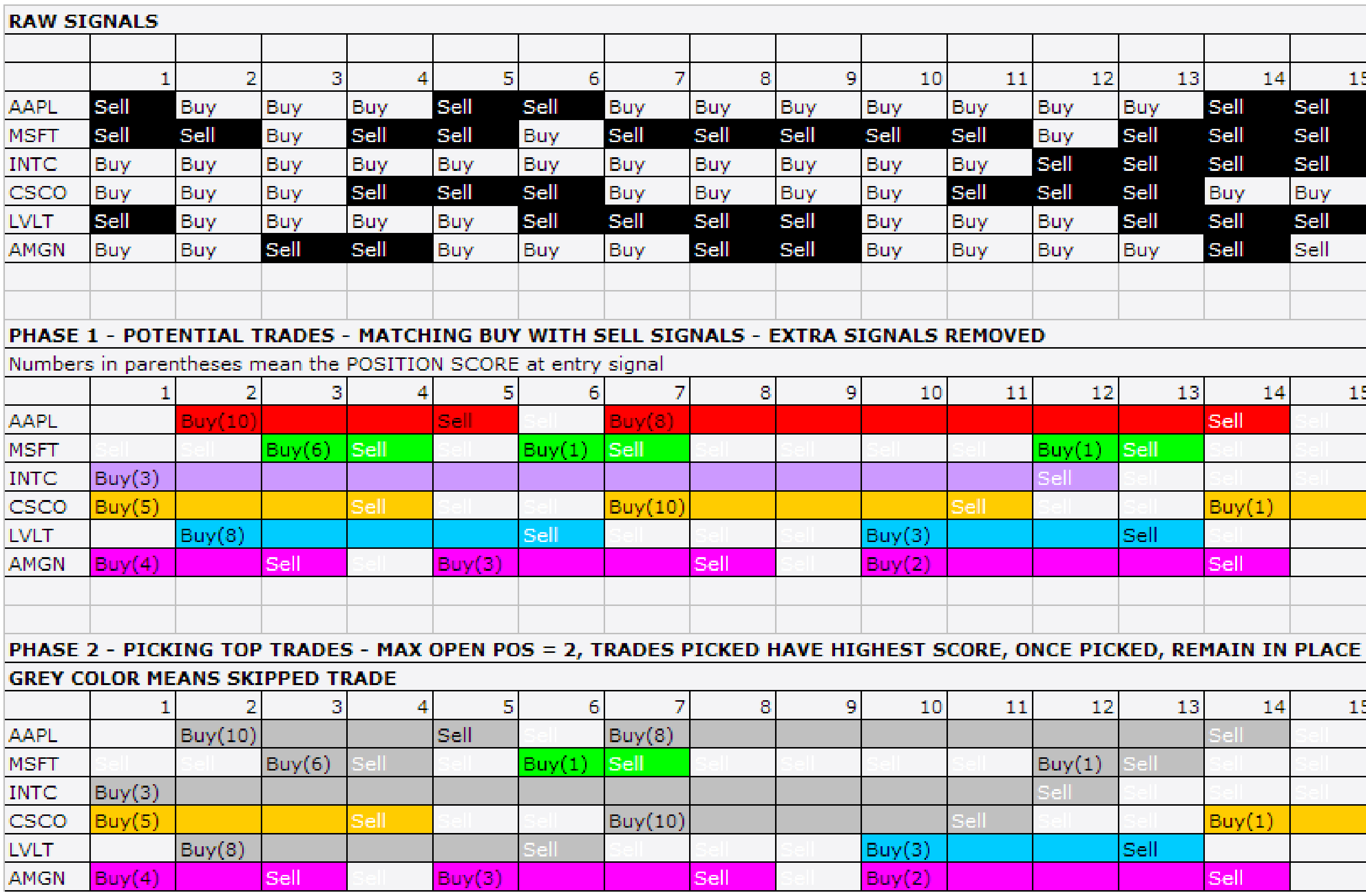
- regular mode (backtestRegular)
- regular raw mode (backtestRegularRaw)
- regular raw + multiple positions mode (backtestRegularRawMulti)
- regular raw2 mode (backtestRegularRaw2)
- regular raw2 + multiple positions mode (backtestRegularRaw2Multi)
- rotational trading mode (backtestRotational)

All "regular" modes use buy/sell/short/cover signals to enter/exit trades, while "rotational" mode (aka "ranking / switching" system) uses only position score and is descibed later.

Backtest modes are switchable using SetBacktestMode() AFL function.

The difference between "regular" modes is how repeated (also known as "redundant" or "extra") entry signals are handled. An "extra" entry signal is the signal that comes AFTER initial entry but before first matching exit signal.

In the regular mode - the default one, redundant entry signals are removed as shown in the picture below.



RAW SIGNALS

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
AAPL	Sell	Buy	Buy	Buy	Sell	Sell	Buy	Buy	Buy	Buy	Buy	Buy	Buy	Sell	Sell
MSFT	Sell	Sell	Buy	Sell	Sell	Buy	Sell	Sell	Sell	Sell	Sell	Buy	Sell	Sell	Sell
INTC	Buy	Buy	Buy	Buy	Buy	Buy	Buy	Buy	Buy	Buy	Buy	Sell	Sell	Sell	Sell
CSCO	Buy	Buy	Buy	Sell	Sell	Sell	Buy	Buy	Buy	Buy	Sell	Sell	Sell	Buy	Buy
LVLT	Sell	Buy	Buy	Buy	Buy	Sell	Sell	Sell	Sell	Buy	Buy	Buy	Sell	Sell	Sell
AMGN	Buy	Buy	Sell	Sell	Buy	Buy	Buy	Sell	Sell	Buy	Buy	Buy	Buy	Sell	Sell

PHASE 1 - POTENTIAL TRADES - MATCHING BUY WITH SELL SIGNALS - EXTRA SIGNALS REMOVED

Numbers in parentheses mean the POSITION SCORE at entry signal

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
AAPL		Buy(10)			Sell	Sell	Buy(8)							Sell	Sell
MSFT	Sell	Sell	Buy(6)	Sell	Sell	Buy(1)	Sell	Sell	Sell	Sell	Sell	Buy(1)	Sell	Sell	Sell
INTC	Buy(3)											Sell	Sell	Sell	Sell
CSCO	Buy(5)			Sell	Sell	Sell	Buy(10)				Sell	Sell	Sell	Buy(1)	
LVLT		Buy(8)				Sell	Sell	Sell	Sell	Buy(3)			Sell	Sell	
AMGN	Buy(4)		Sell	Sell	Buy(3)			Sell	Sell	Buy(2)				Sell	

PHASE 2 - PICKING TOP TRADES - MAX OPEN POS = 2, TRADES PICKED HAVE HIGHEST SCORE, ONCE PICKED, REMAIN IN PLACE

GREY COLOR MEANS SKIPPED TRADE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
AAPL		Buy(10)			Sell	Sell	Buy(8)							Sell	Sell
MSFT	Sell	Sell	Buy(6)	Sell	Sell	Buy(1)	Sell	Sell	Sell	Sell	Sell	Buy(1)	Sell	Sell	Sell
INTC	Buy(3)											Sell	Sell	Sell	Sell
CSCO	Buy(5)			Sell	Sell	Sell	Buy(10)				Sell	Sell	Sell	Buy(1)	
LVLT		Buy(8)				Sell	Sell	Sell	Sell	Buy(3)			Sell		
AMGN	Buy(4)		Sell	Sell	Buy(3)			Sell	Sell	Buy(2)				Sell	

As you can see Buy-Sell signal pairs are matched and treated as a TRADE. If trade is NOT entered on first entry signal due to weak rank, not enough cash or reaching the maximum open position count, subsequent entry signals are ignored until matching exit signal. After exit signal, the next entry signal will be possible candidate for entering trade. The process of removing excess signals occurring after first buy and matching sell (and short-cover pair respectively) is the same as ExRem() AFL function provides. To use regular mode you don't need to call SetBacktestMode function at all, as this is the default mode.

You may or may not consider removing extra signals desirable. If you want to act on ANY entry signal you need to use second mode - backtestRegularRaw. To turn it on you need to include this line in the code:

```
// signal-based backtest, redundant (raw) signals are NOT removed, only one
position per symbol allowed
SetBacktestMode( backtestRegularRaw );
```

It does NOT remove redundant entry signals and will act on ANY entry provided that it is scored highly enough and there is a cash available and maximum number of open positions is not reached. It will however allow only ONE OPEN POSITION per symbol at any given time. It means that if log trade is already open and later in the sequence appears an extra buy signal, it will be ignored until a "sell" signal comes (short-cover signals work the same). Note that you can still use sigScaleIn/sigScaleOut to increase or decrease the size of this existing position, but it will appear as single line in backtest result list.

If you want ALL repeated entry signals to be acted and allow to open multiple, separate positions on the same symbol without scaling in/out effect (so multiple positions on the same symbol open simultaneously appear as separate lines in the backtest report) you need to use backtestRegularRawMulti mode by adding the following line to the code:

```
SetBacktestMode( backtestRegularRawMulti );
```

In this mode MULTIPLE positions per symbol will be open if BUY/SHORT signal is "true" for more than one bar and there are free funds. Sell/Cover exit all open positions on given symbol, Scale-In/Out work on all open positions of given symbol at once.

Remark: The remaining modes are for advanced users only

Raw2 modes are "special" for advanced users of custom backtester. They are only useful if you do custom processing of exit signals in custom backtester procedure. They should NOT be used otherwise, because of performance hit and memory consumption Raw2 modes cause.

The common thing between Raw and Raw2 modes is that they both do NOT remove excess ENTRY signals. The difference is that Raw modes remove excess EXIT signals, while Raw2 do NOT.

In Raw2 modes all exit signals (even redundant ones) are passed to second phase of backtest just in case that you want implement strategy that skips first exit. Lets suppose that you want to exit on some condition from first phase but only in certain hours or after certain numbers of bars in trade or only when portfolio equity condition is met. Now you can do that in Raw2 modes.
Note that Raw2 modes can get significantly slower when you are using custom backtester code that iterates thru signals as there can be zillions of exit signals in the lists even for symbols that never generated any entry signals, therefore it is advised to use it only when absolutely necessary. Raw2 modes are also the most memory consuming. Note also that if you run the system WITHOUT custom backtest procedure there should be no difference between Raw and Raw2 modes (other than speed & memory usage) as first matching exit signal is what is used by default.

ROTATIONAL TRADING

Rotational trading (also known as fund-switching or scoring and ranking) is possible too. For more information see the description of EnableRotationalTrading function.

HOLDMINBARS and EARLY EXIT FEES

(Note that these features are available in portfolio-backtester only and not compatible with old backtester or Equity() function)

HoldMinBars is a feature that disables exit during user-specified number of bars even if signals/stops are generated during that period
Please note that IF during HoldMinBars period ANY stop is generated it is ignored. Also this period is ignored when it comes to calculation of trailing stops (new highest highs and drops below trailing stops generated during HoldMinBars are ignored).This setting, similar to EarlyExitFee/EarlyExitBars is available on per-symbol basis (i.e. it can be set to different value for each symbol)

Example:

```
SetOption("HoldMinBars", 127 );
Buy=BarIndex()==0;
Sell=1;
```

```
// even if sell signals are generated each day,
//they are ignored until bar 128
```

Early exit (redemption) fee is charged when trade is exited during first N bars since entry.
The fee is added to exit commission and you will see it in the commissions reported for example in detailed log. However, it is NOT reflected in the portfolio equity unless trade really exits during first N bars - this is to prevent affecting drawdowns if trade was NOT exited early.

```
 // these two new options can be set on per-symbol basis
// how many bars (trading days)
// an early exit (redemption) fee is applied
SetOption("EarlyExitBars", 128 );
// early redemption fee (in percent)
SetOption("EarlyExitFee", 2 );
```

(note 180 calendar days is 128 or 129 trading days)

```
// how to set it up on per-symbol basis?
// it is simple - use 'if' statement
if( Name() == "SYMBOL1" )
{
 SetOption("EarlyExitBars", 128 );
 SetOption("EarlyExitFee", 2 );
}

if( Name() == "SYMBOL2" )
{
 SetOption("EarlyExitBars", 25 );
 SetOption("EarlyExitFee", 1 );
}
```

In addition to HoldMinBars, EarlyExitBars there are sibling features (added in 4.90) called **HoldMinDays** and **EarlyExitDays** that work with **calendar days** instead of data bars. So we can rewrite previous examples to use calendar days accurately:

```
// even if sell signals are generated each day,
//they are ignored until 180 calendar days since entry
SetOption("HoldMinBars", 180 );
Buy=BarIndex()==0;
Sell=1;

// these two new options can be set on per-symbol basis
// how many CALENDAR DAYS
// an early exit (redemption) fee is applied
SetOption("EarlyExitDays", 180 );
// early redemption fee (in percent)
SetOption("EarlyExitFee", 2 );
```

(note 180 calendar days is 128 or 129 trading days)

```
// how to set it up on per-symbol basis?
// it is simple - use 'if' statement
if( Name() == "SYMBOL1" )
{
 SetOption("EarlyExitDays", 180 );
 SetOption("EarlyExitFee", 2 );
}

if( Name() == "SYMBOL2" )
{
 SetOption("EarlyExitDays", 30 );
 SetOption("EarlyExitFee", 1 );
}
```

RESOLVING SAME BAR, SAME SYMBOL SIGNAL CONFLICTS

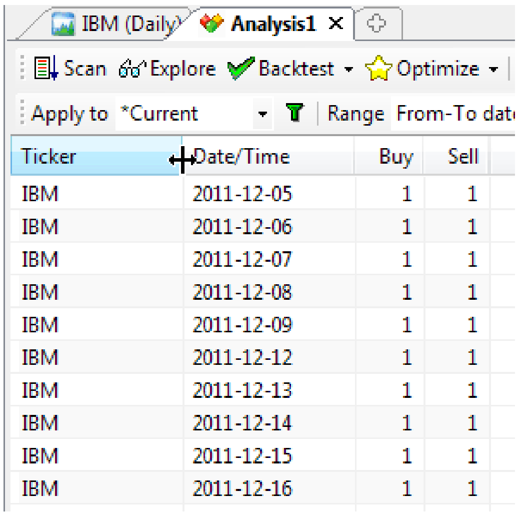
It is possible for the system to generate on the very same symbol both entry and exit signal at the very same bar. Consider for example, this very simple system that generates buy and sell signals on every bar:

```
Buy = 1;
Sell = 1;
```

If you add an exploration code to it to show the signals:

```
AddColumn(Buy,"Buy", 1.0 );
AddColumn(Sell, "Sell", 1.0 );
Filter = Buy OR Sell;
```

you will get the following output (when you press **Explore**);



Ticker	Date/Time	Buy	Sell
IBM	2011-12-05	1	1
IBM	2011-12-06	1	1
IBM	2011-12-07	1	1
IBM	2011-12-08	1	1
IBM	2011-12-09	1	1
IBM	2011-12-12	1	1
IBM	2011-12-13	1	1
IBM	2011-12-14	1	1
IBM	2011-12-15	1	1
IBM	2011-12-16	1	1

Now because of the fact that entry and exit signals do **NOT** carry any timing information, so you don't know which signal comes first, there are three ways how such conflicting same bar, entry and exit signals may be interpreted:

1. **only one signal is taken at any bar**, so trade that begins on bar 1 ends on bar 2 and next trade may only be open on bar 3 and closed on bar 4

2. **both signals are used** and **entry signal precedes exit signal**, so trade that begins on bar 1 ends on bar 1, then next trade opens on bar 2 and ends on bar 2, and so on (we have single-bar trades and we are out of market between bars)
3. **both signals are used** and **entry signal comes after exit signal.** In this situation the very first signal (exit) is ignored because we are flat, and trade is open on same bar entry signal. Then we don't have any more signals for given bar and trade is closed on the next bar exit signal, then we get another entry (same bar). So trade that begins on bar 1 ends on bar 2, then next trade opens on bar 2 and ends on bar 3, and so on (we have trades that span between bars, but both exit and entry signal occuring on the very same bar are acted upon)

Since, as we mentioned already, buy/sell/short/cover arrays do not carry timing information we have to somehow tell AmiBroker how to interpret such conflict. One would think that it is enough to set buyprice to open and sellprice to close to deliver timing information, but it is **NOT** the case. Price arrays themselves **DO NOT** provide timing information neither. You may ask why. This is quite simple, first of all trading prices do not need to be set to exact open/close. In several scenarios you may want to define buyprice as open + slippage and sellprice as close - slippage. Even if you do use exact open and close, it happens quite often that open is equal close (like in a doji candlestick) and then there is no way to find out from price alone, whenever it means close or open. So again **buyprice/sellprice/shortprice/coverprice variables DO NOT provide any timing information.**

The only way to control the way how same bar, same symbol entry/exit conflicts are resolved is via **AllowSameBarExit** option and **HoldMinBars** option.

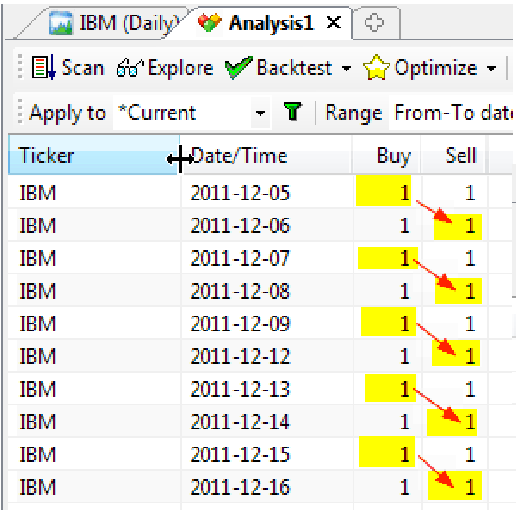
Scenario 1. Only one signal per symbol is taken at any bar

This scenario is used when **AllowSameBarExit** option is set to **False** (turned off).

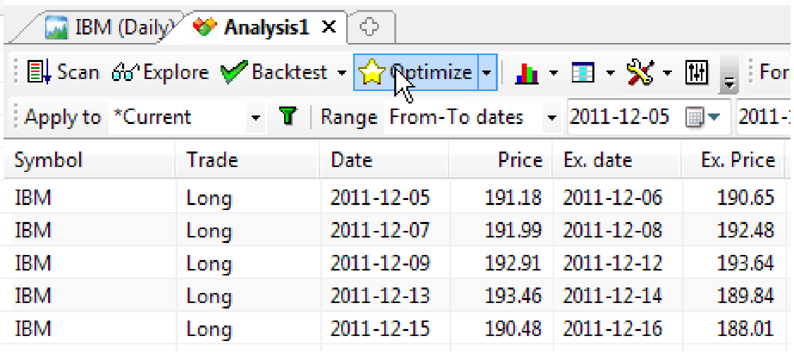
In this case it does not really matter whether exit or entry was the first within single bar. It is quite easy to understand: on any bar only one signal is acted upon. So if we are flat on given symbol, then **entry** signal is taken (with **buy signal taking precedence over short**), other signals are ignored and we move to next bar. If we are long on given symbol, then **sell** signal is taken, trade is exited and we move to next bar ignoring other signals. If we are short on given symbol then **cover** signal is taken, trade is exited and we move to next bar again ignoring other signals. If there we are in the market but there is no matching exit signal - the position is kept and we move to next bar.

```
SetOption("AllowSameBarExit", False );
Buy = 1;
Sell = 1;
```

The following pictures show which signals are taken and resulting trade list. All trades begin one day and end next day. New trade is open on the following day.



Ticker	Date/Time	Buy	Sell
IBM	2011-12-05	1	1
IBM	2011-12-06	1	1
IBM	2011-12-07	1	1
IBM	2011-12-08	1	1
IBM	2011-12-09	1	1
IBM	2011-12-12	1	1
IBM	2011-12-13	1	1
IBM	2011-12-14	1	1
IBM	2011-12-15	1	1
IBM	2011-12-16	1	1



Symbol	Trade	Date	Price	Ex. date	Ex. Price
IBM	Long	2011-12-05	191.18	2011-12-06	190.65
IBM	Long	2011-12-07	191.99	2011-12-08	192.48
IBM	Long	2011-12-09	192.91	2011-12-12	193.64
IBM	Long	2011-12-13	193.46	2011-12-14	189.84
IBM	Long	2011-12-15	190.48	2011-12-16	188.01

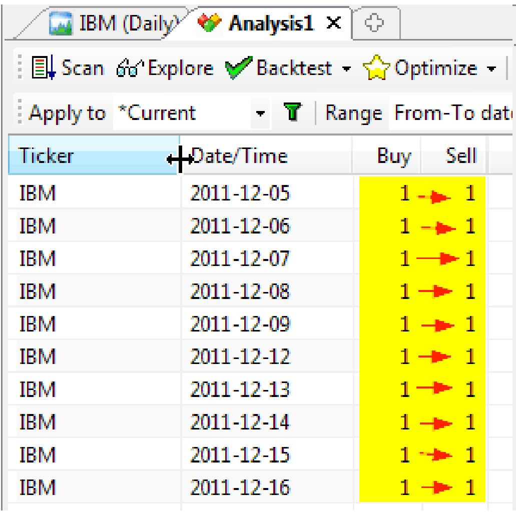
Scenario 2. Both entry and exit signals are used and **entry signal precedes exit signal**

This scenario is used when **AllowSameBarExit** option is set to **True** (turned on) and **HoldMinBars** is set to zero (which is the default setting).

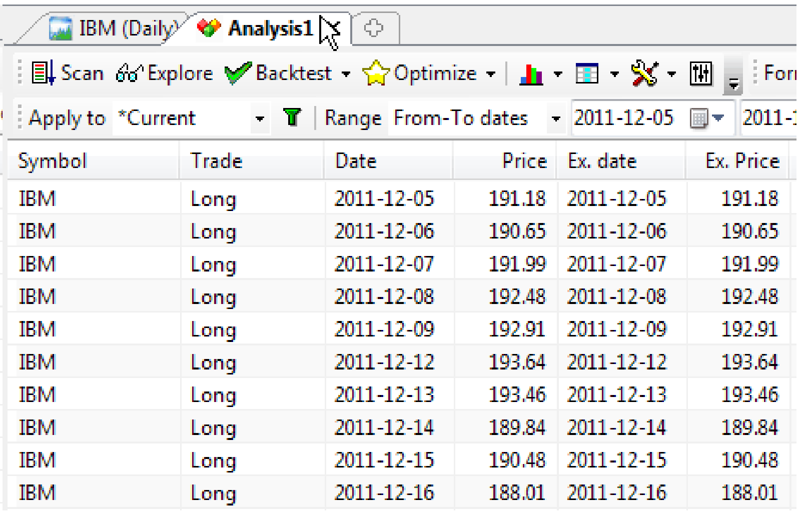
In this case we simply act on both signals immediately (same bar). So if we are flat on given symbol, then **entry** signal is taken (with **buy signal taking precedence over short**), but we do not move to the next bar immediately. Instead we check if exit signals exist too. If we are long on given symbol, then **sell** signal is taken. If we are short on given symbol then **cover** signal is taken. Only after processing all signals we move to the next bar.

```
SetOption("AllowSameBarExit", True );
Buy = 1;
Sell = 1;
```

The following pictures show which signals are taken and resulting trade list. As we can see, this time all signals are acted upon and we have sequence of single-bar trades.



Ticker	Date/Time	Buy	Sell
IBM	2011-12-05	1	1
IBM	2011-12-06	1	1
IBM	2011-12-07	1	1
IBM	2011-12-08	1	1
IBM	2011-12-09	1	1
IBM	2011-12-12	1	1
IBM	2011-12-13	1	1
IBM	2011-12-14	1	1
IBM	2011-12-15	1	1
IBM	2011-12-16	1	1



Symbol	Trade	Date	Price	Ex. date	Ex. Price
IBM	Long	2011-12-05	191.18	2011-12-05	191.18
IBM	Long	2011-12-06	190.65	2011-12-06	190.65
IBM	Long	2011-12-07	191.99	2011-12-07	191.99
IBM	Long	2011-12-08	192.48	2011-12-08	192.48
IBM	Long	2011-12-09	192.91	2011-12-09	192.91
IBM	Long	2011-12-12	193.64	2011-12-12	193.64
IBM	Long	2011-12-13	193.46	2011-12-13	193.46
IBM	Long	2011-12-14	189.84	2011-12-14	189.84
IBM	Long	2011-12-15	190.48	2011-12-15	190.48
IBM	Long	2011-12-16	188.01	2011-12-16	188.01

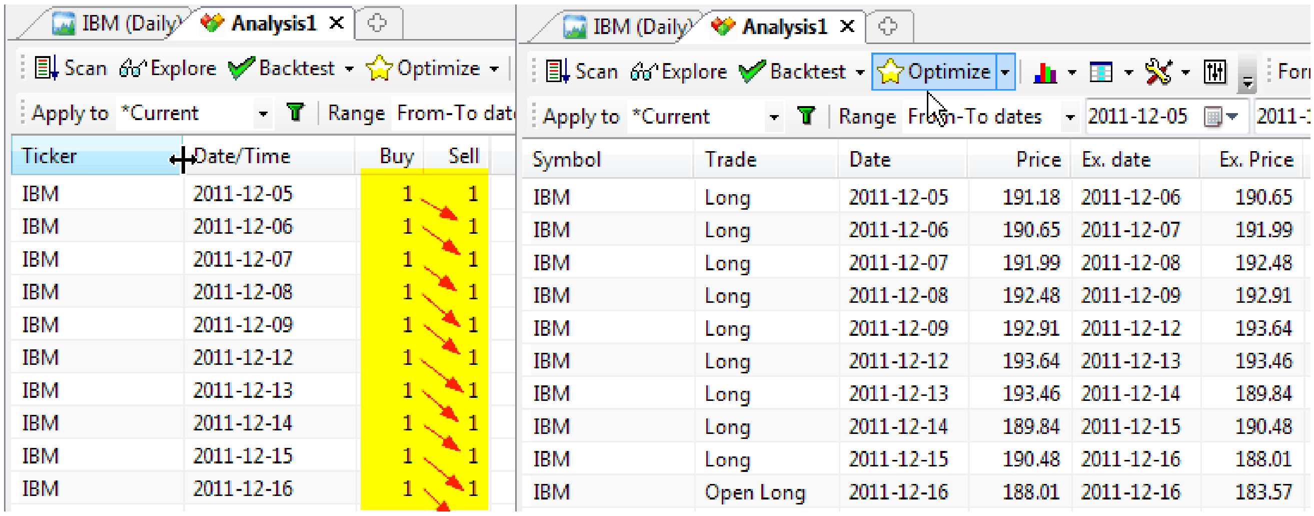
Scenario 3. Both signals are used and **entry signal comes after exit signal.**

This scenario is used when **AllowSameBarExit** option is set to **True** (turned on) and **HoldMinBars** is set to 1 (or more).

In this case we simply act on both signals in single bar, but we respect the HoldMinBars = 1 limitation, so trade that was just open can not be closed the same bar. So if we are long on given symbol, then **sell** signal is taken. If we are short on given symbol then **cover** signal is taken. We don't move to next bar yet. Now if we are flat on given symbol (possibly just exited position on this bar exit signal), then **entry** signal is taken if any (with **buy signal taking precedence over short**) and then we move to the next bar.

```
SetOption("AllowSameBarExit", True );
SetOption("HoldMinBars", 1 );
Buy=1;
Sell=1;
```

The following pictures show which signals are taken and resulting trade list. As we can see, again all signals are acted upon BUT... trade duration is longer - they are **not** same bar trades - they all span overnight.



Ticker	Date/Time	Buy	Sell
IBM	2011-12-05	1	1
IBM	2011-12-06	1	1
IBM	2011-12-07	1	1
IBM	2011-12-08	1	1
IBM	2011-12-09	1	1
IBM	2011-12-12	1	1
IBM	2011-12-13	1	1
IBM	2011-12-14	1	1
IBM	2011-12-15	1	1
IBM	2011-12-16	1	1

Symbol	Trade	Date	Price	Ex. date	Ex. Price
IBM	Long	2011-12-05	191.18	2011-12-06	190.65
IBM	Long	2011-12-06	190.65	2011-12-07	191.99
IBM	Long	2011-12-07	191.99	2011-12-08	192.48
IBM	Long	2011-12-08	192.48	2011-12-09	192.91
IBM	Long	2011-12-09	192.91	2011-12-12	193.64
IBM	Long	2011-12-12	193.64	2011-12-13	193.46
IBM	Long	2011-12-13	193.46	2011-12-14	189.84
IBM	Long	2011-12-14	189.84	2011-12-15	190.48
IBM	Long	2011-12-15	190.48	2011-12-16	188.01
IBM	Open Long	2011-12-16	188.01	2011-12-16	183.57

How does it work in portfolio case?

The mechanism is the same regardless if you test on single symbol or multiple symbols. First same-bar conflicts are resolved on every symbol separately the way described above. Then, when you test on multiple symbols, resulting trade candidates are subject to scoring by PositionScore described in earlier part of this document.

Support for market-neutral, long-short balanced strategies

An investment strategy is considered market neutral if it seeks to entirely avoid some form of market risk, typically by hedging. The strategy holds Long / short equity positions, with long positions hedged with short positions in the same and related sectors, so that the equity market neutral investor should be little affected by sector- or market-wide events. This places, in essence, a bet that the long positions will outperform their sectors (or the short positions will underperform) regardless of the strength of the sectors.

In version 5.20 the following backtester options have been added to simplify implementing market-neutral systems: **SeparateLongShortRank**, **MaxOpenLong**, **MaxOpenShort.**

SeparateLongShortRank backtester option

To enable separate long/short ranking use:
SetOption("SeparateLongShortRank", True);

When separate long/short ranking is enabled, the backtester maintains TWO separate "top-ranked" signal lists, one for long signals and one for short signals. This ensures that long and short candidates are independently even if position score is not symetrical (for example when long candidates have very high positive scores while short candidates have only fractional negative scores). That contrasts with the default mode where only absolute value of position score matters, therefore one side (long/short) may completely dominate ranking if score values are asymetrical.

When SeparateLongShortRank is enabled, in the second phase of backtest, two separate ranking lists are interleaved to form final signal list by first taking top ranked long, then top ranked short, then 2nd top ranked long, then 2nd top ranked short, then 3rd top ranked long and 3rd top ranked short, and so on... (as long as signals exist in BOTH long/short lists, if there is no more signals of given kind, then remaining signals from either long or short lists are appended)

For example:
Entry signals(score):ESRX=Buy(60.93), GILD=Short(-47.56), CELG=Buy(57.68), MRVL=Short(-10.75), ADBE=Buy(34.75), VRTX=Buy(15.55), SIRI=Buy(2.79),

As you can see Short signals get interleaved between Long signals even though their absolute values of scores are smaller than corresponding scores of long signals. Also there were only 2 short signals for that particular bar so, the rest of the list shows long signals in order of position score. Although this feature can be used independently, it is intended to be used in combination with MaxOpenLong and MaxOpenShort options.

MaxOpenLong / MaxOpenShort backtester options

MaxOpenLong - limits the number of LONG positions that can be open simultaneously
MaxOpenShort - limits the number of SHORT positions that can be open simultaneously

Example:
SetOption("MaxOpenPositions", 15);
SetOption("MaxOpenLong", 11);
SetOption("MaxOpenShort", 7);

The value of ZERO (default) means NO LIMIT. If both MaxOpenLong and MaxOpenShort are set to zero (or not defined at all) the backtester works old way - there is only global limit active (MaxOpenPositions) regardless of type of trade.

Note that these limits are independent from global limit (MaxOpenPositions). This means that MaxOpenLong + MaxOpenShort may or may not be equal to MaxOpenPositions.

If MaxOpenLong + MaxOpenShort is greater than MaxOpenPositions then total number of positions allowed will not exceed MaxOpenPositions, and individual long/short limits will apply too. For example if your system MaxOpenLong is set to 7 and maxOpenShort is set to 7 and MaxOpenPositions is set to 10 and your system generated 20 signals: 9 long (highest ranked) and 11 short, it will open 7 long and 3 shorts.

If MaxOpenLong + MaxOpenShort is smaller than MaxOpenPositions (but greater than zero), the system won't be able to open more than (MaxOpenLong+MaxOpenShort).

Please also note that MaxOpenLong and MaxOpenShort only cap the number of open positions of given type (long/short). They do NOT affect the way ranking is made. I.e. by default ranking is performed using ABSOLUTE value of positionscore.

If your position score is NOT symetrical, this may mean that you are not getting desired top-ranked signals from one side. Therefore, to fully utilise MaxOpenLong and MaxOpenShort in rotational balanced ("market neutral") long/short systems it is desired to perform SEPARATE ranking for long signals and short signals. To enable separate long/short ranking use:

SetOption("SeparateLongShortRank", True);

See Also:

Backtesting your trading ideas article.

Backtesting systems for futures contracts article.

Using AFL editor section of the guide.

Insider guide to backtester (newsletter 1/2002)

Reading backtest report

To view the report of last backest simply click **Report** button in the automatic analysis window. To view results of ALL past backtest, click drop down arrow on the **Report** button and choose **Report Explorer** option. This will display the Report Explorer window that will show the list of all backtests performed. If you double click on the line - detailed report will be shown.

New report is hugely enhanced compared to old one. It includes separate statistics for all, long and short sides as well as large number of new metrics. You can get short help on given figure by hovering your mouse over given field name. You will see the description in the tooltip. Short explanations are provided also below:

Exposure % - 'Market exposure of the trading system calculated on bar by bar basis. Sum of bar exposures divided by number of bars. Single bar exposure is the value of open positions divided by portfolio equity.

Net Risk Adjusted Return % - Net profit % divided by Exposure %

Annual Return % - Compounded Annual Return % (CAR)

Risk Adjusted Return % - Annual return % divided by Exposure %

Avg. Profit/Loss, also known as **Expectancy ($) -** (Profit of winners + Loss of losers)/(number of trades), represents expected dollar gain/loss per trade

Avg. Profit/Loss %, also known as **Expectancy (%)** - '(% Profit of winners + % Loss of losers)/(number of trades), represents expected percent gain/loss per trade

Avg. Bars Held - sum of bars in trades / number of trades

Max. trade drawdown - The largest peak to valley decline experienced in any single trade. The lower the better

Max. trade % drawdown - The largest peak to valley percentage decline experienced in any single trade. The lower the better

Max. system drawdown - The largest peak to valley decline experienced in portfolio equity. The lower the better

Max. system % drawdown - The largest peak to valley percentage decline experienced in portfolio equity. The lower the better

Recovery Factor - Net profit divided by Max. system drawdown

CAR/MaxDD - Compound Annual % Return divided by Max. system % drawdown. Good if bigger than 2

RAR/MaxDD - Risk Adjusted Return divided by Max. system % drawdown. Good if bigger than 2.

Profit Factor - Profit of winners divided by loss of losers

Payoff Ratio - Ratio average win / average loss

Standard Error - Standard error measures chopiness of equity line. The lower the better.

Risk-Reward Ratio - Measure of the relation between the risk inherent in a trading the system compared to its potential gain. Higher is better. Calculated as slope of equity line (expected annual return) divided by its standard error.

Ulcer Index - Square root of sum of squared drawdowns divided by number of bars

Ulcer Performance Index - (Annual profit - Tresury notes profit)/Ulcer Index'>Ulcer Performance Index. Currently tresury notes profit is hardcoded at 5.4. In future version there will be user-setting for this.

Sharpe Ratio of trades - Measure of risk adjusted return of investment. Above 1.0 is good, more than 2.0 is very good. More information http://www.stanford.edu/~wfsharpe/art/sr/sr.htm . Calculation: first average percentage return and standard deviation of returns is calculated. Then these two figures are annualized by multipling them by ratio (NumberOfBarsPerYear)/(AvgNumberOfBarsPerTrade). Then the risk free rate of return is subtracted (currently hard-coded 5) from annualized average return and then divided by annualized standard deviation of returns.

K-Ratio - Detects inconsistency in returns. Should be 1.0 or more. The higher K ratio is the more consistent return you may expect from the system. Linear regression slope of equity line multiplied by square root of sum of squared deviations of bar number divided by standard error of equity line multiplied by square root of number of bars. More information: Stocks & Commodities V14:3 (115-118): Measuring System Performance by Lars N. Kestner

Color-coding in the backtest report (new in 5.60)

Version 5.60 brings enhanced backtest report: color-coding 'good' and 'bad' values in backtest report. Some of the metrics in the backtest report are color-coded. Blue means "neutral", Green means "good", Red means "bad". Metrics that are not colorized are always black.
This color coding is of course arbitrary and should be used as guideance only. Treat 'red' as a warning flag and advice to check the value in detail.

As of now the following metrics are colorized:
Net Profit, Net Profit % - bad < 0, good > 0
Annual Profit %, bad < 0, neutral betwen 0 and 10, good > 10
RAR % bad < 0, good > (10 / Exposure)
Avg. Profit/Loss all trades (Expectancy $) - bad < 0, good > 0
Avg Profit/Loss % all trades (Expectancy %) - bad < 0, good > 0
Max. system % drawdown - bad: dd worse than -30%, neutral: dd between -30 and -10%, good - -10% to 0%
CAR/MaxDD, RAR/MaxDD - bad < 1, neutral between 1 and 2, good > 2
Recovery factor - bad < 1, neutral between 1 and 2, good > 2
Payoff ratio - bad < 1, neutral between 1 and 2, good > 2

See Also:

Old backtest report

Backtesting your trading ideas article.

Portfolio Backtesting article.

Backtesting systems for futures contracts article.

Using AFL editor section of the guide.

Insider guide to backtester (newsletter 1/2002)

How to optimize trading system

NOTE: This is fairly advanced topic. Please read previous AFL tutorials first.

Introduction

The idea behind an optimization is simple. First you have to have a trading system, this may be a simple moving average crossover for example. In almost every system there are some parameters (as averaging period) that decide how given system behaves (i.e. is is well suited for long term or short term, how does is react on highly volatile stocks, etc). The optimization is the process of finding optimal values of those parameters (giving highest profit from the system) for a given symbol (or a portfolio of symbols). AmiBroker is one of the very few programs that allow you to optimize your system on multiple symbols at once.

To optimize your system you have to define from one upto 64 parameters to be optimized. You decide what is a minimum and maximum allowable value of the parameter and in what increments this value should be updated. AmiBroker then performs multiple back tests the system using ALL possible combinations of parameters values. When this process is finished AmiBroker displays the list of results sorted by net profit. You are able to see the values of optimization parameters that give the best result.

Writing AFL formula

Optimization in back tester is supported via new function called optimize. The syntax of this function is as follows:

variable = optimize("*Description*", *default*, *min*, *max*, *step*);

where:

variable **-** is normal AFL variable that gets assigned the value returned by optimize function.
With normal backtesting, scanning, exploration and comentary modes the optimize function returns *default* value, so the above function call is equivalent to: variable = *default*;

In optimization mode optimize function returns successive values from *min* to *max* (inclusively) with *step* stepping.

"*Description"* is a string that is used to identify the optimization variable and is displayed as a column name in the optimization result list.

default is a default value that optimize function returns in exploration, indicator, commentary, scan and normal back test modes

min is a minimum value of the variable being optimized

max is a maximum value of the variable being optimized

step is an interval used for increasing the value from *min* to *max*

Notes:

- AmiBroker supports upto 64 calls to optimize function (therefore upto 64 optimization variables), note that if you are using exhaustive optimization then it is really good idea to limit number of optimization

variables to just few.
- Each call to optimize generate *(max - min)/step* optimization loops and multiple calls to optimize multiply the number of runs needed. For example optimizing two parameters using 10 steps will require 10*10 = 100 optimization loops.
- Call optimize function only ONCE per variable at the beginning of your formula as each call generates a new optimization loops
- Multiple-symbol optimization is fully supported by AmiBroker
- Maximum search space is 2^{64} (10^{19} = 10,000,000,000,000,000,000) combinations

Examples

1. Single variable optimization:

```
sigavg = Optimize( "Signal average", 9, 2, 20, 1 );

Buy = Cross( MACD( 12, 26 ), Signal( 12, 26, sigavg ) );
Sell = Cross( Signal( 12, 26, sigavg ), MACD( 12, 26 ) );
```

2. Two-variable optimization (suitable for 3D charting)

```
per = Optimize("per", 2, 5, 50, 1 );
Level = Optimize("level", 2, 2, 150, 4 );

Buy=Cross( CCI(per), -Level );
Sell = Cross( Level, CCI(per) );
```

3. Multiple (3) variable optimization:

```
mfast = Optimize( "MACD Fast", 12, 8, 16, 1 );
mslow = Optimize("MACD Slow", 26, 17, 30, 1 );
sigavg = Optimize( "Signal average", 9, 2, 20, 1 );


Buy = Cross( MACD( mfast, mslow ) , Signal( mfast, mslow, sigavg ) );
Sell = Cross( Signal( mfast, mslow, sigavg ), MACD( mfast, mslow ) );
```

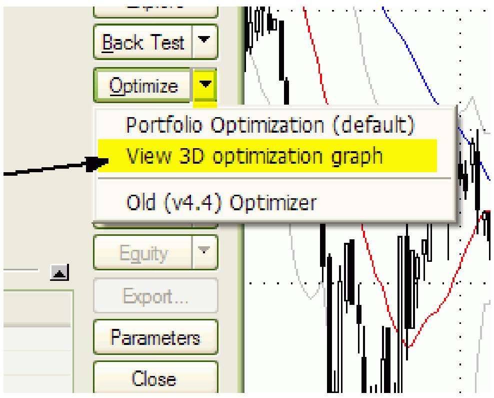
After entering the formula just click on **Optimize** button in "Automatic Analysis" window. AmiBroker will start testing all possible combinations of optimization variables and report the results in the list. After optimization is done the list of result is presented sorted by the Net % profit. As you can sort the results by any column in the result list it is easy to get the optimal values of parameters for the lowest drawdown, lowest number of trades, largest profit factor, lowest market exposure and highest risk adjusted annual % return. The last columns of result list present the values of optimization variables for given test.

When you decide which combination of parameters suits your needs the best all you need to do is to replace the default values in optimize function calls with the optimal values. At current stage you need to type them by hand in the formula edit window (the second parameter of optimize function call).

Displaying 3D animated optimization charts

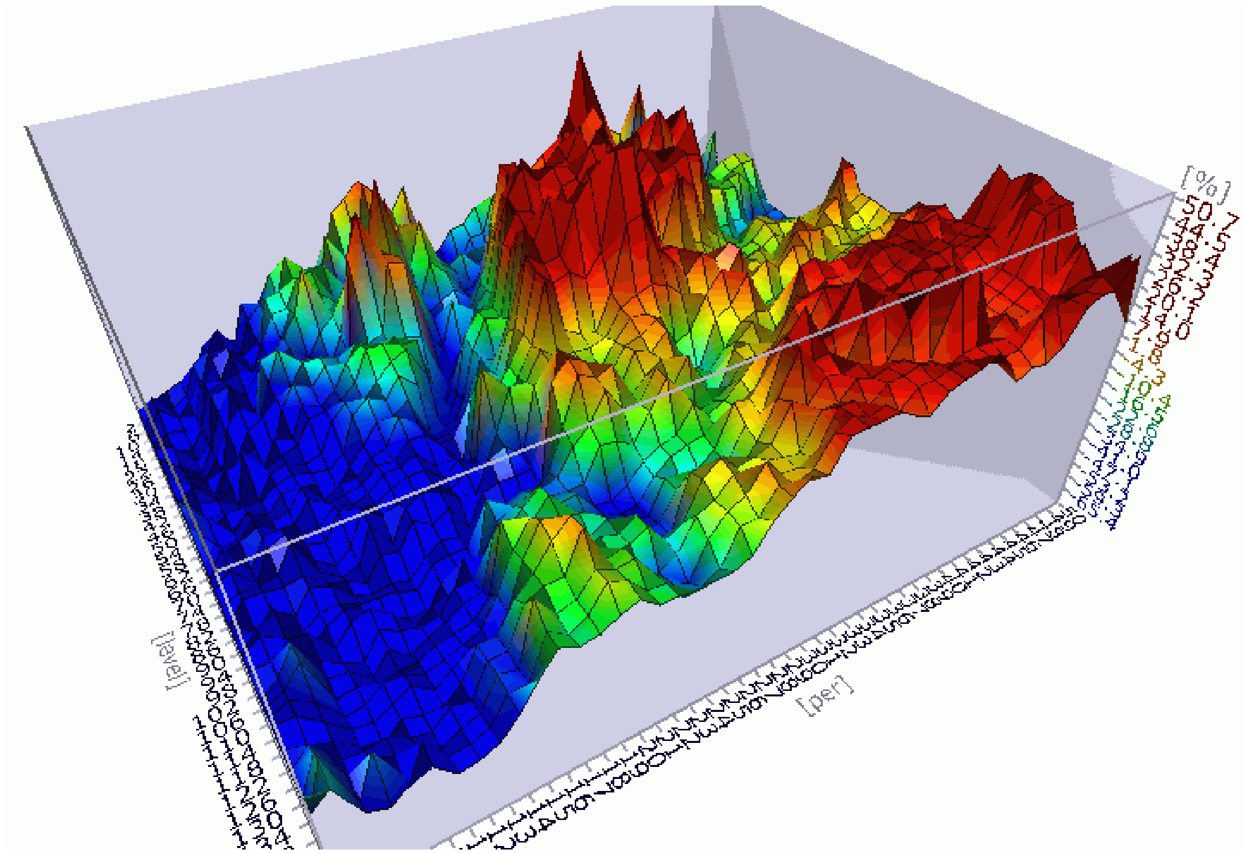
To display 3D optimization chart, you need to run two-variable optimization first. Two variable optimization needs a formula that has 2 Optimize() function calls. An example two-variable optimization formula looks like this:

```
per = Optimize("per", 2, 5, 50, 1 );
Level = Optimize("level", 2, 2, 150, 4 );

Buy=Cross( CCI(per), -Level );
Sell = Cross( Level, CCI(per) );
```



After entering the formula you need to click "Optimize" button.

Once optimization is complete you should click on the drop down arrow on **Optimize** button and choose **View 3D optimization graph**. In a few seconds a colorful three-dimensional surface plot will appear in a 3D chart viewer window. An example 3D chart generated using above formula is shown below.



By default the 3D charts display values of Net profit against optimization variables. You can however plot 3D surface chart for any column in the optimization result table. Just click on the column header to sort it (blue arrow will appear indicating that optimization results are sorted by selected column) and then choose **View 3D optimization graph** again.

By visualizing how your system's parameters affect trading performance, you can more readily decide which parameter values produce "fragile" and which produce "robust" system performance. Robust settings are regions in the 3D graph that show gradual rather than abrupt changes in the surface plot. 3D optimization charts are great tool to prevent curve-fitting. Curve-fitting (or over-optimization) occurs when the system is more complex than it needs to be, and all that complexity was focused on market conditions that may never happen again. Radical changes (or spikes) in the 3D optimization charts show clearly over-optimization areas.

You should choose parameter region that produces a broad and wide plateau on 3D chart for your real life trading. Parameter sets producing profit spikes will not work reliably in real trading.

3D chart viewer controls

AmiBroker's 3D chart viewer offers total viewing capabilities with full graph rotation and animation. Now you can view your system results from every conceivable perspective. You can control the position and other parameters of the chart using the mouse, toolbar and keyboard shortcuts, whatever you find easier for you. Below you will find the list.

Mouse controls:

- to Rotate - hold down LEFT mouse button and move in X/Y directions
- to Zoom-in, zoom-out - hold down RIGHT mouse button and move in X/Y directions
- to Move (translate) - hold down LEFT mouse button and CTRL key and move in X/Y directions
- to Animate - hold down LEFT mouse button, drag quickly and release button while dragging

Keyboard controls:

SPACE - animate (auto-rotate)
LEFT ARROW KEY - rotate vert. left
RIGHT ARROW KEY - rotate vert. right
UP ARROW KEY - rotate horiz. up
DOWN ARROW KEY - rotate horiz. down
NUMPAD + (PLUS) - Near (zoom in)
NUMPAD - (MINUS) - Far (zoom out)
NUMPAD 4 - move left
NUMPAD 6 - move right
NUMPAD 8 - move up
NUMPAD 2 - move down
PAGE UP - water level up
PAGE DOWN - water level down

Smart (non-exhaustive) optimization

Introduction

AmiBroker now offers smart (non-exhaustive) optimization in addition to regular, exhaustive search.
Non-exhaustive search is useful if number of all parameter combinations of given trading system is simply too large to be feasible for exhaustive search.

Exhaustive search is perfectly fine as long as it is reasonable to use it. Let's say you have 2 parameters each ranging from 1 to 100 (step 1).
That's 10000 combinations - perfectly OK for exhaustive search. Now with 3 parameters you got 1 million combinations - it is still OK for exhaustive search (but can be lenghty). With 4 parameters you have 100 million combinations and with 5 parameters (1..100) you have 10 billion combinations. In that case it would be too time consuming to check all of them, and this is the area where non-exhaustive smart-search methods can solve the problem that is not solvable in reasonable time using exhaustive search.

Quick Start

Here is absolutely the SIMPLEST instruction how to use new non-exhaustive optimizer (in this case CMA-ES).

1. Open your formula in the Formula Editor

2. Add this single line at the top of your formula:

```
OptimizerSetEngine("cmae"); // you can also use "spso" or "trib" here
```

3. (Optional) Select your optimization target in Automatic Analysis, Settings, "Walk-Forward" tab, **Optimization target** field. If you skip this step it will optimize for CAR/MDD (compound annual return divided by maximum % drawdown).

and... that's it.

Now if you run optimization using this formula, it will use new evolutionary (non-exhaustive) CMA-ES optimizer.

How does it work ?

The optimization is the process of finding minimum (or maximum) of given function. Any trading system can be considered as a function of certain number of arguments. The inputs are parameters and quotation data , the output is your optimization target
(say CAR/MDD). And you are looking for maximum of given function.

Some of smart optimization algorithms are based on nature (animal behavior) - PSO algorithm, or biological process - Genetic algorithms,
and some are based on mathematical concepts derived by humans - CMA-ES.

These algorithms are used in many different areas, including finance. Enter "PSO finance" or "CMA-ES finance" in Google and you will find lots of info.

Non-exhaustive (or "smart") methods will find global or local optimum. The goal is of course to find global one, but if there is a single sharp peak
out of zillions parameter combinations, non-exhaustive methods may fail to find this single peak, but taking it form trader's perspecive, finding single sharp peak is useless for trading because that result would be instable (too fragile) and not replicable in real trading. In optimization process we are rather looking for plateau regions with stable parameters and this is the area where intelligent methods shine.

As to algorithm used by non-exhaustive search it looks as follows:

a) the optimizer generates some (usually random) starting population of parameter sets
b) backtest is performed by AmiBroker for each parameter set from the population
c) the results of backtests are evaluated according to the logic of algorithm
and new population is generated based on the evolution of results,
d) if new best is found - save it and go to step b) until stop criteria are met

Example stop criteria can include:
a) reaching specified maximum iterations
b) stop if the range of best objective values of last X generations is zero
c) stop if adding 0.1 standard deviation vector in any principal axis direction does not change the value of objective value

d) others

To use any smart (non-exhaustive) optimizer in AmiBroker you need to specify the optimizer engine you want to use in the AFL formula using OptimizerSetEngine function.

OptimizerSetEngine("name")

The function selects external optimization engine defined by *name*. AmiBroker currently ships with 3 engines: Standard Particle Swarm Optimizer ("spso"), Tribes ("trib"), and CMA-ES ("cmae") - the names in braces are to be used in OptimizerSetEngine calls.

In addition to selecting optimizer engine you may want to set some of its internal parameters. To do so use OptimizerSetOption function.

OptimizerSetOption("name", value) function

The function set additional parameters for external optimization engine. The parameters are engine-dependent.
All three optimizers shipped with AmiBroker (SPSO, Trib, CMAE) support two parameters: "Runs" (number of runs) and "MaxEval" (maximum evaluations (tests)per single run). The behaviour of each parameter is engine-dependent, so same values may and usually will yield different results with different engines used.

The difference between Runs and MaxEval is as follows. Evaluation (or test) is single backtest (or evaluation of objective function value).
RUN is one full run of the algorithm (finding optimum value) - usually involving many tests (evaluations).

Each run simply RESTARTS the entire optimization process from the new beginning (new initial random population).
Therefore each run may lead to finding different local max/min (if it does not find global one). So Runs parameter defines number of subsequent algorithm runs. MaxEval is the maximum number of evaluations (bactests) in any single run.

If the problem is relatively simple and 1000 tests are enough to find global max, 5x1000 is more likely to find global maximum
because there are less chances to be stuck in local max, as subsequent runs will start from different initial random population

Choosing parameter values can be tricky. It depends on problem under test, its complexity, etc, etc.
Any stochastic non-exhaustive method does not give you guarantee of finding global max/min, regardless of number of tests if it is smaller
than exhaustive. The easiest answer is to : specify as large number of tests as it is reasonable for you in terms of time required to complete.
Another simple advice is to multiply by 10 the number of tests with adding new dimension. That may lead to overestimating number
of tests required, but it is quite safe. Shipped engines are designed to be simple to use, therefore "reasonable" default/automatic values are used so optimization can be usually run without specifying anything (accepting defaults).

Caveat

It is important to understand that all smart optimization methods work best in continuous parameter spaces and relatively smooth objective functions. If parameter space is discrete evolutionary algorithms may have trouble finding optimum value. It is especially true for binary (on/off) parameters - they are not suited for any search method that uses gradient of objective function change (as most smart methods do). If your trading system contains many binary parameters, you should not use smart optimizer directly on them. Instead try to optimize only continuous parameters using smart optimizer, and switch binary parameters manually or via external script.

SPSO - Standard Particle Swarm Optimizer

Standard Particle Swarm Optimizer is based on SPSO2007 code that is supposed to produce good results provided that correct parameters (i.e. Runs, MaxEval) are provided for particular problem.
Picking correct options for the PSO optimizer can be tricky therefore results may significantly vary from case to case.

SPSO.dll comes with full source codes inside "ADK" subfolder.

Example code for Standard Particle Swarm Optimizer:
(finding optimum value in 1000 tests within search space of 10000 combinations)

```
OptimizerSetEngine("spso");
OptimizerSetOption("Runs", 1 );
OptimizerSetOption("MaxEval", 1000 );

sl = Optimize("s", 26, 1, 100, 1 );
fa = Optimize("f", 12, 1, 100, 1 );

Buy = Cross( MACD( fa, sl ), 0 );
Sell = Cross( 0, MACD( fa, sl ) );
```

TRIBES - Adaptive Parameter-less Particle Swarm Optimizer

Tribes is adaptive, parameter-less version of PSO (particle swarm optimization) non-exhaustive optimizer. For scientific background see:
http://www.particleswarm.info/Tribes_2006_Cooren.pdf

In theory it should perform better than regular PSO, because it can automatically adjust the swarm sizes and algorithm strategy to the problem being solved.

Practice shows that its performance is quite similar to PSO.

The Tribes.DLL plugin implements "Tribes-D" (i.e. dimensionless) variant. Based on http://clerc.maurice.free.fr/pso/Tribes/TRIBES-D.zip by Maurice Clerc. Original source codes used with permission from the author

Tribes.DLL comes with full source code (inside "ADK" folder)

Supported parameters:
"MaxEval" - maximum number of evaluations (backtests) per run (default = 1000).

OptimizerSetOption("MaxEval", 1000);

You should increase the number of evaluations with increasing number of dimensions (number of optimization params).
The default 1000 is good for 2 or maximum 3 dimensions.

"Runs" - number of runs (restarts). (default = 5)
You can leave the number of runs at default value of 5.

By default number of runs (or restarts) is set to 5.

To use Tribes optimizer, you just need to add one line to your code:

OptimizerSetEngine("trib");

OptimizerSetOption("MaxEval", 5000); // 5000 evaluations max

CMA-ES - Covariance Matrix Adaptation Evolutionary Strategy optimizer

CMA-ES (Covariance Matrix Adaptation Evolutionary Strategy) is advanced non-exhaustive optimizer.
For scientific background see:
http://www.bionik.tu-berlin.de/user/niko/cmaesintro.html
According to scientific benchmarks outperforms nine other, most popular evolutionary strategies (like PSO, Genetic and Differential evolution).
http://www.bionik.tu-berlin.de/user/niko/cec2005.html

The CMAE.DLL plugin implements "Global" variant of search with several restarts with increasing population size
CMAE.DLL comes with full source code (inside "ADK" folder)

By default number of runs (or restarts) is set to 5.
It is advised to leave the default number of restarts.

You may vary it using OptimizerSetOption("Runs", N) call, where N should be in range 1..10.
Specifying more than 10 runs is not recommended, although possible.
Note that each run uses TWICE the size of population of previous run so it grows exponentially.
Therefore with 10 runs you end up with population 2^10 greater (1024 times) than the first run.

There is another parameter "MaxEval". The default value is ZERO which means that plugin will automatically calculate MaxEval required. It is advised to NOT to define MaxEval by yourself as default works fine.

The algorithm is smart enough to minimize the number of evaluations required and it converges very fast to solution point, so often it finds solutions faster than other strategies.

It is normal that the plugin will skip some evaluations steps, if it detects that solution was found, therefore you should not be surprised that optimization progress bar may move very fast at some points. The plugin also has ability to increase number of steps over initially estimated value if it is needed to find the solution. Due to its adaptive nature, the "estimated time left" and/or "number of steps" displayed by the progress dialog is only "best guess at the time" and may vary during optimization course.

To use CMA-ES optimizer, you just need to add one line to your code:

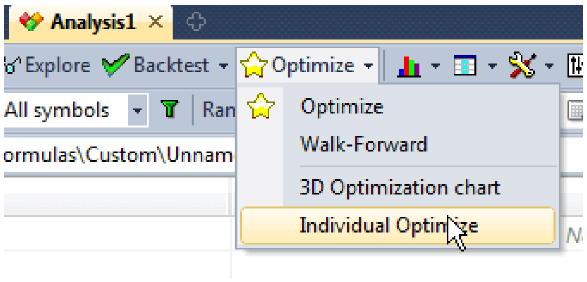
```
OptimizerSetEngine("cmae");
```

This will run the optimization with default settings which are fine for most cases.

It should be noted, as it is the case with many continouos-space search algorithms, that decreasing "step" parameter in Optimize() funciton calls does not significantly affect optimization times. The only thing that matters is the problem "dimension", i.e. the number of different parameters (number of optimize function calls). The number of "steps" per parameter can be set without affecting the optimization time, so use the finest resolution you want. In theory the algorithm should be able to find solution in at most 900*(N+3)*(N+3) backtests where "N" is the dimension. In practice it converges a LOT faster. For example the solution in 3 (N=3) dimensional parameter space (say 100*100*100 = 1 million exhaustive steps) can be found in as few as 500-900 CMA-ES steps.

Multi-threaded individual optimization

Starting from AmiBroker 5.70 in addition to multiple-symbol multithreading, you can perform multi-threaded single-symbol optimization. To access this functionality, click on drop down arrow next to "Optimize" button in the New Analysis window and select "**Individual Optimize**".



"Individual Optimize" will use all available processor cores to perform single-symbol optimization, making it much faster than regular optimization.

In "Current symbol" mode it will perform optimization on one symbol. In "All symbols" and "Filter" modes it will process all symbols sequentially, i.e. first complete optimization for first symbol, then optimization on second symbol, etc.

Limitations:
1. Custom backtester is NOT supported (yet)
2. Smart optimization engines are NOT supported - only EXHAUSTIVE optimization works.

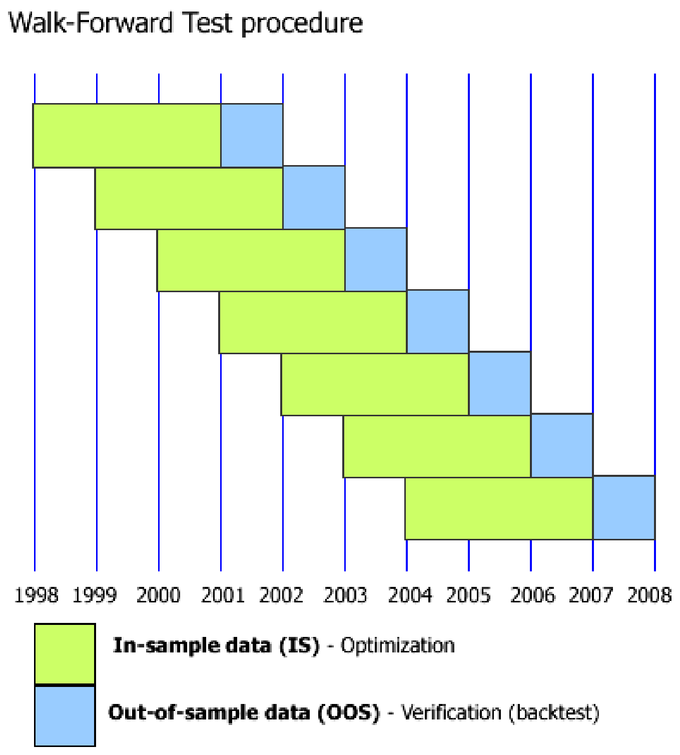
For explanation of these limitations see Tutorial: Efficient use of multi-threading.

Eventually we may get rid of limitation (1) - when AmiBroker is changed so custom backtester does not use OLE anymore. But (2) is probably here to stay for long.

Walk-forward testing

AmiBroker 5.10 features the automatic Walk-Forward test mode.

The automatic Walk forward test is a system design and validation technique in which you optimize the parameter values on a past segment of market data (”in-sample”), then verify the performance of the system by testing it forward in time on data following the optimization segment (”out-of-sample”). You evaluate the system based on how well it performs on the test data (”out-of-sample”), not the data it was optimized on. The process can be repeated over subsequent time segments. The following illustration shows how the process works.

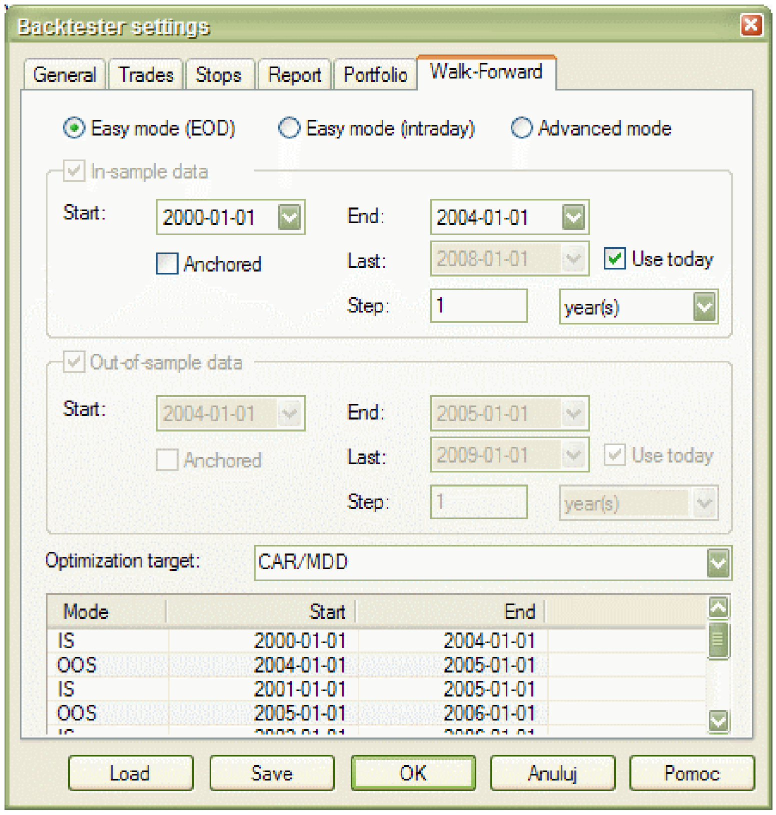

Walk-Forward Test procedure
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
In-sample data (IS) - Optimization
Out-of-sample data (OOS) - Verification (backtest)

The purpose of walk-forward test is to determine whenever the performance of optimized trading system is the realistic or the result of curve-fitting. The performance of the system can be considered realistic if it has predicitive value and performs good on unseen (out-of-sample) market data. When the system is properly designed, the real-time trading performance should be in relation to that uncovered during optimization. If the system is going to work in real trading, it must first pass a walk-forward test. In other words, we don't really care about in-sample results as they are (or should be) always good. What matters is out-of-sample system performance. It is the realistic estimate of how the system would work in real trading and will quickly reveal any curve-fitting issues. If out-of-sample performance is poor then you should not trade such a system.

The premise of performing several optimization/tests steps over time is that the recent past is a better foundation for selecting system parameter values than the distant past. We hope is that the parameter values chosen on the optimization segment will be well suited to the market conditions that immediately follow. This may or may not be the case as markets goes through bear/bull cycle, so care should be taken when choosing the length of in-sample period. For more information about system design and verification using walk-forward procedure and all issues involved, we can recommend Howard Bandy's book: "Quantitative Trading Systems" (see links on AmiBroker page).

To use Walk-Forward optimization please follow these steps:

1. Goto **Tools->Automatic Analysis**
2. Click **Settings** button, then switch to **Walk-Forward tab**


Backtester settings

General | Trades | Stops | Report | Portfolio | Walk-Forward

[● Easy mode (EOD)] ○ Easy mode (intraday) ○ Advanced mode

☑ In-sample data

Start:	2000-01-01	End:	2004-01-01
☐ Anchored		Last:	2008-01-01 ☑ Use today
		Step:	1 year(s)

☑ Out-of-sample data

Start:	2004-01-01	End:	2005-01-01
☐ Anchored		Last:	2009-01-01 ☑ Use today
		Step:	1 year(s)

Optimization target: CAR/MDD

Mode	Start	End
IS	2000-01-01	2004-01-01
OOS	2004-01-01	2005-01-01
IS	2001-01-01	2005-01-01
OOS	2005-01-01	2006-01-01

3. Here you can see Walk forward settings for In-sample optimization, out-of-sample backtest

   Start and **End** dates mark initial period begin / end
   This period will be moved forward by **Step** until the **End** reaches the **Last** date.

   The **Start** date can move forward by **step** too, or can be anchored (constant) if **Anchored** check is on.

   If you mark **Use today** then **Last** date entered will be ignored and TODAY (current date) will be used instead.
   By default an “EASY MODE” is selected which simplifies the process of setting up WF parameters.

   It assumes that:
   a) Out-of-sample segment immediatelly follows in-sample segment
   b) the length of out-of-sample segment equals to the walk-forward step

   Based on these two assumptions the “EASY” mode takes in-sample END date and sets out-of-sample START date to the following day. Then adds in-sample STEP and this becomes out-of-sample END date.

   In-sample and Out-of-sample step values are set to the same values. The “EASY” mode guarantees correctness of WF procedure settings.

You should use **Easy mode (EOD)** when testing on end-of-day data or **Easy mode (Intraday)** when testing on intraday data. The difference is that in EOD mode the END date of previous period and START date of next period are the same - thus avoiding gap
between periods. Intraday mode set START date of the next period as NEXT DAY after END of previous period. That guarantees
that boundary day is not counted twice when testing on intraday data.

In the **Advanced mode**, the user has complete control over all values, to the extent that they may not constitute valid WF procedure.
The interface allows to selectivelly disable in-sample and out-of-sample phases using checkboxes at top (for special things like running sequential backtests without optimization).
All settings are immediatelly reflected in the PREVIEW list that shows all generated IS/OOS segments and their dates.

4. The "**Optimization target**" field defines the optimization raport COLUMN NAME that
   will be used for sorting results and finding the BEST one. Any built-in column can be used
   (as appears in the optimization output), or you can use any custom metric that you define
   in custom backtester. The default is CAR/MDD, you can however select any other built-in metric from the combo.
   You can also TYPE-IN any custom metric that you have added via custom backtester interface.
5. Once you defined Walk-Forward settings, please go to Automatic Analysis and
6. press the dropdown ARROW on the Optimize button and select "Walk Forward Optimization"

   This will run sequence of optimizaitons and backtest and the results will be displayed in the "Walk Forward" document that is open in the main application frame. When optimization is running you can click "MINIMIZE" button on the Progress dialog to minimize it - this allows to see the Walk Forward output during the optimization steps.

IN-SAMPLE and OUT-OF-SAMPLE combined equity

Combined in-sample and out-sample equities are available by ~~~ISEQUITY and ~~~OSEQUITY composite tickers (consecutive periods of IS and OOS are concatenated and scaled to maintain continuity of equity line - this approach assumes that you generally speaking are compounding profits).

To display IS and OOS equity you may use for example this:

```
PlotForeign("~~~ISEQUITY","In-Sample Equity", colorRed, styleLine);
PlotForeign("~~~OSEQUITY","Out-Of-Sample Equity", colorGreen, styleLine);
Title = "{{NAME}} - {{INTERVAL}} {{DATE}} {{VALUES}}";
```

OUT-OF-SAMPLE summary report (new in 5.60)

Version 5.60 brings a new walk-forward summary report that covers all out-of-sample steps. It is visible in the Report Explorer as last one and has "PS" type.
There were significant changes to walk forward testing made to allow summary out-of-sample report. The most important change is that each subsequent out-of-sample test uses initial equity equal to previous step ending equity. (Previously it used constant initial equity). This change is required for proper calculation of all statistics/metrics throughout all sections of out-of-sample test.

Summary report shows the note that built-in metrics correctly represent all out-of-sample steps but summary custom metrics are composed using user-definable method:
1 first step value, 2 last step value, 3 sum, 4 average, 5 minimum, 6 maximum.

By default summary report shows last step value of custom metrics UNLESS user specifies different combining method in
bo.AddCustomMetrics() call.

bo.AddCustomMetrics has now new optional parameter - CombineMethod

bool AddCustomMetric(string Title, variant Value, [optional] variant LongOnlyValue, [optional] variant ShortOnlyValue , [optional] variant DecPlaces = 2, [optional] variant CombineMethod = 2)

This method adds custom metric to the backtest report, backtest "summary" and optimization result list. Title is a name of the metric to be displayed in the report, Value is the value of the metric, optional arguments LongOnlyValue, ShortOnlyValue allow to provide values for additional long/short-only columns in the backtest report. Last argument DecPlaces controls how many decimal places should be used to display the value.

Supported CombineMethod values are:
1 first step value, - summary report will show the value of custom metric from very first out-of-sample step
2 last step value (default), - summary report will show the value of custom metric from the last out-of-sample step
3 sum, - summary report will show the sum of the values of custom metric from all out of sample steps
4 average, - summary report will show the average of the values of custom metric from all out of sample steps
5 minimum, - summary report will show the smallest value of custom metric from all out of sample steps
6 maximum.- summary report will show the largest value of custom metric from all out of sample steps

Note that certain metrics calculation methods are complex and for example averaging them would not lead to mathematically correct representation of all out of sample test. Summaries of all built-in metrics are mathematically correct out-of-the-box (i.e. they are *not* averages, but properly calculated metrics using method that is appropriate for given value). This contrasts with custom metrics, because they are user-definable and it is up to the user to select 'combining' method, and still it may happen that none of the available methods is appropriate.
For that reason the report includes the note that explains what user-definable method was used to combine custom metrics.

Back-testing systems for futures contracts

Introduction

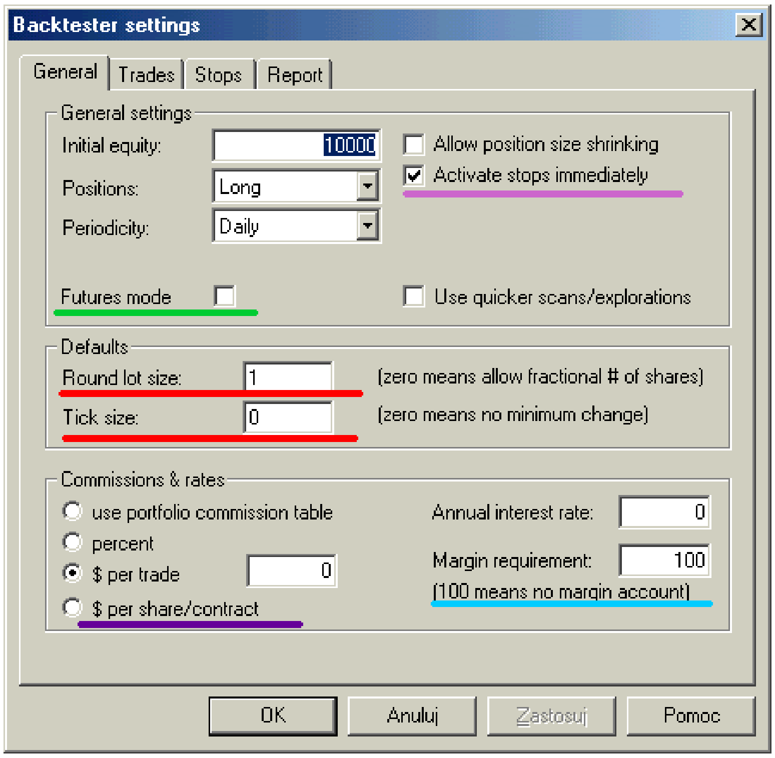
Before you read this article you should read first "Backtesting your trading ideas" section as it gives necessary background of backtesting in general.

When you open long position on stocks you just buy given number of shares at given price, then after some time you sell them and your profit is given by difference between sell and buy price mutliplied by number of shares. If you want to open long position on future contract you pay a deposit - margin - for each contract. The margin is just a little part of full contract value (for example 10%). So you can buy 10 contracts paying no more than full value of one contract. This gives you a leverage that makes trading futures more risky than trading stocks. When price of the contract changes your profit/loss changes accordingly. If contract's point value is 1 each 1$ change in contract price represents 1$ profit/loss per contract - like in stocks. But futures can have point value different that 1. If, for example, point value is 5 each 1 point change in price of the contract represents 5$ profit/loss in your equity. When you close position you get the margin deposit back, so your profit/loss is given by number of contracts multiplied by point value mutlipled by difference between sell and buy prices.

Futures mode of the backtester

There are 3 futures-only settings in the backtester:

- Futures mode check box (Settings-General page)
- Margin deposit (Symbol-Information page)
- Point value (Symbol-Information page)



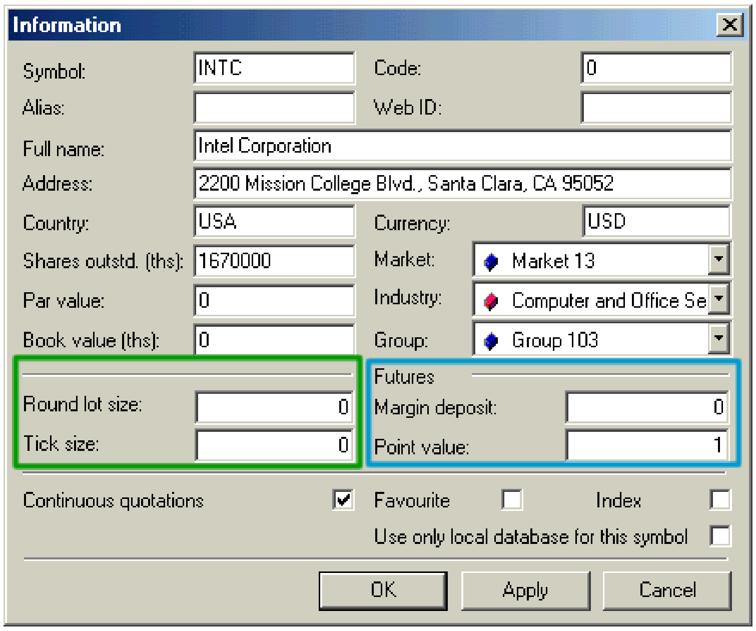
General settings
Initial equity: 10000
Positions: Long
Periodicity: Daily
☐ Allow position size shrinking
☑ Activate stops immediately
Futures mode ☐
☐ Use quicker scans/explorations

Defaults
Round lot size: 1 (zero means allow fractional # of shares)
Tick size: 0 (zero means no minimum change)

Commissions & rates
○ use portfolio commission table
○ percent
● $ per trade 0
○ $ per share/contract
Annual interest rate: 0
Margin requirement: 100
(100 means no margin account)

Futures mode check box in the settings page (underscored with green line in the picture above) is the key to backtesting futures. It instructs backtester to use margin deposit and point value in calculations.

The remaining settings are per-symbol and they are accessible from Symbol->Information window.



Information

Symbol:	INTC	Code:	0
Alias:		Web ID:	
Full name:	Intel Corporation		
Address:	2200 Mission College Blvd., Santa Clara, CA 95052		
Country:	USA	Currency:	USD
Shares outstd. (ths):	1670000	Market:	Market 13
Par value:	0	Industry:	Computer and Office Se
Book value (ths):	0	Group:	Group 103
Round lot size:	0	Futures Margin deposit:	0
Tick size:	0	Point value:	1

Continuous quotations [✓] Favourite [ ] Index [ ]
Use only local database for this symbol [ ]

Margin deposit

The margin is the amount of money required to open single contract position. You can specify per-symbol margin in the Symbol-Information page (picture above). Positive values describe margin value in dollars, while negative express margin value as percentage of contract price. Margin value of zero is used for stocks (no margin). Margin can be also specified in the formula by using MarginDeposit reserved variable:

```
MarginDeposit = 675;
```

In the Futures mode margin setting is used to determine how many contacts can be purchased. Let's suppose that your initial equity is set to $50000 and you want to invest upto 20% of equity in single trade and the margin deposit is $675. In that case your "desired" position size is 50'000 * 0.2 = 10'000. Provided that you have set round lot size to 1, the backtester will "buy" 10000/675 = (integer)14.8148 = 14 contracts, and true positon value will be $9450 (18.9% of the initial equity).

To simulate this in AmiBroker you would need to enter 50000 in the Initial Equity field in the backtester, switch on futures mode, and setup remaining parameters in your formula:

```
PositionSize = -20; // use 20% of equity
MarginDeposit = 675; // this you can set also in the
Symbol-Information page
RoundLotSize = 1; // this you can set also in the Settings page
```

All further trades will use the same logic but position will be sized according to current cumulated equity instead of initial equity level, unless you specify fixed position size in your formula (PositionSize = 10000 for example).

Point value

Point-value is per-symbol setting (definable in Symbol-Information window **-** (picture above)) that determines the amount of profit generated by one contract for a one point increase in price. Example: copper is quoted in cents per pound, a price quote of 84.65 (or 8465) equals 84 cents and 65/100 of a cent per pound. A change of +.37 or 37 represents 37/100ths of a cent you will normally hear it quoted as 37 points. But because of the fact that point value for copper is 2.5 every point change gives $2.5 profit/loss, so in this example profit/loss for the day would be 2.5 * 37 = $92.50.

You can also set it from the formula level using PointValue reserved variable, for example:

```
      PointValue = 2.5;
```

Note: When you load old database AmiBroker presets point value field to 1 and assumes that by default 1 point represents one dollar so one dollar change gives one dollar profit/loss. This is done to ensure that you get correct results even if you (by mistake) run futures mode test on stocks.

Note 2: Although point value setting affects (multiplies) profits/losses it does NOT affect built-in stops. The stops ALWAYS operate on price movement alone. So you should be aware that setting 10% profit target stop will result in 25% profit on trade exited by this stop when point value is set to 2.5.

Simple cases

Points-only test

Points only test is equivalent to trading just one contract. This can be easily accomplished using Futures mode of the backtester and adding the following one line to your formula:

```
      PositionSize = MarginDeposit = 1;
```

Trading 'n' contracts

In a similar way you can setup your formula so it always trades say 7 contracts. All you need to do is to add the following to your formula:

```
      NumContracts = 7;
      PositionSize = NumContracts * MarginDeposit;
```

Monte Carlo Simulation of your trading system

NOTE: Advanced topic. Make sure to read previous parts of the tutorial first. In order to interpret properly Monte Carlo simulation results ***you need to*** *read this section of the manual. Non-trivial settings and non-obvious details are explained below. Please don't skip it.*

Introduction

Generally speaking "Monte Carlo" methods represent broad class of computer algorithms that use repeated random sampling to obtain statistical properties of given process. It was invented by Polish mathematican Stanislaw Ulam working on nuclear weapons projects at the Los Alamos lab. As he was unable to analyse complex physical processes using conventional mathematical methods, he thought that he could set up a series of random experiments, observe the outcomes and use them to derive statistical properties of the process.

More on Monte Carlo methods in general can be found here:
https://en.wikipedia.org/wiki/Monte_Carlo_method

In trading system development, Monte Carlo simulation refers to process of using randomized simulated trade sequences to evaluate statistical properties of a trading system.

There are many ways to perform actual computations that differ when it comes to implementation details, but probably the most straightforward and reliable is bootstraping method that performs random sampling with replacement of actual trade list generated by the back-test.

See https://en.wikipedia.org/wiki/Bootstrapping_(statistics) for detailed discussion of bootstrapping method.

Various Monte Carlo simulation methods allow to verify robustness of the trading system, find out probability of ruin and many other statistical properties of the trading system.

How does it work in AmiBroker?

In order to perform Monte Carlo simulation (or bootstrap test) of your trading system, AmiBroker performs the following:

A. Creating input set

A.1 Perform back-testing of your trading system to produce original set of *N* trades

B. Repeatedly (1000+ times)

B.1 pick randomly trades from the original trade list to produce new, random set of *N* trades (called 'realization')

This random set contains the same number of trades, they are ordered randomly and some original trades may be skipped and some used more than once (permutation with repetition, or random sampling with replacement).

Since number of unique realizations is *N^N* (so with just 100 input trades we have 100^{100} unique realizations), with sufficient number of trades (>100) the probability of picking identical sequence as original is virtually zero.

B.2 sequentially perform gain/loss calculation for each randomly picked trade, using position sizing defined by the user to produce system equity

B.3 record system equity in the distribution

C. Post-process

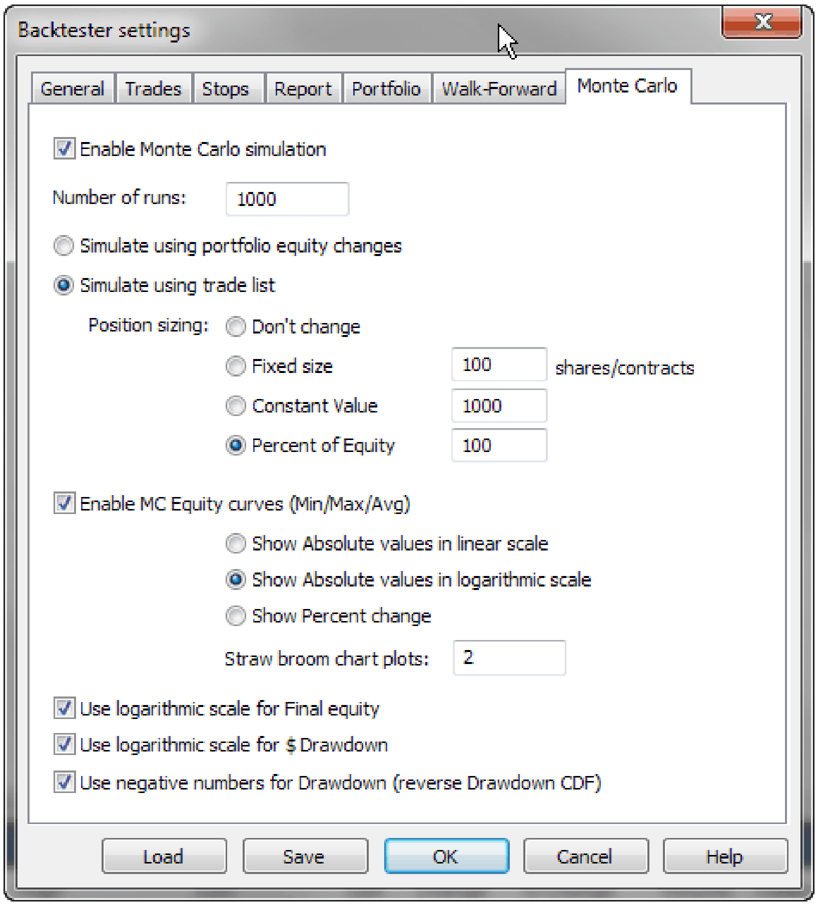
C.1 Process data obtained in B to generate distribution statistics and charts

All of the above happens when you press **Backtest** button in the New Analysis window. AmiBroker's Monte Carlo simulator is so fast that it usually costs just a fraction of second on top of normal backtest procedure.

It should be well noted that simulated trades during bootstrap are performed sequentially. If your original trading system traded multiple positions at once (so some or all of the trades are overlapped) it may result in smaller system drawdowns being reported by bootstrap test, because drawdowns from individual trades would occur sequentially (not in parallel as with overlapping trades).

Settings

The way how Monte Carlo simulator works can be controlled from the Analysis Settings page, "Monte Carlo" tab:



[✓] Enable Monte Carlo simulation

Number of runs: 1000

( ) Simulate using portfolio equity changes

(●) Simulate using trade list

Position sizing: ( ) Don't change
( ) Fixed size 100 shares/contracts
( ) Constant Value 1000
(●) Percent of Equity 100

[✓] Enable MC Equity curves (Min/Max/Avg)

( ) Show Absolute values in linear scale
(●) Show Absolute values in logarithmic scale
( ) Show Percent change

Straw broom chart plots: 2

[✓] Use logarithmic scale for Final equity

[✓] Use logarithmic scale for $ Drawdown

[✓] Use negative numbers for Drawdown (reverse Drawdown CDF)

Enable Monte Carlo simulation

this check box controls whenever MC simulation is performed automatically as a part of backtest (right after backtest generates trade list)

Number of runs

defines the number of MC simulations to run (should be 1000 or more)

Simulate using portfolio equity changes

this option causes that MC simulation uses bar-by-bar portfolio equity percent changes instead of individual trades. Those individual equity changes are randomly picked and permutted to create simulation run. In this mode bar-by-bar equity changes are computed as ratio (so 10% increase is represented as 1.1), selected randomly and multiplied cumulatively. This setting allows to handle situations when you have multiple overlapping trades in your system and does not require any special setting for position sizing.

Simulate using trade list

this option causes that MC simulation uses individual trades from the original backtest to create simulation run. To perform simulation in this mode MC simulator randomly picks original trades and applies new position sizing as defined below. This mode is useful in cases when you don't have overlapping trades.

Position sizing

defines position sizing method used by MC simulator in "trade list" mode:

> *Don't change* - uses original position size as used during backtest. *Keep in mind that it always uses original dollar value of the trade (or whatever currency you use), even if your formula is using percent of portfolio equity.*
>
> *Fixed size* - uses fixed number of shares/contracts per trade
>
> *Constant value* - uses fixed dollar amount for opening any trade
>
> *Percent of equity* - uses defined percent of current simulated equity value. *Be careful when using this setting - it causes that position size of one trade depends on profits on previous trades (compounding profits) and creates serial dependence. It may also lead to extra compounding effect when you have overlapping trades in your original backtest as bootstrap performs trades sequentially (so they don't overlap). For this reason its use is limited to cases when no overlapping trades occur.*

Enable MC equity curves (Min/Max/Avg)

turns on MC equity charts (including highest, lowest and average equity plots plus straw broom equity charts). Note that green and red lines (min/max equity) are not really single "best" and "worst" equities. They are bar-by-bar highest (max) and lowest(min) points of ALL equities generated during MC.
So they are actually best points from all equities and worst points from all equities. And blue line (avg) is the average from all equity lines (all runs).

> *Show absolute value*s in linear scale - displays equities in absolute dollar values using linear scale chart

*Show absolute value*s in logarithmic scale - displays equities in absolute dollar values using semi-log chart

Show Percent change - displays equities as "rate of change" since the beginning

Straw broom chart plots - defines how many individual test equites should be plotted as 'straw broom chart' (large number may slow down processing/drawing)

Use logarithmic scale for Final Equity

Displays final equity CDF chart using semi-log scale instead of linear

Use logarithmic scale for $Drawdown

Displays dollar drawdown CDF chart using semi-log scale instead of linear

Use negative numbers for Drawdown (reverse Drawdown CDF)

When this option is turned on, both dollar and percent drawdown are reported as negative numbers. This has also effect on CDF distribution. It reverses the ordering of "drawdown" column in the MC table and reverses the meaning (i.e. 10% percentile value means that there is 10% chance of drawdowns being equal or worse (more negative) than presented amount. With this option turned off (as in old versions), drawdowns are reported as numbers greater than zero (positive) and 10% percentile value means 10% chance of drawdowns being equal or better (smaller) than presented amount.

Best practices

To remove risks of serial correlation affecting the results of Monte Carlo simulation it is highly encouraged to use fixed position sizing (either fixed dollar value of trades or fixed number of shares/contracts), so the order in which given trade occurs in original sequence does not affect its profit/loss due to compounding.

Also depending on whenever your system opens multiple overlapping positions, choose the simulation method as follows

- **Simulate using trade list -** for systems with non-overlapping trades

  or
- **Simulate using portfolio equity changes**: for systems with overlapping trades (simultaneous positions)

Interpreting the results

The results of Monte Carlo simulation are displayed in the "Monte Carlo" page of Backtest report.

At the top of the page we can see a table that gives values of few key statistics derived from the cumulative distribution charts (**CDFs**) of Monte Carlo simulation results.

Here are sample results (highlights are added manually for the purpose of illustration). Starting equity was 10000 in this example. Test was done over 7 years (EOD data).

Percentile	Final Equity	Annual Return	Max. Drawdown $	Max. Drawdown %	Lowest Eq.

1%	5706	-7.37%	1302	7.23%	3618
5%	7987	-3.02%	1549	9.76%	5853
10%	9706	**-0.41%**	1726	11.32%	6690
25%	12851	3.48%	2136	14.38%	8107
50%	16174	6.78%	2747	19.77%	9135
75%	19632	9.64%	3563	27.63%	9640
90%	23258	12.21%	4626	**38.48%**	9922
95%	25269	13.48%	5292	45.47%	10000
99%	29139	15.71%	7685	**63.82%**	10000

First column shows percentile level (the value below which a given percentage of test observations (realizations) fall). So say 10th percentile tells us that 10% of time observed value is below shown amount. For example, the annual return value at 10th pecentile (in this case -0.41%) means that 10% of tests (realizations) had annual profit less or equal than shown (-0.41%). So we can say that there is about 10% chance that our system would not make any money (would not breakeven). A max. drawdown figure at 90th percentile (38.48%) means that in 90% of cases drawdown will be **less than 38.48%**. So in other words, we can say that there is 10% of chance that it will be higher than that. If we look further in the table we can also notice that in 99% of cases drawdown will be **less than 63.82%.** It is important to note that the table above is generated with "Use negative numbers for Drawdown" turned OFF.

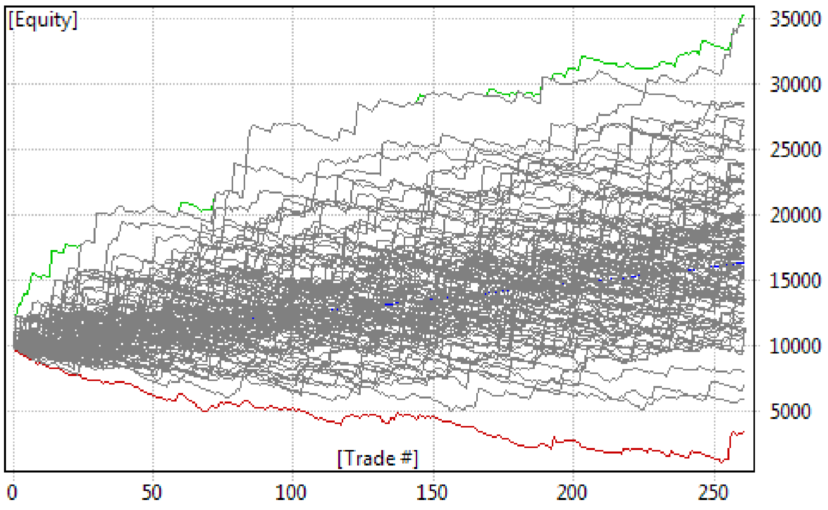
If we turn ON "Use negative numbers for Drawdown" option, all drawdown numbers will become negative, and the order would be reversed and the meaning of drawdown column would be reversed too, like in the table below:

Percentile	Final Equity	Annual Return	Max. Drawdown $	Max. Drawdown %	Lowest Eq.
1%	5706	-7.37%	-7685	**-63.82%**	3618
5%	7987	-3.02%	-5292	-45.47%	5853
10%	9706	**-0.41%**	-4626	**-38.48%**	6690
25%	12851	3.48%	-3563	-27.63%	8107
50%	16174	6.78%	-2747	-19.77%	9135
75%	19632	9.64%	-2136	-14.38%	9640
90%	23258	12.21%	-1726	-11.32%	9922
95%	25269	13.48%	-1549	-9.76%	10000
99%	29139	15.71%	-1302	-7.23%	10000

This time the meaning of drawdown column is reverse - it tells you that the drawdowns would be worse (more negative) than specified amount, 99% percentile value of -7.23% means that in 99% of cases you will see drawdowns worse (more negative) than -7.23%. 1% percentile valuue of -63.82% tells you that in 1% of cases you would experience drawdowns equal or worse (more negative) than -63.82%

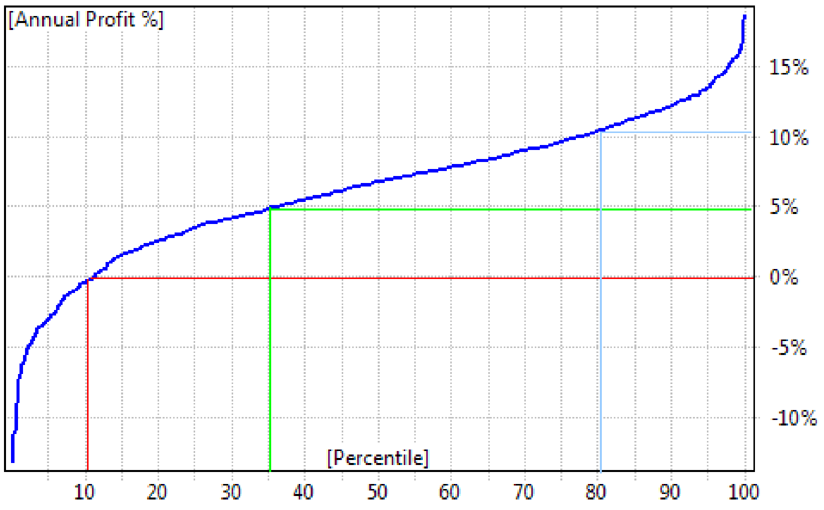
This way the table can be read "row-wise" and the top of the table (small percentiles) refer to "pessimistic" scenarios.

Below the table we can find min/avg/max + straw broom chart of simulated equities:



Note that green and red lines (min/max equity) are not really single "best" and "worst" equities. They are bar-by-bar highest (max) and lowest(min) points of ALL equities generated during MC. So they are actually best points from all equities and worst points from all equities. And blue line (avg) is the average from all equity lines (all runs). The 'cloud' of gray lines represents individual test equities - as we can see the same trading system may generate different outcomes when market conditions change and MC simulation attempts to simulate various outcomes and provide you some statistical information on how bad/good it may be.

After straw broom chart you can find cumulative distribution function (**CDF**) charts of final equity, CAR, drawdowns and lowest equity (again green and red annotation lines were added manually):



Cumulative distribution charts presents the same information that was included in the table at the top of "Monte Carlo" page but in the graphical form. Again, when we take a look at annual profit % (CAR) distribution chart we can see that in approximately 10% of cases our system would not break even (produces negative CAR). We can also see that in approximately 35% of cases our CAR would be below 5%. Profits above 10% per year only occur in top 20% of tests.

All other charts in the MC page are constructed the same and you can read them using the same methodology.

Final equity chart shows the cumulative distribution function of final value of the equity (at the end of test period)

Annual return chart shows the cumulative distribution function of compound annual percentage return of the test

Max. Drawdown $ and Max. Drawdown % charts show the cumulative distribution function of drawdowns (maximum peak to valey dollar/percent distances) experienced during the test

Lowest Equity chart shows the cumulative distribution function of lowest equity ever experienced during the test

How to control it from the formula level?

In addition to using Settings dialog, you can control Monte Carlo simulator using SetOption() function. You can also retrieve those values using GetOption function.

SetOption("MCEnable", 0); // value == 0 disables MC simulation

SetOption("MCEnable", 1); // value == 1 enables MC only in portfolio backtests (default)

SetOption("MCEnable", 2); // value == 2 forces MC to be enabled everywhere (in every mode including optimization - SLOW !)

Note that enabling MC in optimization is highly discouraged unless you actually use MC metrics as optimization target via custom backtester
or otherwise use MC distributions in the optimization process. Monte Carlo process is computationally costly and while few hundred milliseconds added to one backtest don't matter much, in case of optimizations when these are multipled by number of steps you can easily increase optimization time by orders of magnitude. So unless you REALLY need MC distribution as custom metric and optimization target, do NOT enable MC in optimization.

SetOption("MCRuns", 1000); // define number of MC simulation runs (realizations)

Other MC parameters that can be set using SetOption and retrived using GetOption:

- "MCChartEquityCurves" (true/false)
- "MCStrawBroomLines" (0..100)
- "MCPosSizePctEquity" (0..100)
- "MCPosSizeMethod" - 0 - don't change, 1 - fixed size, 2 - constant amount, 3 - percent of equity,
- "MCPosSizeShares" (number),
- "MCPosSizeValue" (number)
- "MCPosSizePctEquity" (number)
- "MCUseEquityChanges" (number), 1 means use equity changes instead of trade list
- "MCChartEquityScale" (number), 1 for log scale, 0 for linear scale
- "MCLogScaleFinalEquity" (number), 1 for log scale, 0 for linear scale
- "MCLogScaleDrawdown" (number), 1 for log scale, 0 for linear scale
- "MCNegativeDrawdown" (number), 1 - use negative numbers for drawdown (reverse drawdown CDF)

How to add custom metric based on MC test distribution(s) to the backtest report ?

In addition to built-in MC report, you can add your own custom metrics to the report using GetMonteCarloSim() method of the Backtester object and MonteCarloSim object that this function returns. If you are new to custom metrics, please consult "How to add custom metrics to backtester report" part of this manual first.

MonteCarloSim object has one function GetValue("field", percentile) that allows to access CDF values. Available "field" values are:

- "FinalEquity"
- "CAR"
- "LowestEquity"
- "MaxDrawdown"
- "MaxPercDrawdown"

Now here is the sample code that presents how to add 30th percentile FinalEquity and CAR to the report:

```
SetOption( "MCEnable", True );
SetOption( "MCRuns", 1000 );
SetCustomBacktestProc( "" );

if( Status( "action" ) == actionPortfolio )
{
    bo = GetBacktesterObject();

    bo.Backtest(); // run default backtest procedure

    // get access to Monte Carlo results
    // note 1: it may be NULL if MC is NOT enabled
    // note 2: MC results are available after Backtest() or PostProcess
    // as MC simulation is done in final phase of post processing
    mc = bo.GetMonteCarloSim();

    if( mc )
    {
         // get 30-th percentile of final equity and CAR distribution
         bo.AddCustomMetric( "FinalEq30", mc.GetValue( "FinalEquity", 30 ) );
         bo.AddCustomMetric( "CAR30", mc.GetValue( "CAR", 30 ) );

         // you can also combine MC stats with normal stats
         st = bo.GetPerformanceStats(0);
         bo.AddCustomMetric( "CAR30/MDD", mc.GetValue( "CAR", 30 ) / st.GetValue(
"MaxSystemDrawdownPercent" ) );
    }
}
```

Once custom metrics is added, it can be used as Optimization target (don't forget to change MCEnable to 2) and used in Walk Forward test process as objective function. To select custom metric as optimization target, you would need to type its name exactly as it appears in the AddCustomMetric call into "Optimization Target" field in the Settings dialog, Walk Forward page. This way you can run optimization / walk forward test that is directed by values of MC simulation distribution. So for example instead of using CAR/MDD you can use CAR30/MDD (30th percentile MC CAR divided by max. system drawdown).

How about Monte Carlo randomization instead of bootstrap test?

The Monte Carlo randomization is different than bootstrap test because it does not use actual (realized) trade list from the backtest but it attempts to use "all individual returns whenever they are realized or hyphotetical". For example when trading system is generating way more signals than we can actually trade due to limited buying power, then we have to choose which trades we would take and which we would skip. Normally this selection is a part of trading system and in AmiBroker PositionScore variable tells the backtester which positions are preferred and should be traded. In randomization test, instead of using some analytic/deterministic PositionScore, you use random one. If there are more signals to open positions than we could take, this process would lead to randomized trade picks. Now using Optimize() function and random PositionScore we can run thousands of such random picks to produce Monte Carlo randomization test:

```
step = Optimize( "step", 1, 1, 1000, 1 ); // 1000 backtests
// with random trade picks from the broad universe (make sure you run it on large
watch lists)
PositionScore = mtRandom();
```

Randomization test has one big disadvantage: can not be used in many cases. When system does not produce enough signals each bar there is not much (if any) to choose from. Also, more importantly, MC randomization makes false assumption that all "trading opportunities" (signals) are equal. In many cases they are not. Pretty often our trading system has specific, deterministic way to pick trades from many oppotunities by some sort of ranking/scoring. When system is using a score (rank) as a core component of the system (rotational systems do that) - if you replace analytic score of with random number you are just testing white noise not the system.

Pyramiding (scaling in/out) and mutliple currencies in the portfolio backtester

IMPORTANT: Please read first Tutorial: Backtesting your trading ideas article and Portfolio Backtesting

Starting from version 4.70 portfolio backtester allows position scaling and supports multiple currencies. Note that these advanced features are supported by PORTFOLIO backtester only. Old single-security backtester and single-security equity() function do NOT support these features.

Pyramiding / Scaling

Two special constants: sigScaleIn / sigScaleOut added to provide means to tell the backtester when you want to scale-in/out

All you have to do to implement pyraminding is to:
- Assign sigScaleIn to BUY/SHORT variable if you want to scale-in (increase size of) LONG/SHORT position
- Assign sigScaleOut to BUY/SHORT variable if you want to scale-out (decrease size of) LONG/SHORT position

Scaling size is defined by PositionSize variable which in case of scaling defines not absolute positionsize but dollar increase or decrease.

IMPORTANT: Please note that backtester treats trade that you scale-in/out as SINGLE trade (i.e. will show single row in trade list). The only difference versus plain trade is that it will calculate average entry price (and avg. entry fx rate) based on all partial entries and average exit price (and avg. exit fx rate) based on all parial exits and will show average prices in entry/exit price field. The commission is of course applied correctly to each (partial) entry/exit depending on partial buy/sell size.

If you want to see details about scaling you have to run backtest in "DETAIL LOG" mode as only then you will see how scaling-in /out works and how average prices are calculated.

Note also that scaling-in/-out and multiple-currency support is available only in portfolio backtester. Old backtester as well as Equity() function do NOT handle scaling-in/out nor multiple currencies (they simply ignore scaling commands).

Easy examples:

Example 1: dollar-cost averaging (each month you buy stocks for fixed dollar amount)

```
FixedDollarAmount = 500;
MonthBegin = Month() != Ref( Month(), -1 );

FirstPurchase = Cum( MonthBegin ) == 1;

Buy = IIf( FirstPurchase, 1, // True (or 1) represents regular buy signal
      IIf( MonthBegin, sigScaleIn, // each month increase position
           0 ) ); // otherwise no signal

Sell = 0; // we do not sell
```

```
PositionSize = FixedDollarAmount;
```

Example 2: dollar-cost averaging
(simplified formula because AB treats first sigScaleIn as buy anyway)

```
FixedDollarAmount = 500;
MonthBegin = Month() != Ref( Month(), -1 );

FirstPurchase = Cum( MonthBegin ) == 1;

Buy = IIf( MonthBegin, sigScaleIn, 0 ); // each month increase position

Sell = 0; // we do not sell

PositionSize = FixedDollarAmount;
```

Example 3: increasing position when profit generated by trade without pyramiding becomes greater than 5% and decreasing position when loss is greater than -5%

```
// percent equity change threshold when pyramiding is performed
PyramidThreshold = 5;

// regular trading rules (no pyramiding)
Buy = Cross( MACD(), Signal() );
Sell = Cross( Signal(), MACD() );

e = Equity(1); // generate equity without pyramiding effect

PcntProfit = 100 * ( e - ValueWhen( Buy, e ) )/ValueWhen( Buy, e );

InTrade = Flip( Buy, Sell );

// ExRem is used here to ensure that scaling-in/out occurs
// only once since trade entry
DoScaleIn = ExRem( InTrade AND PcntProfit > PyramidThreshold, Sell );
DoScaleOut = ExRem( InTrade AND PcntProfit < -PyramidThreshold, Sell );

// modify rules to handle pyramiding
Buy = Buy + sigScaleIn * DoScaleIn + sigScaleOut * DoScaleOut;

PositionSize = IIf( DoScaleOut, 500, 1000 ); // enter and scale-in size $1000,
scale-out size: $500
```

Example 4: partial exit (scaling out) on profit target stops

Example of code that exits 50% on first profit target, 50% on next profit target and everything at trailing stop:

```
Buy = Cross( MA( C, 10 ), MA( C, 50 ) );
Sell = 0;

// the system will exit
// 50% of position if FIRST PROFIT TARGET stop is hit
// 50% of position is SECOND PROFIT TARGET stop is hit
// 100% of position if TRAILING STOP is hit

FirstProfitTarget = 10; // profit
SecondProfitTarget = 20; // in percent
TrailingStop = 10; // also in percent

priceatbuy=0;
highsincebuy = 0;

exit = 0;

for( i = 0; i < BarCount; i++ )
{
   if( priceatbuy == 0 AND Buy[ i ] )
    {
       priceatbuy = BuyPrice[ i ];
    }

   if( priceatbuy > 0 )
    {
       highsincebuy = Max( High[ i ], highsincebuy );

      if( exit == 0 AND
          High[ i ] >= ( 1 + FirstProfitTarget * 0.01 ) * priceatbuy )
       {
         // first profit target hit - scale-out
         exit = 1;
         Buy[ i ] = sigScaleOut;
       }

      if( exit == 1 AND
          High[ i ] >= ( 1 + SecondProfitTarget * 0.01 ) * priceatbuy )
       {
         // second profit target hit - exit
         exit = 2;
         SellPrice[ i ] = Max( Open[ i ], ( 1 + SecondProfitTarget * 0.01 ) *
priceatbuy );
       }

      if( Low[ i ] <= ( 1 - TrailingStop * 0.01 ) * highsincebuy )
       {
         // trailing stop hit - exit
         exit = 3;
         SellPrice[ i ] = Min( Open[ i ], ( 1 - TrailingStop * 0.01 ) *
highsincebuy );
       }
      if( exit >= 2 )
       {
         Buy[ i ] = 0;
         Sell[ i ] = exit + 1; // mark appropriate exit code
         exit = 0;
         priceatbuy = 0; // reset price
         highsincebuy = 0;
       }
    }
}

SetPositionSize( 50, spsPercentOfEquity );
SetPositionSize( 50, spsPercentOfPosition * ( Buy == sigScaleOut ) ); // scale
out 50% of position
```

Mulitple Currency Support

The portfolio backtester allows to backtest systems on securites denominated in different currencies. It includes ability to use historical (variable) currency rates. Currency rates are definable in "Currencies" page in the preferences. The currency in which given symbol is denominated in can be entered in Symbol->Information page.

"Currencies" page in Preferences - allows to define base currency and exchange rates (fixed or dynamic) for different currencies. This allows to get correct backtest results when testing securities denominated in different currency than your base portfolio currency.

How does AB know whether I want the fixed or dynamic quote?

There are following requirements to use currency adjustements:
a) Symbol->Information, "Currency" field shows currency different than BASE currency
b) Appropriate currency (defined in Symbol) has matching entry in Preferences->Currencies page
c) the dynamic rate "FX SYMBOL" defined in the preferences EXISTS in your database and HAS QUOTES for each day under analysis range.

What is "INVERSE" check box for in the preferences?

Let's for example take EURUSD.

When "USD" is your BASE currency then EUR exchange rate would be "straight" EURUSD fx (i.e. 1.3).
But when "EUR" is your BASE currency then USD exchange rate would be INVERSE of EURUSD (i.e. 1/1.3).
Opposite would be true with FX rates like USDJPY (which are already "inverse").

Using formula-based alerts

Introduction

AmiBroker allows you to define formula-based alerts. When alert is triggered a text can be displayed, user-defined sound played back, e-mail notification can be sent and any external application can be launched. This is all handled by single AlertIF function.

By default all alerts generate text that is displayed in the Alert Output window.

To show this window you have to select Window->Alert Output menu.

There is also Easy Alerts window that allows you to define simple alerts that do not require any coding (but do not offer full flexibility of AlertIf function).

Settings

Alert - related settings are present in the "Alerts" tab of Tools->Preferences window.

It allows to define e-mail account settings, test sound output and define which parts of AmiBroker can generate alerts via AlertIF function.

E-mail setting page now allows to choose among most popular authorization schemes like: AUTH LOGIN (most popular), POP3-before-SMPT (popular), CRAM-MD5, LOGIN PLAIN.

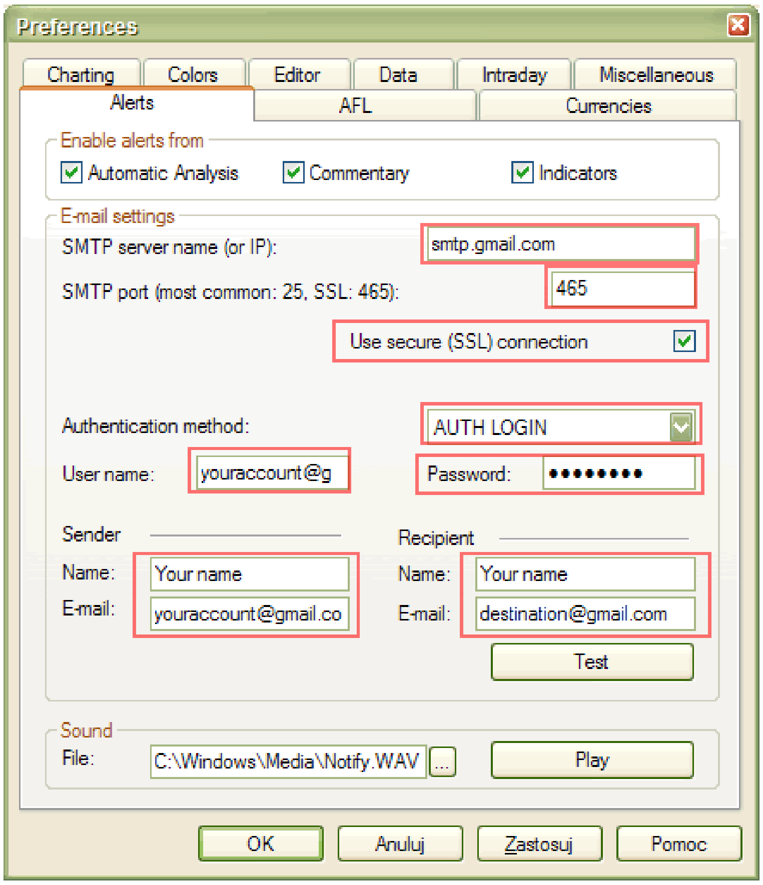
"Enable alerts from" checkboxes allow you to selectively enable/disable alerts generated by Automatic analysis, Commentary/Interpretation and custom indicators.

Alert output window now has an additional column that shows the source of alert - if this is Automatic Analysis, Commentary or one of your custom indicators. This makes it easier to find out which part of AmiBroker generates alerts.

New in AmiBroker 5.30 - support for SSL (secure connection) used by GMail for example.

In order to enable SSL support you need to follow these steps:

1. Download and run SSL add-on from http://www.amibroker.com/bin/SSLAddOn.exe
2. Configure (Tools->Preferences->Alerts) with SSL enabled as shown below



Enable alerts from
✓ Automatic Analysis ✓ Commentary ✓ Indicators

E-mail settings
SMTP server name (or IP): smtp.gmail.com
SMTP port (most common: 25, SSL: 465): 465
Use secure (SSL) connection ✓
Authentication method: AUTH LOGIN
User name: youraccount@g Password: ••••••••

Sender
Name: Your name
E-mail: youraccount@gmail.co

Recipient
Name: Your name
E-mail: destination@gmail.com

Sound
File: C:\Windows\Media\Notify.WAV

AlertIF function

AlertIF function is similar to WriteIF. But instead of just writing the text to the output window (commentary/interpretation) it allows to:

- direct the customized text to "alert output" window,
- make a sound (just by computer beeper or from .WAV file)
- send an e-mail
- launch any external application

The syntax is as follows:

AlertIf(*BOOLEAN_EXPRESSION*, *command*, *text*, *type* = 0, *flags* = 1+2+4+8, *lookback* = 1);

1. BOOLEAN_EXPRESSION is the expression that if evaluates to True (non zero value) triggers the alert. If it evaluates to False (zero value) no alert is triggered. Please note that only *lookback* most recent bars are considered.

2. The *command* string defines the action taken when alert is triggered. If it is empty the alert *text* is simply displayed in the Alert output window (Window->Alert Output). Other supported values of *command* string are:

SOUND *the-path-to-the-WAV-file*
EMAIL
EXEC *the-path-to-the-file-or-URL <optional args>*

SOUND command plays the WAV file once.

EMAIL command sends the e-mail to the account defined in the settings (Tools->Preferences->E-mail). The format of the e-mail is as follows:

Subject: Alert type_name (*type*) Ticker on Date/Time
Body: *text*

EXEC command launches external application or file or URL specified after EXEC command. <optional args> are attached after file name and *text* is attached at the end

3. Text defines the text that will be printed in the output window or sent via e-mail or added as argument to the application specified by EXEC command

4. *Type* defines type of the alert. Pre-defined types are 0 - default, 1 - buy, 2 - sell, 3 - short, 4- cover. YOu may specify higher values and they will get name "other". Type is important. SEE THE COMMENTS BELOW.

5. *Flags* control behaviour of AlertIF function. This field is a combination (sum) of the following values:
(1 - display text in the output window, 2 - make a beep (via computer speaker), 4 - don't display repeated alerts having the same type, 8 - don't display repeated alerts having the same date/time) By default all these options are turned ON.

6. *lookback* parameter controls how many recent bars are checked

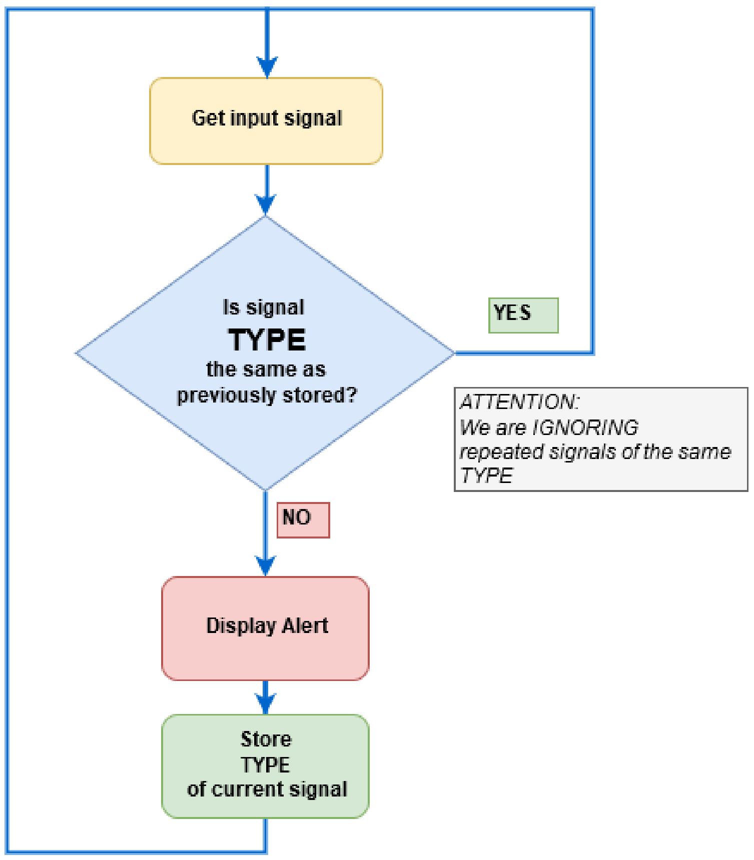
Examples:

```
Buy = Cross( MACD(), Signal() );
Sell = Cross( Signal(), MACD() );
Short = Sell;
Cover = Buy;

AlertIF( Buy, "EMAIL", "A sample alert on "+FullName(), 1 );

AlertIF( Sell, "SOUND C:\\Windows\\Media\\Ding.wav", "Audio alert", 2 );

AlertIF( Short, "EXEC Calc.exe", "Launching external application", 3 );

AlertIF( Cover, "", "Simple text alert", 4 );
```

Note EXEC command uses ShellExecute function and allows not only EXE files but URLs too.

Internal logic

Alertif function implements internal logic in the form of finite state machine that prevents repeated signals of the same type from occuring. State (i.e. the TYPE of last alert) is stored on PER-SYMBOL basis. So each symbol has its "last alert" attached. For per-symbol logic, eee the flowchart below:


Get input signal
Is signal TYPE the same as previously stored?
YES
ATTENTION: We are IGNORING repeated signals of the same TYPE
NO
Display Alert
Store TYPE of current signal

Notes

1. Please once again note that by default AlertIf function **does not generate repetitive signals when the same scan is run multiple times.** During experimentation you may prefer to get repeated signals in subsequent scans. To do so you should change default flags to 1 + 2:

```
AlertIF( condition, "", "Text", 1, 1+2 );
```

2. If you want to generate the alert only on COMPLETED bar you may need to add this code:

```
barcomplete = BarIndex() < LastValue(BarIndex());

AlertIF( barcomplete AND condition, "", "Text", 1 );
```

3. There are simpler functions that do actions like sending email, playing a sound or executing external program without being involved in complex internal logic:

- If you just need to simply send email use: SendEmail
- If you need to play a sound use PlaySound
- If you need to execute program use ShellExecute

Using interpretation window

Note: Please read How to write your own chart commentary article before proceeding.

Interpretation window (Window->Interpretation) shows chart-sensitive commentaries. To add a interpretation just use Formula Editor and add commentary code after the code for the indicator. Please note that to get the best performance you should use conditional statement that ensures that interpretation code is executed only in "commentary" mode.

```
if( Status("action") == actionCommentary )
{
// printf statements here....
}
```

Example:

```
Plot( Close, "Price", -1, 64 );
Plot( SAR( Prefs( 50 ), Prefs( 51 ) ), "SAR",-17, 8+16 );

if( Status("action") == actionCommentary )
{
  printf("The Parabolic SAR provides excellent exit points. \n");
 printf("You should Close long positions when the price falls below\n");
 printf("the SAR AND Close Short positions when the price rises above the
SAR.\n");
 printf( WriteIf( Graph1 > Close, "SAR is above close", "SAR is below close" ) );
}
```

Multiple Time Frame support in AFL

IMPORTANT: TimeFrame functions are NOT intended to replace Periodicity setting. To switch periodicity/interval you should only use Periodicity setting. TimeFrame functions are ONLY for formulas that MIX many different intervals at once in single formula.

Release 4.41 brings ability to use multiple time frames (bar intervals) in single formula. The time frame functions can be divided into 3 functional groups:

1. switching time frame of build-in O, H, L, C, V, OI, Avg arrays: **TimeFrameSet**, **TimeFrameRestore**
2. compressing/expanding single arrays to/from specified interval: **TimeFrameCompress, TimeFrameExpand**
3. immediate access to price/volume arrays in different time frame: **TimeFrameGetPrice**

First group is used when your formula needs to perform some calculations on indicators in different time frame than currently selected one. For example if you need to calculate 13-bar moving average on 5 minute data and 9 bar exponential avarage from hourly data while current interval is 1 minute you would write:

```
TimeFrameSet( in5Minute ); // switch to 5 minute frame

/* MA now operates on 5 minute data, ma5_13 holds time-compressed 13 bar MA of
5min bars */

ma5_13 = MA( C, 13 );

TimeFrameRestore(); // restore time frame to original

TimeFrameSet( inHourly ); // switch now to hourly

mah_9 = EMA( C, 9 ); // 9 bar moving average from hourly data

TimeFrameRestore(); // restore time frame to original

Plot( Close, "Price", colorWhite, styleCandle );

// plot expanded average

Plot( TimeFrameExpand( ma5_13, in5Minute), "13 bar moving average from 5 min
bars", colorRed );
Plot( TimeFrameExpand( mah_9, inHourly), "9 bar moving average from hourly bars",
colorRed );
```

TimeFrameSet(interval) - replaces current built-in price/volume arrays: open, high, low, close, volume, openint, avg with time-compressed bars of specified interval once you switched to a different time frame all calculations and built-in indicators operate on selected time frame. To get back to original interval call TimeFrameRestore() funciton. If you want to call TimeFrameSet again with different interval you have to restore original time frame first using TimeFrameRestore(). Interval is time frame interval in seconds. For example: 60 is one minute bar. You should use convenient constants for common intervals: in1Minute, in5Minute, in15Minute, inHourly, inDaily, inWeekly, inMonthly.

With version 4.70 you can also specify N-tick intervals. This is done by passing NEGATIVE value as interval. For example -5 will give 5-tick bar compression, and -133 will give 133-tick compression. Please note that using N-tick intervals works only if your database uses Tick base time interval set in **File -> Database Settings** dialog.

```
TimeFrameSet( -133 ); // switch to 133-tick interval
```

IMPORTANT: TimeFrameSet() is NOT an equivalent of the periodicity setting in Analysis Settings. The only use for time-frame functions is when you want to have the trading rules based on MULTIPLE time frames at once. See details in "How it works internally" below.

TimeFrameRestore() - restores price arrays replaced by SetTimeFrame.Note that only OHLC, V, OI and Avg built-in variables are restored to original time frame when you call TimeFrameRestore(). All other variables created when being in different time frame remain compressed. To de-compress them to original interval you have to use TimeFrameExpand.

Once you switch the time frame using TimeFrameSet, all AFL functions operate on this time frame until you switch back the time frame to original interval using TimeFrameRestore or set to different interval again using TimeFrameSet. It is good idea to ALWAYS call TimeFrameRestore when you are done with processing in other time frames.

When time frame is switched to other than original interval the results of all functions called since TimeFrameSet are time-compressed too. If you want to display them in original time frame you would need to 'expand' them as described later. Variables created and assigned before call to TimeFrameSet() remain in the time frame they were created. This behaviour allows mixing unlimited different time frames in single formula.

PLEASE NOTE that you can only compress data from shorter interval to longer interval. So when working with 1-minute data you can compress to 2, 3, 4, 5, 6, ...N-minute data. But when working with 15 minute data you can not get 1-minute data bars. In a similar way if you have only EOD data you can not access intraday time frames.

Second group: TimeFrameCompress/TimeFrameExpand allow to compress and expand single arrays to / from different time frames. Especially worth mentioning is TimeFrameExpand that is used to decompress array variables that were created in different time frame. Decompressing is required to properly display the array created in different time frame. For example if you want to display weekly moving average it must be 'expanded' so the data of one weekly bar covers five daily bars (Monday-Friday) of corresponding week.

TimeFrameExpand(array, interval, mode = expandLast) - expands time-compressed array from 'interval' time frame to base time frame ('interval' must match the value used in TimeFrameCompress or TimeFrameSet)
Available modes:
expandLast - the compressed value is expanded starting from last bar within given period (so for example weekly close/high/low is available on Friday's bar)
expandFirst - the compressed value is expanded starting from first bar within given period (so for example weekly open is available from Monday's bar)
expandPoint - the resulting array gets not empty values only for the last bar within given period (all remaining bars are Null (empty)).
Caveat: expandFirst used on price different than open may look into the future. For example if you create weekly HIGH series, expanding it to daily interval using expandFirst will enable you to know on MONDAY what was the high for entire week.

IMPORTANT: TimeFrameExpand IS REQUIRED for any formula that uses TimeFrame* functions. If you don't expand time compressed data you will have incorrect timestamps (see description below in "How it works").

TimeFrameCompress is provided for completeness and it can be used when you want to compress single array without affecting built-in OHLC,V arrays. If you call TimeFrameCompress it does not affect results of other functions.

```
wc = TimeFrameCompress( Close, inWeekly );

/* now the time frame is still unchanged (say daily) and our MA will operate on
daily data */

dailyma = MA( C, 14 );

/* but if we call MA on compressed array, it will give MA from other time frame
*/

weeklyma = MA( wc, 14 ); // note that argument is time-compressed array

Plot( dailyma, "DailyMA", colorRed );

weeklyma = TimeFrameExpand( weeklyma, inWeekly ); // expand for display

Plot( weeklyma, "WeeklyMA", colorBlue );
```

During this formula the time frame remained at original setting we only compressed single array.

TimeFrameCompress(array, interval, mode = compressLast)
- compresses single array to given interval using given compression mode available modes:
compressLast - last (close) value of the array within interval
compressOpen - open value of the array within interval
compressHigh - highest value of the array within interval
compressLow - lowest value of the array within interval
compressVolume - sum of values of the array within interval

```
Graph0 = TimeFrameExpand( TimeFrameCompress( Close, inWeekly, compressLast ),
inWeekly, expandLast );
Graph1 = TimeFrameExpand( TimeFrameCompress( Open, inWeekly, compressOpen ),
inWeekly, expandFirst );
```

Third group consist of just one useful function: TimeFrameGetPrice which allows to reference price and volume from other time frames without switching /compressing/expanding time frames. Just one function call to retrieve price from higher time frame. It allows also to reference not only current but past bars from different time frames.

TimeFrameGetPrice(pricefield, interval, shift = 0, mode = expandFirst);
- references OHLCV fields from other time frames. This works immediatelly without need to call TimeFrameSet at all.
Price field is one of the following: "O", "H", "L", "C", "V", "I" (open interest). Interval is bar interval in seconds. shift allows to reference past (negative values) and future (positive values) data in higher time frame. For example -1 gives previous bar's data (like in Ref function but this works in higher time frame).

Examples:

```
TimeFrameGetPrice( "O", inWeekly, -1 ) // gives you previous week Open price
TimeFrameGetPrice( "C", inWeekly, -3 ) // gives you weekly Close price 3 weeks
ago
TimeFrameGetPrice( "H", inWeekly, -2 ) // gives you weekly High price 2 weeks ago
TimeFrameGetPrice( "O", inWeekly, 0 ) // gives you this week Open price.
TimeFrameGetPrice( "H", inDaily, -1 ) // gives previous Day High when working on
intraday data
```

Shift works as in Ref() function but it is applied to compressed time frame.

Note these functions work like these 3 nested functions

```
TimeFrameExpand( Ref( TimeFrameCompress( array, interval, compress(depending on
field used) ), shift ), interval, expandFirst )
```

therefore if shift = 0 compressed data may look into the future (weekly high can be known on monday). If you want to write a trading system using this function please make sure to reference PAST data by using negative shift value.

The only difference is that TimeFrameGetPrice is 2x faster than nested Expand/Compress.

Note on performance of TimeFrame functions:

a) Measurements done on Athlon 1.46GHz, 18500 daily bars compressed to weekly time frame

TimeFrameGetPrice("C", inWeekly, 0) - 0.0098 sec (9.8 milliseconds)
TimeFrameSet(inWeekly) - 0.012 sec (12 milliseconds)
TimeFrameRestore( ) - 0.006 sec (6 milliseconds)
TimeFrameCompress(Close, inWeekly, compressLast); - 0.0097 sec (9.7 milliseconds)
TimeFrameExpand(array, inWeekly, expandLast); - 0.0098 sec (9.8 milliseconds)
b) Measurements done on Athlon 1.46GHz, 1000 daily bars compressed to weekly time frameall functions below 0.0007 sec (0.7 millisecond)

How does it work internally ?

Time-frame functions **do not** change the BarCount - they just squeeze the arrays so you have first N-bars filled with NULL values and then - last part of the array contains the actual time-compressed values.

This is why it is essential to expand the data back to the original frame with TimeFrameExpand.

The following simple exploration shows what happens after you switch to a higher timeframe. Run Exploration on current symbol, all quotations, periodicity set to daily and you will see how "weekly close compressed" column contains empty values at the beginning and weekly compressed data at the end of array.

```
Filter = 1;
AddColumn(Close, "Daily close");

TimeFrameSet(inWeekly);
AddColumn(wc = Close, "weekly close compressed");
TimeFrameRestore();
```

```
AddColumn( TimeFrameExpand(wc, inWeekly), "weekly close expanded");
```

EXAMPLES

EXAMPLE 1: Plotting weekly MACD and cross arrows from daily data

```
TimeFrameSet( inWeekly );
m = MACD(12, 26 ); // MACD from WEEKLY data
TimeFrameRestore();

m1 = TimeFrameExpand( m, inWeekly );

Plot( m1, "Weekly MACD", colorRed );
PlotShapes( Cross( m1, 0 ) * shapeUpArrow, colorGreen );
PlotShapes( Cross( 0, m1 ) * shapeDownArrow, colorGreen );
```

EXAMPLE 2: weekly candlestick chart overlaid on line daily price chart

```
wo = TimeFrameGetPrice( "O", inWeekly, 0, expandPoint );
wh = TimeFrameGetPrice( "H", inWeekly, 0, expandPoint );
wl = TimeFrameGetPrice( "L", inWeekly, 0, expandPoint );
wc = TimeFrameGetPrice( "C", inWeekly, 0, expandPoint );

PlotOHLC( wo, wh, wl, wc, "Weekly Close", colorWhite, styleCandle );
Plot( Close, "Daily Close", colorBlue );
```

EXAMPLE 3: Simplified Triple screen system

```
/* switch to weekly time frame */
TimeFrameSet( inWeekly );
whist = MACD( 12, 26 ) - Signal( 12, 26, 9 );
wtrend = ROC( whist, 1 ); // weekly trend - one week change of weekly macd
histogram
TimeFrameRestore();

/* expand calculated MACD to daily so we can use it with daily signals */
wtrend = TimeFrameExpand( wtrend, inWeekly );


/* elder ray */
bullpower= High - EMA(Close,13);
bearpower= Low - EMA(Close,13);

Buy = wtrend > 0 /* 1st screen: positive weekly trend */
AND
bearpower < 0 AND bearpower > Ref( bearpower, -1 ) /* 2nd screen bear power
negative but rising */
AND
H > Ref( H, -1 ); /* 3rd screen, if prices make a new high */

BuyPrice = Ref( H, -1 ); // buy stop level;
```

```
Sell = 0 ; // exit only by stops
ApplyStop( stopTypeProfit, stopModePercent, 30, True );
ApplyStop( stopTypeTrailing, stopModePercent, 20, True );
```

Efficient use of multithreading

AmiBroker 5.50 fully supports multithreading (parallel execution on all CPU cores) in both charting and New Analysis window. This greatly enhances speed of operation and improves responsivity of application as worker AFL execution threads do not block the user interface. For example on 4 core Intel i7 that can run upto 8 threads, it can run upto 8 times faster than old Analysis window. Exact speed up depends on complexity of the formula (the more complex it is, the more speedup is possible), amount of data processed (RAM access may be not as fast as CPU thus limiting possible speed gains).

This chapter describes how to avoid pitfalls that can affect multithreaded performance.

Understanding how multithreading is implemented

It is important to understand one simple rule first - in AmiBroker one thread can run one operation on one symbols' data:

1 operation * 1 symbol = 1 thread

The operation is displaying single chart pane, scan, exploration, backtest, optimization. The consequences are as follows: single chart pane always uses one thread. Also a single backtest or optimization running on one symbol uses one thread only.

But a chart that consists of 3 panes uses 3 threads, even though they all operate on the same symbol. So we can also write:

N operations * 1 symbol = N threads

We can also run single operation (like scan/exploration/backtest/optimization) on multiple symbols, then

1 operation * N symbols = N threads

Of course you can also run multiple Analysis windows each of it running multiple symbols or run multiple charts on multiple symbols, then

P operations * N symbols = (P * N) threads

It is also important to understand that some operations consist of not only AFL execution part but some extra processing and/or user-interface work. In such cases only AFL execution can be done with multiple threads. This has consequences for Individual Backtest mode which will be described in detail further.

Note: In version 5.70 there is one exception from this rule: new multi-threaded individual optimization, that allows to run single-symbol optimization using multiple threads.

Limits

The number of threads that actually are launched depends on your CPU and the version of AmiBroker you are using. **Standard Edition** has a limit of **2 (two) threads** per Analysis window. **Professional Edition** has a limit of **32 threads per Analysis** window. In addition to this limit, AmiBroker will detect how many logical processors are reported by Windows (for example a single Intel i7 920 CPU is recognized as 8 logical processors (4 cores x 2 hyperthreading)) and will not run more threads per single Analysis window than the number of logical processors.

Common pitfals

There are following areas of AFL programming that require some attention if you want to write multithreading-friendly AFL formulas:

1. Avoiding the use of OLE / CreateObject
2. Reducing use of AddToComposite / Foreign to minimum
3. Efficient and correct use of static variables
4. Implementing pre-processing / initialisation in the Analysis window
5. Accessing ~~~Equity symbol

Generally speaking the AFL formula can run in full speed only if it does not access any shared resources. Any attempt to access shared resource may result in formula execution waiting for the semaphore/critical section that protects shared resource from simultaneous modification.

1. Avoiding the use of OLE / CreateObject

AmiBroker fully supports calling OLE objects from AFL formula level, and it is still safe to use, but there are technical reasons to advocate against using OLE. The foremost reason is that OLE is slow especially when called not from "owner" thread.

OLE was developed by Microsoft back in 1990's in the 16-bit days it is old technology and it effectivelly prevents threads from running at full speed as all OLE calls must be served by one and only user-interface thread. For more details see this article:
http://blogs.msdn.com/b/oldnewthing/archive/2008/04/24/8420242.aspx

For this reason, if only possible you should strictly **avoid using OLE / CreateObject** in your formulas.

If you fail to do so, the performance will suffer. Any call to OLE from a worker thread causes posting a message to OLE hidden window and waiting for the main application UI thread to handle the request. If multiple threads do the same, the performance would easily degrade to single-thread level, because all OLE calls are handled by main UI thread anyway.

Not only that. Threads waiting for OLE can easily deadlock when OLE server is busy with some other work. AmiBroker contains some hi-tech patented code that checks for such OLE deadlock condition and is able to unlock from it, but it may take even upto 10 seconds to unlock. Even worse. OLE calls made from non-UI thread suffer from overhead of messaging and marshaling and can be as much as 30 slower compared to when they are called from same process main UI thread. To avoid all those troubles, avoid using OLE if only possible.

For example instead of using OLE to do RefreshAll like this:

```
AB = CreateObject("Broker.Application"); // AVOID THIS
AB.RefreshAll(); // AVOID THIS
```

Use AmiBroker native RequestTimedRefresh function which is orders of magnitude faster and does not cause any problems. If you want to refresh UI after Scan/Analysis/Backtest use
SetOption("RefreshWhenCompleted", True)

Keep in mind that in most cases the refresh is completely automatic (for example after AddtoComposite) and does not require any extra coding at all.

If you use OLE to read Analysis filter settings (such as watch list number), like this:

AB = CreateObject("Broker.Application"); // AVOID THIS
AA = AB.Analysis; // AVOID THIS
wlnum = AA.Filter(0, "watchlist"); // AVOID THIS

you should replace OLE calls by simple, native call to GetOption that allows to read analysis formula filter settings in multithreading friendly manner. For example to read Filter Include watch list number use:

```
wlnum = GetOption("FilterIncludeWatchlist"); // PROPER WAY
```

For more information about supported filter settings fields see GetOption function reference page.

Also note that AB.Analysis OLE object always refers to OLD automatic analysis window. This has side effect of launching/displaying old automatic analysis whenever you use AB.Analysis in your code. As explained above, all calls to OLE should be removed from your formulas if you want to run in New multithreaded Analysis window. It is only allowed to access new Analysis via OLE from external programs / scripts. To access new Analysis from external program you need to use AnalysisDocs/AnalysisDoc objects as described in OLE Automation interface document.

2. Reducing use of AddToComposite / Foreign to minimum

Any access to other than "current" symbol from the formula level involves global lock (critical section) and therefore may impact the performance. For this reason it is recommended to reduce use of AddToComposite/Foreign functions and use static variables wherever possible

3. Efficient and correct use of static variables

The access to static variables is fast, thread safe and atomic on single StaticVarSet/StaticVarGet call level. It means that it reads/writes entire array in atomic way, so no other thread will read/write that array in the middle of other thread updating it.

However, care must be taken if you write multiple static variables at once. Generally speaking when you write static variables as a part of multi-symbol Analysis scan/exploration/backtest, optimization, you should do the writing (StaticVarSet) on very first step using Status("stocknum")==0 as described below. This is recommended way of doing things:

```
if( Status("stocknum") == 0 )
{
 // do all static variable writing/initialization here
}
```

Doing all initialization/writes to static variables that way provides best performance and subsequent reads (StaticVarGet) are perfectly safe and fast. You should avoid making things complex when it is possible to follow simple and effective rule of one writer - multiple readers. As long as only one thread writes and many threads just read static variables, you are safe and you don't need to worry about synchronization.

For advanced formula writers only:
If you, for some reason, need to write multiple static variables that are shared and accessed from multiple threads at the same time, and when you must ensure that all updates are atomic, then you need to protect regions of your formula that update multiple static variables with a semaphore or critical section. For best performance you should group all reads/writes in one section like this:

```
if( _TryEnterCS( "mysemaphore" ) ) // see StaticVarCompareExchange function for
implementation
{
  // you are inside critical section now
  // do all static var writing/reading here - no other thread will interfere here
  _LeaveCS();
}
else
{
    _TRACE("Unable to enter CS");
}
```

The implementation of both semaphore and critical section in AFL is shown in the examples to StaticVarCompareExchange function.

4. Implementing pre-processing / initialisation in the Analysis window

Sometimes there is a need to do some initialization or some time consuming calculation before all the other work is done. To allow for that processing without other threads interferring with the outcome you can use the following if clause:

```
if( Status("stocknum") == 0 )
{
 // initialization / pre-processing code
}
```

AmiBroker detects such statement and runs very first symbol in one thread only, waits for completion and only after completion it launches all other threads. This allows things like setting up static variables for use in further processing, etc. *Caveat: the above statement must NOT be placed inside #include.*

5. Accessing ~~~Equity symbol

Using Foreign("~~~Equity", "C") makes sense only to display chart of the equity of the backtest that **has completed**. It is important to understand that new Analysis window supports multiple instance, and therefore it can not use any shared equity symbol, because if it did, multiple running backtest would interfere with each other. So New Analysis has local, private instance of all equity data that is used during backtesting and only AFTER backtesting is complete, it copies ready-to-use equity data to ~~~Equity symbol. This means that if you call Foreign("~~~Equity", "C") from within the formula that is currently being backtested, you will receive **previous backtest** equity, not current one.

To access current equity, you need to use custom backtester interface. It has "Equity" property in the backtester object that holds current account equity. If you need equity as an array there are two choices, either collect values this way:

```
SetOption("UseCustomBacktestProc", True );

if( Status("action") == actionPortfolio )
{
  bo = GetBacktesterObject();

  bo.PreProcess(); // Initialize backtester
```

```
  PortEquity = Null; // will keep portfolio equity values

  for(bar=0; bar < BarCount; bar++)
  {
    bo.ProcessTradeSignals( bar );

    // store current equity value into array element
    PortEquity[ i ] = bo.Equity;
  }

   bo.PostProcess(); // Finalize backtester

  // AT THIS POINT YOU PortEquity contains ARRAY of equity values

}
```

Or you can use EquityArray property added to Backtester object in v5.50.1

```
if( Status("action") == actionPortfolio )
{
 bo = GetBacktesterObject();
 bo.Backtest();
 AddToComposite( bo.EquityArray, // get portfolio Equity array in one call
                 "~~~MY_EQUITY_COPY", "X",
                 atcFlagDeleteValues | atcFlagEnableInPortfolio );
}
```

Please note that values are filled during backtest and all values are valid only after backtest is complete (as in above example). If you call it in the middle of backtest, it will contain equity only upto given bar. Avoid abusing this function and it is costly in terms of RAM/CPU (however it is less costly than Foreign).

Both ways presented will access local, current copy of equity in New Analysis (unlike Foreign that accesses global symbol values from previous backtest)

Single-symbol operations run in one thread

As explained at the beginning of the article, any operation such as scan, exploration, backtest, optimization or walk forward test that is done on **single symbol** can only use **one thread**. For that reason there is almost no speed advantage compared to running same code in the old versions of AmiBroker.

Update as of 5.70: This version has a new "Individual Optimize" functionality that allows to run single-symbol optimization using multiple threads, albeit some limitations: only exhaustive optimization is supported and no custom backtester is supported. This is for two reasons: a) smart optimization engines need the result of previous step to decide what parameter combination choose for the next step; b) second phase of backtest talks to UI and OLE (custom backtester) and as such can not be run from non-UI thread (see below for the details).

Individual Backtest can only be run in one thread

The most important thing to understand is that the Individual backtest is a portfolio-level backtest ran on just ONE symbol. Even if you run it on watch list, it still executes things sequentially, single backtest on single symbol at once, then moving to next symbol in the watch list. Why this is so is described below.

Both portfolio level and individual backtests consist of the very same two phases
I. running your formula and collecting signals
II. actual backtest that may involve second run of your formula (custom backtester)

Phase I runs the formula on each symbol in the list and it can be multi-threaded (if there is more than one symbol in the list).

Phase II that processes the signals collected in phase I, generates raport and displays results is done only once per backtest.
It can not be multi-threaded because:
a) it talks to User Interface (UI)
b) it uses OLE/COM to allow you to run custom backtester.

Both OLE and UI + access can not be done from worker (non user-interface) thread. Even worse OLE/UI + multithreading equals death, see:
http://blogs.msdn.com/b/oldnewthing/archive/2008/04/24/8420242.aspx

Usually, in case of multi symbol portfolios, Phase I takes 95% of time needed to run portfolio backtest so once you run phase I in multiple threads, you get very good scalability as only 5% is not multi-threaded.

Since individual backtest runs on ONE symbol then the only phase that can be run in multiple threads, i.e. phase 1 - consists of just one run, and as such is run in one thread.

To be able to run Phase II from multiple threads you would NOT be able to talk to UI and would NOT be able to use COM/OLE (no custom backtester).

That causes that Individual Backtest can NOT be any faster than in old Automatic Analysis.

Doing the math & resonable expectations

Some users live in fantasy land and think that they can throw say 100GB data and the data will be processed fast because "they have latest hardware". This is dead wrong. What you will get is a crash. While 64-bit Windows removes 2GB per-application virtual address space barrier, it is not true that there are no limits anymore.

Unfortunatelly even people with technical background forget to do the basic math and have some unresonable expectations. First and foremost thing that people are missing is the huge difference between access speeds made by data size. The term "Random Access Memory" in the past (like back in 1990) meant that accessing data takes the same amount of time, regardless of location. That is NO LONGER the case. There are huge differences in access speeds depending on where data is located. For example Intel i7 920, triple channel configuration accesses L1 cached data with 52GB/second speed, L2 cached data 30GB/second (2x slower!), L3 cached data 24GB/second and regular RAM with 11GB/second. It means that cached data access is 5 times faster than RAM access. Things get even more dramatic if you run out of RAM and system has to go to the disk. With most modern SSD disks we speak about just 200MB/sec (0.2GB/sec). That is two orders (100x) of magnitude slower than RAM and three orders of magnitude slower than cache. That assumes zero latency (seek). In real world, disk access can be 10000 times slower than RAM.
Now do yourself a favour and do the math. Divide 100GB by 0.2GB/second SSD disk speed. What you will get ? 500 seconds - almost ten minutes just to read the data. Now are you aware that if application does not process messages for just 1 second is considered as "not responding" by Windows? What does that mean? It means that even in 64-bit world, any Windows application will have trouble processing data sets that exceed

5GB just because of raw disk read speed that in best case does not exceed 200MB/sec (usually much worse). Attempting to backtest such absurd amounts of data on high-end PC will just lead to crash, because timeouts will be reached, the Windows will struggle processing messages and you will overrun system buffers. And it has nothing to do with software. It is just brutal math lesson that some forgot. First and most important rule for getting more speed is limit your data size, so it at least fits in RAM.

Ranking functionality

A ranking is a relationship between a set of items such that, for any two items, the first is either 'ranked higher than', 'ranked lower than' or 'ranked equal to' the second. The simplest way to obtain the rank is to sort items by 'value' or 'score'. For example you can take 100-bar rate of change for symbols - it will be your item 'score' or 'value. Then sort the results by it so you will get symbol list where first one is best performing (highest rate of change) and the last one is worst performing one.

AmiBroker allows user to perform/use three different kind of rankings

1. use ranking of trade entry signals (buy/short) to decide which entries are preferred over the others during portfolio backtesting/optimization
2. display multiple rankings in tables created using Exploration
3. generate numeric ranks for later use (general-purpose functionality)

The **first kind of ranking** is performed automatically if your trading system formula defines **PositionScore** variable. You can use **PositionScore** variable to decide which trades should be entered if there are more entry signals on different securities than maximum allowable number of open positions or available funds. In such case AmiBroker will use the absolute value of PositionScore variable to decide which trades are preferred. For the details about ranking functionality during backtesting see Portfolio Backtester tutorial.

Second kind of ranking is simply assigning a number (rank) to the line of exploration output. The rank column is added to the exploration output just by calling AddRankColumn function after performing a sort using SetSortColumns function. You can call SetSortColumns multiple times and you can call AddRankColumn multiple times to achieve many different ranks based on multiple-columns. See example below:

```
Filter = 1;
AddColumn( Close, "Close" );
AddColumn( Volume, "BI" );
AddSummaryRows( 31 + 32, 1.5 );

AddRankColumn(); // without prior sorting AddRankColumn just adds line number
SetSortColumns( -4 );
AddRankColumn(); // rank according to 4th column (descending)
SetSortColumns( -3 );
AddRankColumn(); // rank according to 3rd column (ascending)
```

A third kind of ranking is general-purpose, bar-by-bar ranking that is performed using static variables. It is most resource hungry (computationally intensive) but also gives most possibilities.

Generally the process involves creating static variables with values to be used for sorting/ranking, i.e. "scores" and then calling a special function (StaticVarGenerateRanks) that generates new set of static variables that hold calculated ranks.

NOTE: This function is NOT intended to replace bakctester's built-in ranking via PositionScore. Just the opposite: whenever you can, you should use PositionScore as it is way way faster and less memory-consuming way to perform backtests with ranking.

StaticVarGenerateRanks is generally intended to be used for tasks OTHER than backtesting such as explorations or indicators that may require ranking functionality, but of course it can also be used for backtesting when/where PositionScore alone does not allow to implement what you need in your trading system.

WARNING: this function is computationally and memory intensive. It takes about 20ms per 15K bars and 7 symbols. Try to call it just once per scan/exploration/backtest using if(Status("stocknum")==0) or better yet, use separate scan just once to pre-calculate ranks and use it later (like composite creation scan). If you fail to do so and call StaticVarGenerateRanks for every symbol performance would drop significantly as this function not only needs lots of time to compute but it also has to lock the access to shared memory used by static variables so other threads trying to access static variables would wait until this function completes.

StaticVarGenerateRanks function

StaticVarGenarateRanks("outputprefix", "inputprefix", topranks, tiemode) is a core element of general purpose ranking system. It takes 4 parameters: "outputprefix" - the prefix appended to output static variables that hold the ranks, "inputprefix" the prefix of static variables holding scores (input), topranks - which defines how many top/bottom ranking symbols should be included in the generated rank set and tiemode that defines how ties (equal ranks) should be resolved.

The "inputprefix" is a prefix that defines names of static variables that will be used as input for ranking. AmiBroker will search for all static variables that begin with that prefix and assume that remaining part of the variable name is a stock symbol. Say you want to rank stocks by ROC (rate of change). All you need to do is to store values into static variables. Let us say that we will use static variable names like "ItemScoreAPPL", "ItemScoreMSFT", and so on.

To fill input static variables you can use this loop:

```
for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    SetForeign( sym );
    Value = ROC( C, 10 );
    RestorePriceArrays();
    StaticVarSet( "ItemScore" + sym, Value );
}
```

Now you are ready to perform sorting/ranking. There are two modes, normal ranking mode and Top/Bottom Rank mode. Normal ranking mode is performed when toprank argument is set to zero.

StaticVarGenerateRanks("rank", "ItemScore", 0, 1224);

In this case StaticVarGenerateRanks call would generate set of static variables starting with prefix defined by 2nd argument each variable holding the rank of particular symbol, so in this case RankItemScoreMSFT will hold ranking of MSFT, RankItemScoreAAPL will hold ranking of AAPL. Note that in AmiBroker rank count start from ONE.

Third argument (topranks) is zero in normal ranking mode. Fourth argument (tiemode) defines how ties are ranked. Supported modes are 1234 and 1224. In 1224 mode ties are numbered with equal rank.

Example code for normal ranking mode (everything done is done in one pass, can be used in indicator):

```
symlist = "C,CAT,DD,GE,IBM,INTC,MSFT";

// delete static variables
StaticVarRemove( "ItemScore*" );

// fill input static arrays

for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    SetForeign( sym );
     Value = ROC( C, 10 );
    RestorePriceArrays();
    StaticVarSet( "ItemScore" + sym, Value );
}

// perform ranking
StaticVarGenerateRanks( "rank", "ItemScore", 0, 1224 ); // normal rank mode

// read ranking
for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    Plot( StaticVarGet( "RankItemScore" + sym ), sym, colorCustom10 + i );
}
```

Top/bottom ranking mode (that generates top/bottom ranking tables that hold indexes to top ranking values. When topranks > 0 top ranked values are used, when topranks < 0 then bottom ranked values are used. The values are stored in variables that have format of:
OutputprefixInputprefixN where N is a number 1, 2, 3 representing top/bottom ranks. Let us assume that OutputPrefix parameter is "Top" and Inputprefix parameter is ROC. In such case variable TopROC1 would hold the index of top rated value. TopROC2 would hold second top rated value, and so on.
StaticVarGenerateRanks function uses rank numbering that starts from ONE. In top ranking mode StaticVarGenerateRanks will also prepare static variable that contains comma separated list of variable names that can be used to find out which index refers to which symbol. So if TopROC1 holds 1 you would lookup first substring in TopROCSymbols variable to find out what variable (symbol) ranked at the top.
Additionally StaticVarGetRankedSymbols gives easy-to-use method to retrieve comma separated list of ranked symbols for particular datetime.

Example code for top ranking mode:

```
symlist = "C,CAT,DD,GE,IBM,INTC,MSFT";

// delete static variables
StaticVarRemove( "ItemScore*" );

// fill input static arrays

for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    SetForeign( sym );
    Value = ROC( C, 10 );
    RestorePriceArrays();
    StaticVarSet( "ItemScore" + sym, Value );
}

// perform ranking
StaticVarGenerateRanks( "rank", "ItemScore", 0, 1224 ); // normal rank mode

StaticVarGenerateRanks( "top", "ItemScore", 3, 1224 ); // top-N mode

StaticVarGenerateRanks( "bot", "ItemScore", -3, 1224 ); // bottom-N mode

// read ranking
for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    Plot( StaticVarGet( "RankItemScore" + sym ), sym, colorCustom10 + i );
}

sdt = SelectedValue( DateTime() );

Title = "{{NAME}} -{{DATE}} - {{VALUES}} TOP: " + StaticVarGetRankedSymbols(
"top", "ItemScore", sdt ) +
        " BOT: " + StaticVarGetRankedSymbols( "bot", "ItemScore", sdt ) ;
```

How to use StaticVarGenerateRanks in Analysis window

Since ranking is resource hungry process, it should be performed just once per Analysis run, not for every symbol. You can achieve it either by running separate ranking-generation formula once by hand prior to running Analysis or using Status("stocknum") == 0 statement that would ensure that ranking process is done only for the very first symbol from the watch list under analysis.

Here is an example code for exploration that takes currently active watch list or all symbol list and performs ranking

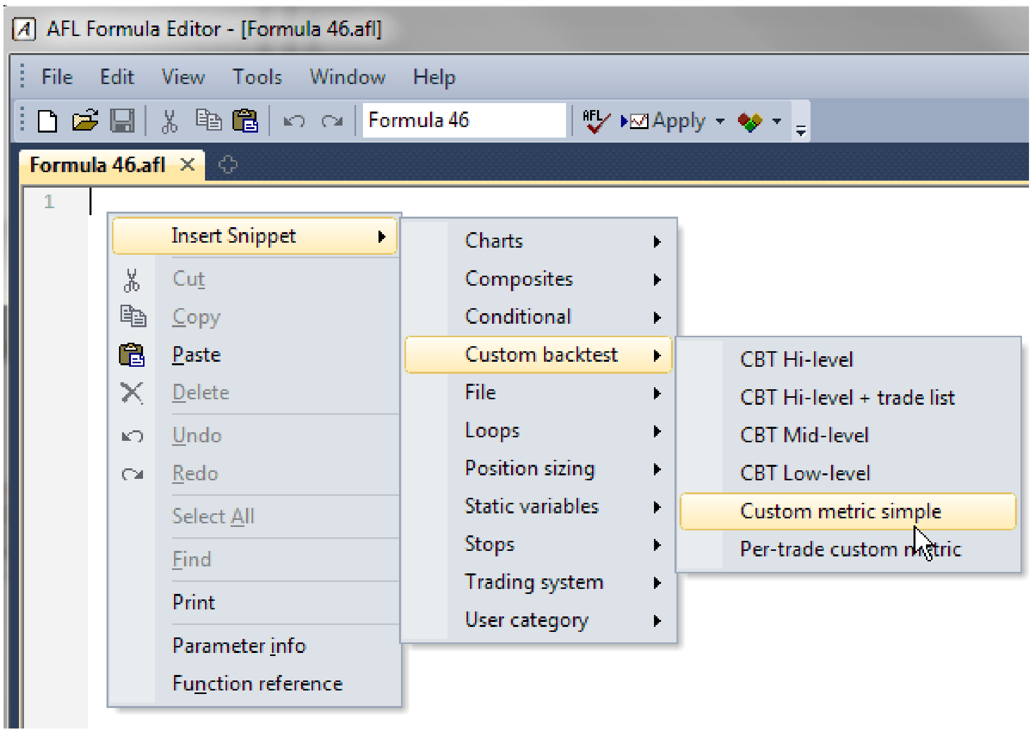
```
if ( GetOption( "ApplyTo" ) == 2 )
{
     wlnum = GetOption( "FilterIncludeWatchlist" );
     List = CategoryGetSymbols( categoryWatchlist, wlnum ) ;
}
else
if ( GetOption( "ApplyTo" ) == 0 )
{
     List = CategoryGetSymbols( categoryAll, 0 );
}
else
{
     Error( "The formula works fine if your ApplyTo setting is 'Filter' or 'All'
" );
}


if ( Status("stocknum") == 0 ) // GENERATE RANKING WHEN WE ARE ON VERY FIRST
SYMBOL
```

```
{
    StaticVarRemove( "values*" );

     for ( n = 0; ( Symbol = StrExtract( List, n ) )  != "";  n++    )
    {
        SetForeign ( symbol );
        values = RSI();
        RestorePriceArrays();
        StaticVarSet (  "values"  +  symbol, values );
        _TRACE( symbol );
    }

    StaticVarGenerateRanks( "rank", "values", 0, 1224 );
}

symbol = Name();

values = StaticVarGet ( "values" +  symbol );
rank = StaticVarGet ( "rankvalues" +  symbol );

AddColumn ( values, "values" );
AddColumn ( rank, "rank" );


Filter = 1;

SetSortColumns( 2, 4 );
```

Using AFL Code snippets

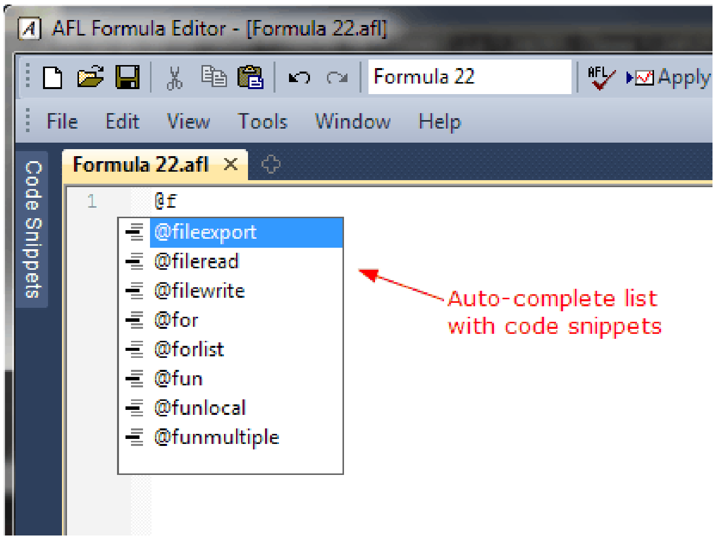
Code snippet is a small piece of re-usable AFL code. It can be inserted by

- right-clicking in the AFL editor window and choosing "Insert Snippet" menu, or
- dragging a snippet from Code Snippet window, or
- typing keyboard trigger (such as @for) in the editor



In version 5.90 Code snippets are also available in auto complete list in the AFL Editor. Just type @ plus first letter of snippet key trigger and auto-complete list would show you the list of available snippets that have keyboard triggers defined starting with that letter.

Replacement of keyboard triggers works even without auto complete activated, so just typing @keytrigger is replaced by snippet text.

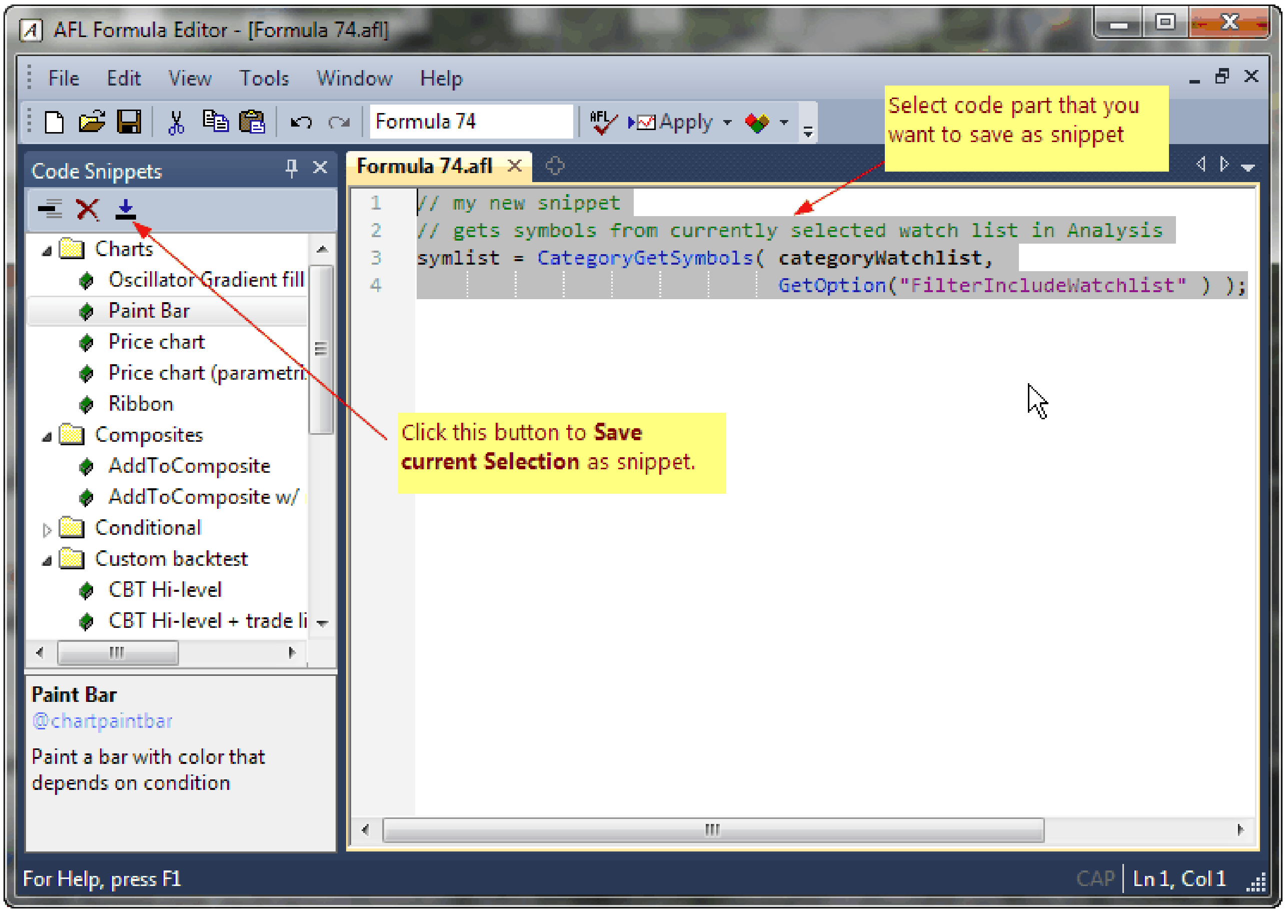


DEFINING YOUR OWN SNIPPETS

You can add your own snippets fairly easy using new Code Snippet window. Code Snippets window is available in new AFL editor. It can be shown/hidden using Window menu.

To create your own snippet, do the following:

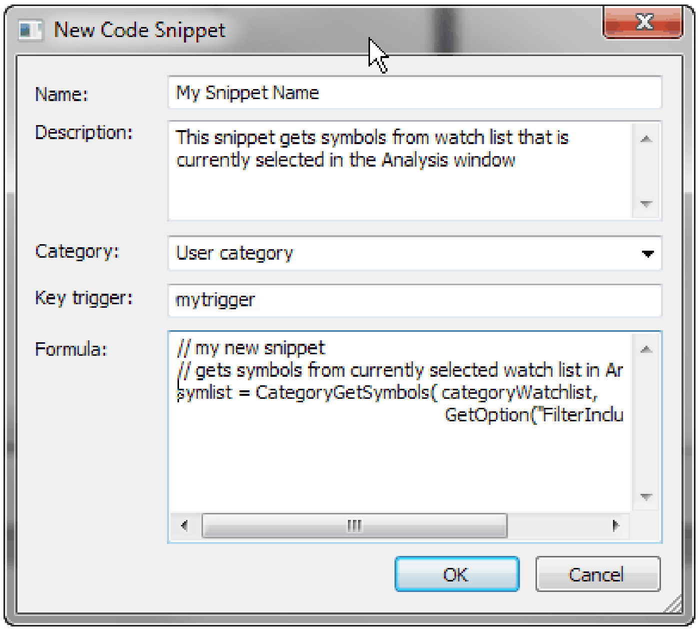
1. type the code you want
2. select (mark) the code you want to place in a snippet
3. press **Save selection** as snippet button in the Code Snippets window



```
// my new snippet
// gets symbols from currently selected watch list in Analysis
symlist = CategoryGetSymbols( categoryWatchlist,
                              GetOption("FilterIncludeWatchlist" ) );
```

Paint Bar
@chartpaintbar
Paint a bar with color that depends on condition

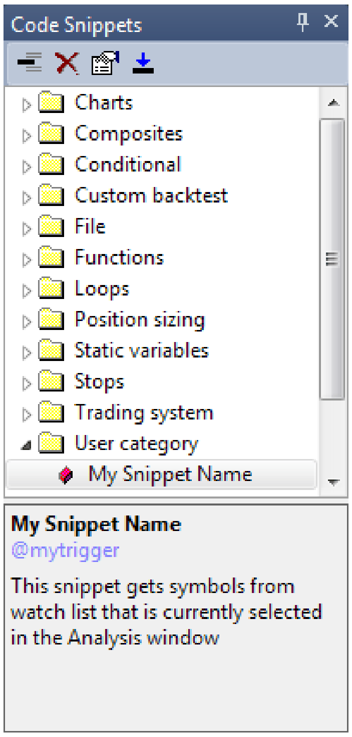
If you do the steps above the following dialog will appear:



Name: My Snippet Name
Description: This snippet gets symbols from watch list that is currently selected in the Analysis window
Category: User category
Key trigger: mytrigger
Formula:

```
// my new snippet
// gets symbols from currently selected watch list in Ar
symlist = CategoryGetSymbols( categoryWatchlist,
                              GetOption("FilterInclu
```

Now you need to enter the **Name** of the snippet, the **Description** and **Category**. **Category** can be selected from already existing items (using drop down box), or new category name can be typed in the category field. **Key trigger** field is optional and contains snippet auto-complete trigger (described above). The **Formula** field is the snippet code itself. Once you enter all fields and press **OK**, your new snippet will appear in the list.

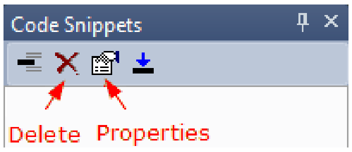


My Snippet Name
@mytrigger
This snippet gets symbols from watch list that is currently selected in the Analysis window

From then on you can use your own snippet the same way as existing snippets. Perhaps most convenient method is using drag-drop from the list to AFL editor.

As you may have noticed user-defined snippets are marked with red color box in the Code Snippets list. Only user-defined snippets can be overwritten and/or deleted.

To edit existing user-defined snippet, you can either follow the steps above and give existing name. AmiBroker will ask then if you want to overwrite existing snippet, or you can simply click on **Properties** button and edit the snippet directly, without re-inserting it.



To delete a snippet, select the snippet you want to delete from the list and press **Delete** (X) button in the Code Snippet window.

TECHNICAL INFO (advanced users only)

There are two files located in AmiBroker directory that hold snippets:
CodeSnippets.xml - these are snippets shipped with AmiBroker installation (and can be replaced in subsequent installations, so don't modify it!)
UserSnippets.xml - these are user-definable snippets. This file is NOT present in the installation and user can create it by him/herself.

The XML schema for snippets file is simple (as below). Key trigger functionality is NOT yet implemented, however Keytrigger fields should be included in the definition for future use. It will be work like 'autocomplete' so that you type the shortcut it, it will unfold to the formula.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<AmiBroker-CodeSnippets CompactMode="0">

<Snippet>
<Name>First Snippet</Name>
<Description>Description of the snippet</Description>
<Category>User category</Category>
<KeyTrigger>?trigger1</KeyTrigger>
<Formula>
<![CDATA[

// the formula itself

]]>
</Formula>
</Snippet>

<Snippet>
<Name>Second Snippet</Name>
<Description>Description of the snippet</Description>
<Category>User category</Category>
<KeyTrigger>?trigger2</KeyTrigger>
<Formula>
<![CDATA[

// the formula itself

]]>
</Formula>
</Snippet>
</AmiBroker-CodeSnippets>
```

How to use AFL visual debugger

Basics

To run the formula under debugger simply choose **Debug->Go (F5 key)** menu in the Formula Editor or click toolbar button marked below as **Start/Continue**.

In most cases the code would execute in a blink of an eye so you won't even see if it worked. You can stop the execution long enough to see the what is happening by setting a **breakpoint** and then stepping ahead.

Breakpoints

To set a breakpoint move the cursor to the line where you want to stop the execution and choose **Debug->Insert/Remove Breakpoint** menu or press press red circle button in the toolbar, or press F9 key. The breakpoint will be shown as red circle in the left margin.

Breakpoints can be added/removed at any time, before you start debugging and while you are debugging. They can be added/removed when debugger is stopped at breakpoint and while the code is running.

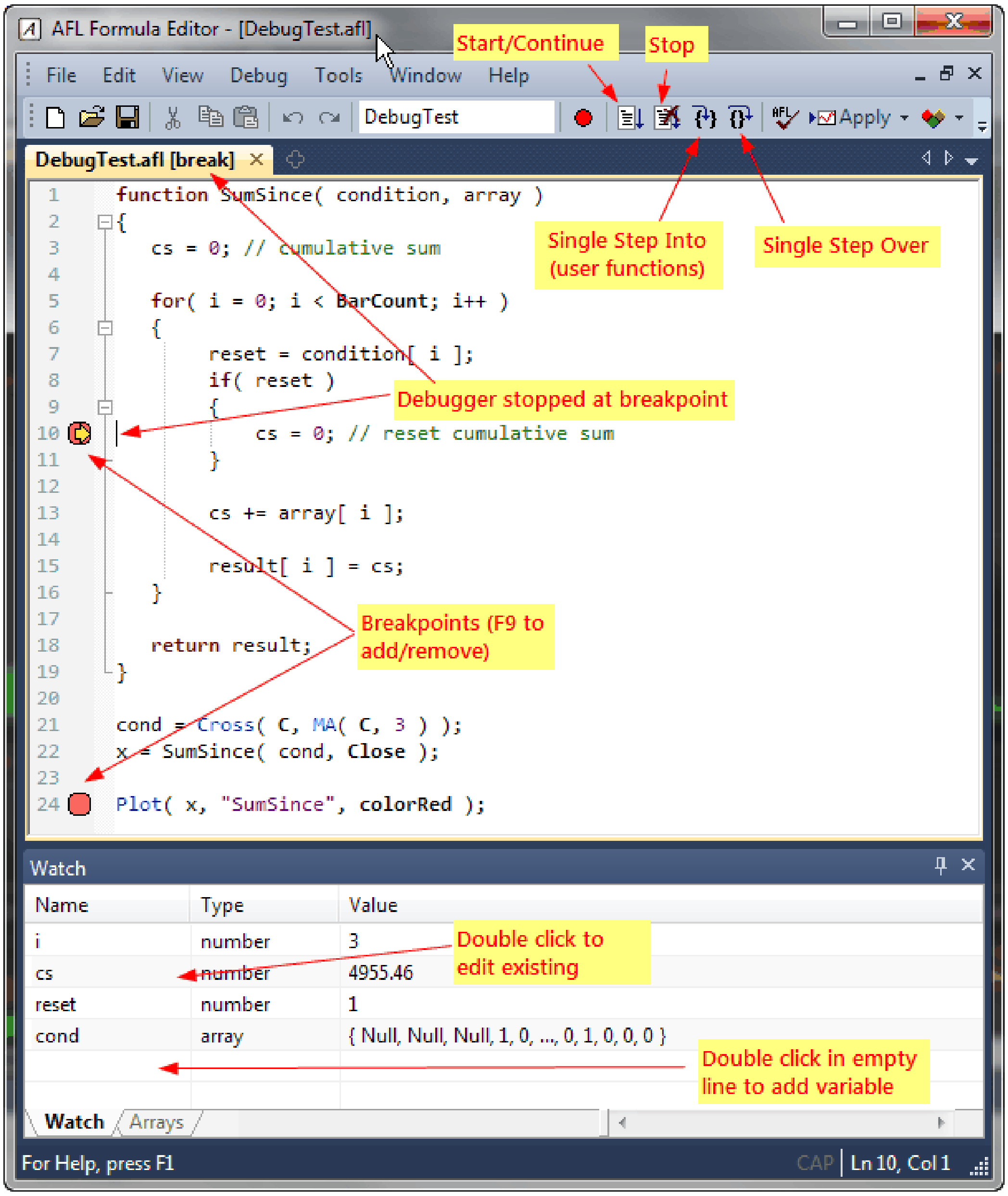
Now, start debugging again (**Debug->Go**). Execution should stop at line marked with breakpoint sign, before this line's code actually executes. A yellow arrow shows current position of execution as shown in the picture below.

Single stepping

You can cause the code to execute by stepping ahead (**Debug->Step Over**, **F10** key). You could also click **Debug->Step Into** (**F11** key) option. The difference between **Step Into** and **Step Over** is when you try to single-step a call to user-defined function.

If the line contains a call to user-defined function:

- **Step Into** executes the call itself and stops at the first line of code inside the function
- **Step Over** executes the called function as one step without stopping inside the function


AFL Formula Editor - [DebugTest.afl]
File Edit View Debug Tools Window Help
DebugTest
Apply
Start/Continue
Stop
Single Step Into (user functions)
Single Step Over
DebugTest.afl [break]
Debugger stopped at breakpoint
Breakpoints (F9 to add/remove)
Watch
Double click to edit existing
Double click in empty line to add variable
Watch Arrays
For Help, press F1
CAP Ln 10, Col 1

```
function SumSince( condition, array )
{
   cs = 0; // cumulative sum

   for( i = 0; i < BarCount; i++ )
   {
       reset = condition[ i ];
       if( reset )
       {
          cs = 0; // reset cumulative sum
       }

       cs += array[ i ];

       result[ i ] = cs;
   }

   return result;
}

cond = Cross( C, MA( C, 3 ) );
x = SumSince( cond, Close );

Plot( x, "SumSince", colorRed );
```

Name	Type	Value
i	number	3
cs	number	4955.46
reset	number	1
cond	array	{ Null, Null, Null, 1, 0, ..., 0, 1, 0, 0, 0 }

Examining contents of the variables

When debugger is stopped at breakpoint or stopped by single-stepping, you can examine the contents of variables anywhere in the code. To quickly check the content of any variable simply **hover the mouse over the variable**. After less than one second a tooltip will appear showing type and value of the variable.

The Watch window

If you are interested in watching several variables easily you can use **Watch** window. To open Watch window use **Window->Watch** menu.

To add a new variable to Watch window **double click on empty row** and type the variable name. You can also simply select the variable in the editor and drag it over the Watch window.

To edit existing variable in the Watch window **double click on its name** and type the variable name.

To delete the variable from Watch window, click on the name once (select it) and **press DELETE key**.

In addition to variables, you can use entire expression in the Watch window. You can either drag-drop expressions from editor onto Watch window, or type the expression after double clicking as you would with variables. The expression evaluator used by Watch window supports the following:

- all arithmetic operators like +, -, *, /
- parentheses like (High+Low)/2
- array subscript expressions with variables and expressions as array indexes such as Close[i + 2]
- matrices and matrix expressions with dynamic subscripts, like mat[i + 1][j]
- mixing various types of variables such as concatenating string and numeric value like this "Value is = " + Close[i + 2]

Note that only identifiers that can be used in Watch window are variable identifiers. You can not use function identifiers, so you can't call functions from the Watch window.

The values entered in the Watch window are watched for changes and if change in value occurs between two debugging steps, given value is highlighted with yellow background. Numeric (scalar) values are displayed in green when they increased or red if they are decreased.

Arrays tab

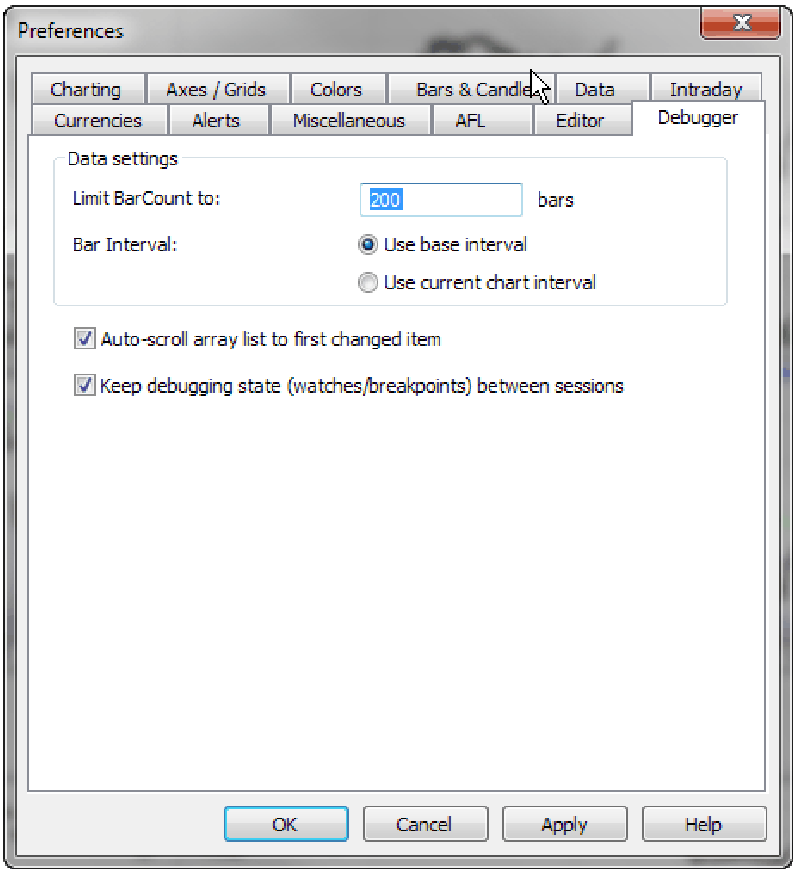
The array tab in the Watch window shows exploration-like array output for detailed examination of array contents (first 20 arrays from watch window are reported). It is very convenient as arrays are usually large and having them in the table format together with bar number and corresponding date/time stamp makes it a lot easier to see their contents. Additionally each element of each array is watched for changes and individual item change is highlighted and auto-scroll occurs so first changed item is always visible in the 'Arrays' tab.

Output window

When you run the code in the debugger, you can use the Output window (open it using Window->Output menu) to display all texts that are printed using printf() function. When formula finishes its execution the text "Debug session has ended" is displayed in the Output window.

Debugger preferences

Tools->Preferences window contains now a dedicated page with Debugger settings.



Data settings

Limit BarCount to: 200 bars

Bar Interval: (●) Use base interval ( ) Use current chart interval

[✓] Auto-scroll array list to first changed item

[✓] Keep debugging state (watches/breakpoints) between sessions

The settings are as follows:

- **Limit BarCount to** - defines maximum number of bars in arrays (BarCount) used during debugging (**by default it is 200 bars**)
- **Bar interval** - decides if debugger uses Base time interval or current Chart interval. The default is "**Use base time interval**" because there can be no chart open at all!
- **Auto-scroll to first changed item** - when this is ON, the "Arrays" list in the Watch window is scrolled automatically to first array item that has changed (so you don't need to locate elementsm that changed manually)
- **Keep debugging state** - when this is ON, AFL editor saves the debug state (breakpoints and watches) and bookmarks in the .dbg file along with the formula when closing the editor and restores the state when formula is reopened. Note that .dbg file is automatically deleted by AmiBroker when there are no breakpoints, no bookmarks and no watches.

Keyboard shortcuts

The following keyboard shortcuts are available in the debugger mode:

- F5 - Start/Continue debugging
- F9 - Toggle Breakpoint
- F10 - Step Over
- F11 - Step Into

Tips & tricks

If you want to stop execution when certain condition is met, put the breakpoint inside the if statement like this:

```
      if( reset > 0 AND param > 4 )
      {
          cs = 0; // PUT BREAKPOINT IN THIS LINE and it will stop only when
condition is met
      }
```

Breakpoints currently work with:

- regular statements (that end with semicolon). For multi-line statements place breakpoint at the beginning line of the statement
- for loops
- while loops
- do-while loops (you need to place breakpoint where 'while' clause is located, it won't break at the 'do' line as it essentially is no-op, if you want to break at the beginning of do, just place breakpoint on first statement inside { block }
- if statements
- return statements
- switch/case statements
- break statements

Breakpoints that you place on other lines, won't trigger. The AFL editor won't allow to place breakpoint on empty line, or line that beginnins with // comment or sole brace

You need to be also aware about the fact that breakpoints placed inside conditional statements like if() are only hit when given condition is met. So for example if you have code like below, the code inside 'if' statements won't execute unless it is actually run in the chart (indicator) or in portfolio backtest in New Analysis window.

```
if( Status("action") == actionIndicator )
{
   // breakpoint placed here won't trigger under debugger because condition above
is not met
}

if( Status("action") == actionPortfolio )
{
   // breakpoint placed here won't trigger under debugger because condition above
is not met
}
```

AmiBroker executes code in various contexts indicated by "action" status. Parts of the code can be conditionally executed in certain contexts using statements like shown above. This allows for example to call Plot() only in charts and access backtest object only when it is available.

As of version 6.10, the debugger runs the code with **actionBacktest** context. To debug code that is run conditionally in other contexts, you can temporarily disable the conditional check, but please note that you still can not access certain objects because they are simply not available. Specificaly GetBacktesterObject() when called anywhere outside second phase of portfolio backtest within New Analysis would return empty (Null) object. In the future this limitation may be removed.

Using on-chart GUI controls

Basics

AmiBroker 6.30 brings up ability to create and utilise user-definable GUI controls such as buttons, checkboxes, radio buttons, edit fields, date time pickers, sliders, etc that can be placed on chart and interacted with.

Creation of GUI controls

On-chart GUI controls can be created by calls to appropriate functions like: GuiButton, GuiEdit, GuiToggle, GuiRadio, GuiDateTime, GuiSlider.

All UI creation functions (GuiButton, GuiEdit, GuiToggle, GuiRadio, GuiDateTime, GuiSlider) take control ID as parameter. Control ID is a **unique identifier** that is used to distinguish between controls and to manipulate given control. Control ID has to integer number greater than zero. As the code below shows, it is good idea to assign some meaningful names to control IDs and and use them instead of plain numbers.

The following example shows how to create a button:

```
// control IDs
idMyButton = 2;

GuiButton( "MyButton", idMyButton, 10, 20, 100, 20, notifyClicked );
```

The operation model of on-chart GUI controls is such that controls can be destoroyed at any time when for example user switches to different chart sheet. For this reason the formula needs to be able to re-create UI controls if needed. This is done by simply calling creation functions each time formula is run. The creation functions are intelligent enough not to create duplicates.

If control with given ID already exits, it will be checked if it is the same. If already existing control of the same ID and the same type is found, nothing is done, and GUI creation function returns **guiExisting** value. If control exists but has different type, it will be destroyed and control of new type will be created. If control with given ID does not exist it will be created. In such cases the creation function will return **guiNew**. The code below shows how to check return value:

```
// control IDs
idMyButton = 2;

rc = GuiButton( "MyButton", idMyButton, 10, 20, 100, 20, notifyClicked );

if( rc == guiNew ) _TRACE("Control just created");
if( rc == guiExisting ) _TRACE("Control already existing");
```

It is not really needed for buttons that don't require initialization, but it is useful when you would like to set up some initial values for controls only when they are created. The following example shows how to set initial text for edit field:

```
// control IDs
idEdit = 1;
```

```
function CreateGUI()
{
   rc = GuiEdit( idEdit, 10, 20, 100, 20, notifyEditChange );

   if( rc == guiNew ) GuiSetText("Initial text", idEdit );

}

CreateGUI();
```

Handling events / UI notifications

GUI controls are created to allow the user to interact with them. When this happens (for example when user clicks on a button) an event is created and the formula gets executed again. Several controls may trigger events on several occasions, for example when clicking on it, editing text, gaining or losing focus, etc. To decide whenever given control triggers events or not we use notify parameter in the Gui creation functions. For example:

```
GuiButton( "MyButton", idMyButton, 10, 20, 100, 20, notifyClicked );
```

will create a button that will trigger event when it is clicked. We can combine various flags using | (binary OR operator) if we want to be notified about many different events by given control for example, the code below will create an edit control that will notify us when its contents is changed and when edit field loses focus:

```
GuiEdit( idEdit, 10, 20, 100, 20, notifyEditChange | notifyKillFocus );
```

There are many possible notifications including:

- notifyClicked - when button/toggle/checkbox/radio is clicked
- notifySetFocus - when control gets input focus (via mouse or keyboard)
- notifyKillFocus - when control loses input focus (via mouse or keyboard)
- notifyHitReturn - when ENTER/RETURN key was pressed in edit field
- notifyEditChange - when control value changes (text inside edit field, date inside date/time, slider is moved to new position)
- notifySelChange - when selection changes in the control (for example selected text range in the edit field)
- notifyMouseEnter - when mouse hovers over the control
- notifyMouseLeave - when mouse leaves control area

When an event triggers, our formula is executed again. If multiple events occur, they are placed in the queue. To handle such UI events the formula should call GuiGetEvent() function to find out if there are any events in the queue and to handle the appropriately.

GuiGetEvent(*num*, *what* = 0) function has two parameters:

num parameter defines the index of notification in the queue. 0 is first received (or "oldest") since last execution. The second parameter, *what,* defines what value is returned by GuiGetEvent function:

- what = 0 - ID of control that received event (number)
- what = 1 - the notification code (number)
- what = 2 - the ID, the code and the notification description as text (string)

Usually there is zero (when formula execution was not triggered by UI event) or one event in the queue but note that there can be more than one event notification in the queue if your formula is slow to process. To retrieve them all use increasing "num" parameter as long as GuiGetEvent does not return zero (in what =0, =1 mode) or empty string (what=2).

```
// read all pending events
for( i = 0; id = GuiGetEvent( i ); i++ )
{
    code = GuiGetEvent( i, 1 );
    text = GuiGetEvent( i, 2 );
}
```

It is good practice to have a separate function that handles the events as shown below:

```
idMyFirstButton = 1;
idMySecondButton = 2;

function CreateGUI()
{
     GuiButton( "enable", idMyFirstButton, 10, 60, 100, 30, notifyClicked );
     GuiButton( "disable", idMySecondButton, 110, 60, 100, 30, notifyClicked );
}

function HandleEvents()
{
    for ( n = 0; id = GuiGetEvent( n, 0 ); n++ ) // get the id of the event
    {
         code = GuiGetEvent( n, 1 );

         switch ( id )
         {
             case idMyFirstButton:
             // do something
                break;

             case idMySecondButton:
             // do something else
                break;

             default:
                 break;
         }
     }
}

CreateGUI();

HandleEvents();
```

Manipulating UI controls

Controls can be manipulated in a number of ways:

All controls can be hidden using and shown back again using GuiSetVisible() function:

```
GuiSetVisible( id, False ); // hides the control
GuiSetVisible( id, True ); // shows the control
```

All controls can be disabled (so they don't react to user input and become 'grayed') and enabled back again using GuiEnable() function:

```
GuiEnable( id, False ); // disables the control
GuiEnable( id, True ); // enables the control
```

You can set font for all GUI controls using GuiSetFont call:

```
GuiSetFont("Tahoma", 13 );
```

Buttons (as all other controls) by default use system colors. But buttons can have their own custom colors defined using GuiSetColors (other controls only use system colors).

```
GuiButton( "First", idFirst, 0, 50, 200, 30, 7 );
GuiButton( "Second", idSecond, 200, 50, 200, 30, 7 );
GuiButton( "Third", idThird, 400, 50, 200, 30, 7 );
// two first buttons will have red text, border and black background
GuiSetColors( idFirst, idSecond, 2, colorRed, colorBlack, colorRed );
// and the third green text, border and blue background
GuiSetColors( idThird, idThird, 2, colorYellow, colorBlue, colorYellow );
```

In this simple example above, we are just setting normal color of buttons, but they also have separate colors for different states (selected, hovered) and they can be set too.

All controls can have their text updated via GuiSetText and retrieved using GuiGetText .

```
// control IDs
idEdit = 1;

function CreateGUI()
{
   rc = GuiEdit( idEdit, 10, 20, 100, 20, notifyEditChange );

   if( rc == guiNew ) GuiSetText("Initial text", idEdit );

}
```

If you rather prefer numeric value of text in the control you can use GuiSetValue()/GuiGetValue() pair (especially useful for sliders to get their position). Sliders can also have their min-max range defined using GuiSetRange() as shown in example below.

```
idSlider = 1;

if ( GuiSlider( idSlider, 10, 30, 200, 30, notifyEditChange ) == guiNew )
{
   // init values on control creation
   GuiSetValue( idSlider, 5 );
   GuiSetRange( idSlider, 1, 100, 0.1, 100 );
}
```

Preserving state of controls

Basic controls such as push buttons don't have 'permanent' state. They are only used to trigger some actions when user pushes the button. But some controls have their state (additional information stored in them). For example the text entered in the edit field, the position of the slider, date picked in the datetime control, checkbox state, etc.

Each control (including on-chart controls) in Windows OS is actual "window" object. **Controls** (window objects in general) **preseve their state on their own as long as they "live", i.e. as long as they exist as "window objects", which means as long as given chart pane is displayed.**

So as long as you don't care about preserving state when chart is not displayed, you don't need to do anything. Otherwise follow steps below.

The thing is that on-chart GUI controls are dynamically created by your code when formula is executed. They can be automatically destroyed if you close chart window or if you switch to another sheet. Then they can be re-created next time formula executes.

This fact has one consequence - if you want to keep your control state between "destory" and "recreation", you have to store the state yourself. This can be done using for example static variables (or permanent static variables if you want to keep the state between AmiBroker runs).

As discussed earier, control creation functions such as GuiButton, GuiEdit, GuiDateTime, GuiCheckbox, GuiRadio, GuiSlider, etc, all return **guiNew** value if control is newly created or **guiExisting** if given control (window object) already exists. This allows us to know when we need to restore previously saved state. For example assuming that we have edit control text saved in the "mySavedText" static variable, we could restore the state using code like below:

```
// control IDs
idEdit = 1;

function CreateGUI()
{
     rc = GuiEdit( idEdit, 10, 20, 100, 20, notifyEditChange );

     if( rc == guiNew ) GuiSetText( StaticVarGetText( "mySavedText" ) , idEdit );

}
```

Now we may also want to store the text to static variable whenever it changes:

```
function HandleEvents()
{
     for( n = 0; id = GuiGetEvent( n, 0 ); n++ )
     {
         switch( id )
         {
             case idEdit:
                 StaticVarSetText( "mySavedText", GuiGetText( idEdit ), True /*
make it persistent */ );
                 break;
                 /// the rest of the event handling code ....
        }
     }
}
```

As we store the value of edited text on each change into persistent static variable it will be preserved between AmiBroker runs and restored whenever edit control is (re-)created.

The same technique can be used for other controls as well.

Examples

Example code below shows many techniques described above, including setting inital values on creation, modification of control properties, enabling/disabling controls, showing/hiding controls, handling GUI events.

```
// control identifiers
idSlider = 1;
idEnable = 2;
idDisable = 3;
idShow = 4;
idHide = 5;

function CreateGUI()
{
     if ( GuiSlider( idSlider, 10, 30, 200, 30, notifyEditChange ) == guiNew )
     {
        // init values on control creation
        GuiSetValue( idSlider, 5 );
        GuiSetRange( idSlider, 1, 100, 0.1, 100 );
     }

     GuiButton( "enable", idEnable, 10, 60, 100, 30, notifyClicked );
     GuiButton( "disable", idDisable, 110, 60, 100, 30, notifyClicked );
     GuiButton( "show", idShow, 10, 90, 100, 30, notifyClicked );
     GuiButton( "hide", idHide, 110, 90, 100, 30, notifyClicked );
}


function HandleEvents()
{
    for( n = 0; id = GuiGetEvent( n, 0 ); n++ )
    {
       switch ( id )
       {
          case idEnable:
             GuiEnable( idSlider, True );
             break;

          case idDisable:
             GuiEnable( idSlider, False );
```

```
                break;

            case idShow:
                GuiSetVisible( idSlider, True );
                break;

            case idHide:
                GuiSetVisible( idSlider, False );
                break;
        }
    }
}

CreateGUI();
HandleEvents();

Title = "Value = " + GuiGetValue( idSlider );
```

Caveats

Please be resonable with Gui* functions and be aware of Windows limits. As all controls in Windows (buttons, edit boxes, etc) are actual Window objects they are subject to Windows limitation. And there is a limit of 10000 windows PER PROCESS. So don't try to create thousands of buttons (like a button for every bar of data) because first you will see huge performance decrease and next you will hit the limit and run into problems (crash)
https://blogs.msdn.microsoft.com/oldnewthing/20070718-00/?p=25963

Best practice is to keep the number under 100-200. If you need more consider using low-level graphics instead.

Video Tutorials (on-line)

The following are video tutorials showing various areas of AmiBroker.

Charting

- How to customize price chart
  CustomizePrice.mp4
- How to use drag-drop charting
  dragdrop1.mp4
- How to display charts of two symbols simultaneously
  TwoSymbols.mp4
- Comparing two symbols on one chart:
  TwoSymbolsOneChart.mp4
- How to display Volume At Price:
  VolumeAtPrice.mp4
- How to create custom indicator:
  CustomIndicator.mp4

Data

- How to download free data from US markets using AmiQuote
  amiquote3.mp4
- How to setup AmiBroker with free real time data from Interactive Brokers
  ib.mp4
- How to configure with IQFeed data
  IQFeed.mp4
- How to configure with eSignal data
  esignal.mp4
- How to download free forex data:
  forex.mp4
- How to configure with Quotes Plus:
  qp3.mp4
- How to configure with TC2000/TCNet
  tc2k.mp4

User Interface

- How to use docking windows
  DockingWindows.mp4

- How to add define custom intraday interval
  CustomTimeInterval.mp4

- How to add Report Explorer as a custom tool / button:
  RepExInToolbar.mp4

- How to customize user interface
  uicustomize.mp4

- How to resize symbol entry box
  uiresizeticker.mp4

- How to use layers
  layers.mp4

- How to use layouts
  layouts.mp4

Analysis

- How to run system backtest (UPDATED)
  BackTest.mp4

- How to create your own explorations (4.80)
  exploration.mp4

- How to optimize parameters of trading system (4.80)
  optimize.mp4

For more video tutorials please check: https://www.amibroker.com/support.html

AmiBroker Reference Guide

Windows

This part describes functionality of AmiBroker windows.

All these windows are asynchronous i.e. you can open as many windows as you like, and work with all of them at the same time.

Charting

- Chart window pane
- Data window
- Parameters window
- Study drawing tools
- Line study properites window
- Text box properties window
- Formula editor
- Risk-yield map window
- Place order window

Settings

- Database settings / Intraday settings
- Preferences
- Customize tools window

Symbol / Data

- Symbol tree
- Information window
- Notepad window
- Quote Editor window
- Symbol finder window
- Finance window
- Profile view
- Assignment organizer window
- Composite calculator window
- Categories window
- Import Wizard window
- Metastock importer window
- Real-time Quote window
- Easy Alerts window
- Time/Sales window
- Bar Replay window

Analysis/Tools

- Formula editor
- Code Snippets window
- Quick review window
- Analysis window
- Filter settings window

Chart window pane

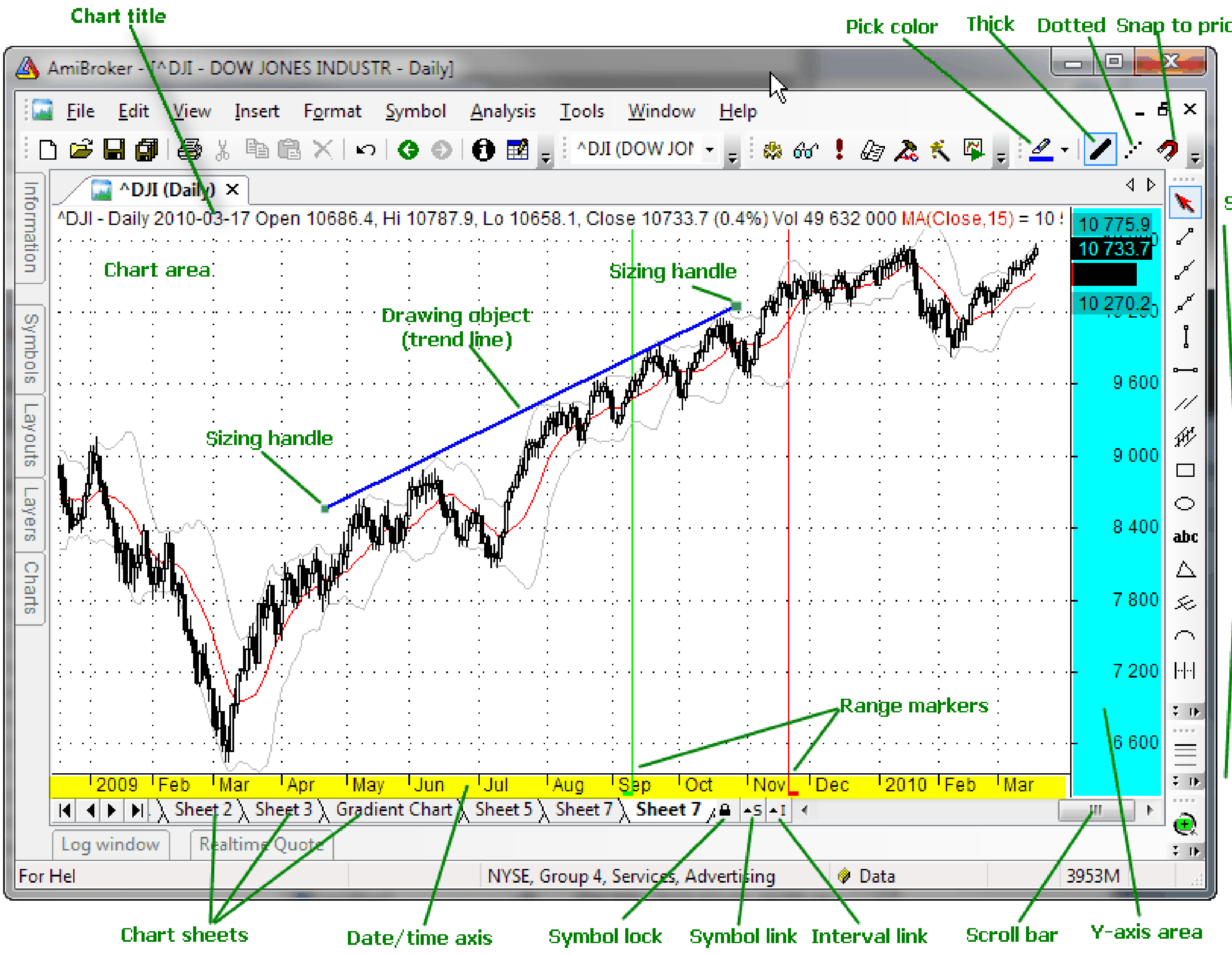


Chart title
Pick color
Thick
Dotted
Snap to pric
AmiBroker - [^DJI - DOW JONES INDUSTR - Daily]
File Edit View Insert Format Symbol Analysis Tools Window Help
^DJI (Daily)
^DJI - Daily 2010-03-17 Open 10686.4, Hi 10787.9, Lo 10658.1, Close 10733.7 (0.4%) Vol 49 632 000 MA(Close,15) = 10
Chart area
Sizing handle
Drawing object (trend line)
Sizing handle
Range markers
Information
Symbols
Layouts
Layers
Charts
2009 Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec 2010 Feb Mar
Sheet 2 Sheet 3 Gradient Chart Sheet 5 Sheet 7 Sheet 7
Log window
Realtime Quote
For Hel
NYSE, Group 4, Services, Advertising
Data
3953M
Chart sheets
Date/time axis
Symbol lock
Symbol link
Interval link
Scroll bar
Y-axis area

This window shows the chart of different technical indicators.

In the bottom of the chart you can see X axis, depending on Parameter window setting it may or may not display dates, and below you can see scroll bar and chart sheets tab control. Scroll bar can be used to display past quotes, while sheet tab allows to view different chart pages/sheets (click here to learn more about chart sheets).

To the right you can see Y-axis area (marked with blue color) that shows Y-scale and value labels. Value labels are color fields that display precisely the "last value" of plots. "Last value" is the value of the indicator (or price) for the last currently displayed (rightmost) bar. Y-axis area is used also to move/size chart vertically.

Chart parameters and settings can be adjusted by clicking with RIGHT MOUSE button over chart and choosing **Parameters** option from the chart context menu.

Chart can also be scrolled, resized, moved, shrinked, resized - to learn more about it please read Tutorial: Basic Charting Guide.

Parameters window

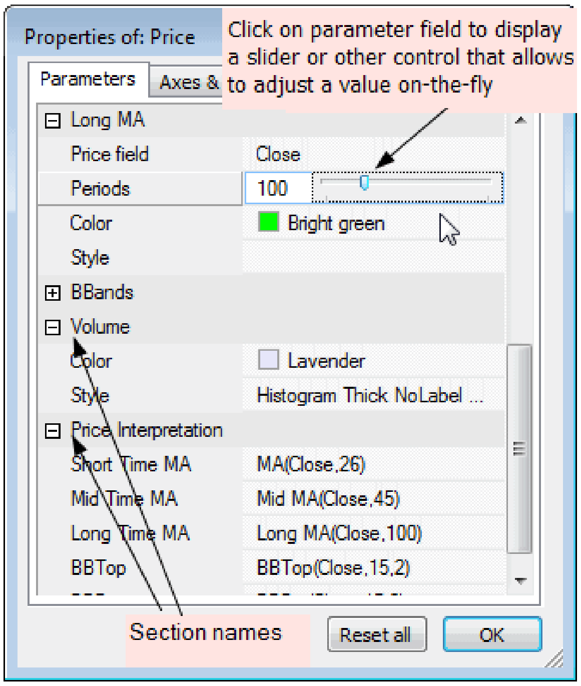
This window allows the user to modify parameters specified in the AFL formula via Param, ParamStr, ParamColor, ParamStyle, ParamField, ParamToggle, ParamDate, ParamTime, ParamList functions and also to adjust axes and grid settings.

It is accessible via chart context menu (right click the mouse over the chart pane to see the context menu) : choose **Parameters** and a small window with parameter list will appear. To edit parameter value simply click on the item value field as shown in the picture. Then depending on type of the parameter appropriate control(s) will appear.

For example, if given parameter is a string then text field will appear, and if given parameter is color then color-picker control will allow you to change the color.

When editing numeric parameters you can adjust the value by either entering the value to the edit field or by moving a slider control. To show the edit field - click on the number itself (marked with blue color in the picture below). To show a slider control click next to the number (right-hand side).

If given parameter is a number then slider or the edit field will be shown as in the picture below:


Properties of: Price
Click on parameter field to display a slider or other control that allows to adjust a value on-the-fly
Parameters
Axes &
Section names
Reset all
OK

⊟ Long MA	
Price field	Close
Periods	100
Color	🟩 Bright green
Style	
⊞ BBands	
⊟ Volume	
Color	⬜ Lavender
Style	Histogram Thick NoLabel ...
⊟ Price Interpretation	
Short Time MA	MA(Close,26)
Mid Time MA	Mid MA(Close,45)
Long Time MA	Long MA(Close,100)
BBTop	BBTop(Close,15,2)

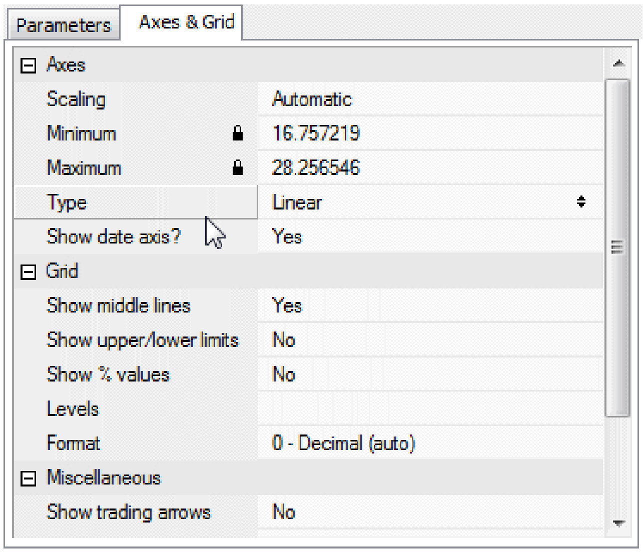
You can move the slider using mouse, <- -> cursor keys and mouse wheel. As changes are made underlying chart is immediately refreshed giving great feedback for the user.

Parameters are grouped into "sections". Sections represent part of the codes surrounded by _SECTION_BEGIN/_SECTION_END markers. To learn more about this check Tutorial: Using drag-and-drop interface.

At any time you can press **Reset all** button that will reset all parameters to default values.

For more information on using parameters please read Tutorial: Using colors, styles, titles and parameters in the indicators and Tutorial: Using drag-and-drop interface.

Parameter window allows also to control axes and grid appearance as well as some other per-chart settings. These controls are available in the second **Axes & Grid** tab as shown below:



⊟ Axes	
Scaling	Automatic
Minimum	16.757219
Maximum	28.256546
Type	Linear
Show date axis?	Yes
⊟ Grid	
Show middle lines	Yes
Show upper/lower limits	No
Show % values	No
Levels	
Format	0 - Decimal (auto)
⊟ Miscellaneous	
Show trading arrows	No

The following options are available:

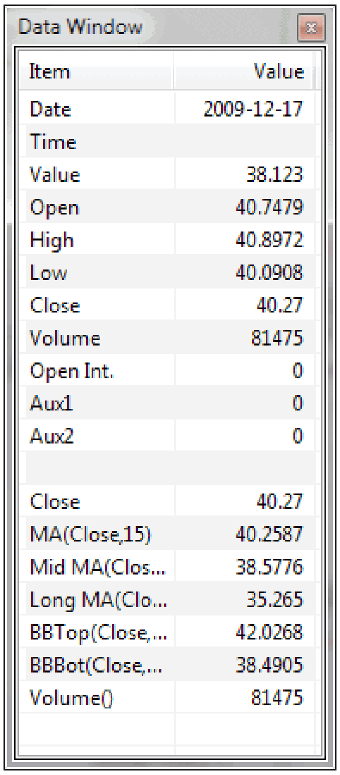
- **Axes**
    - **Scaling**:
        - **Automatic** - minimum and maximum value of Y axis is determined automatically by AmiBroker
        - **Custom** - minimum and maximum value of Y axis are user-defined
    - **Minimum** - minimum Y axis value (this property is locked if automatic scaling is selected, to unlock choose Custom scaling)
    - **Maximum** - maximum Y axis value (this property is locked if automatic scaling is selected, to unlock choose Custom scaling)
    - **Type**
        - **Linear** - use linear Y axis scale
        - **Logarithmic** - use logarithmic Y axis scale
    - **Show date axis** - turn on/off date display on X axis
- **Grid**
    - **Show middle lines** - display automatic Y axis grid lines spaced evenly between minimum and maximum
    - **Show upper/lower limits** - display minimum and maximum Y axis value labels
    - **Show % values** - display values as percents
    - **Levels** - allows to turn on grid lines at some fixed, popular levels such as 30/70, 20/80, 10/90, -100/+100, 0
- **Miscellaneous**
    - **Show trading arrows** - when turned ON this pane will show buy/sell/short/cover arrows generated by corresponding options available from Automatic Analysis menu.
    - **File path** (locked) - shows the path to the formula file that given chart uses
    - **Chart ID** (locked) - shows the numeric value of Chart ID given pane uses. Chart ID does not matter unless you use Study() function in your formula(s).

Data window

Data window can be displayed using **Window->Data Window** menu

The Data Window shows the date/time and values of open, high, low, close, volume, open interest, aux1 and aux2 of the bar under the mouse cursor. It also shows mouse cursor Y-coordinate ("Value") expressed in terms of price corresponding to current mouse cursor location.

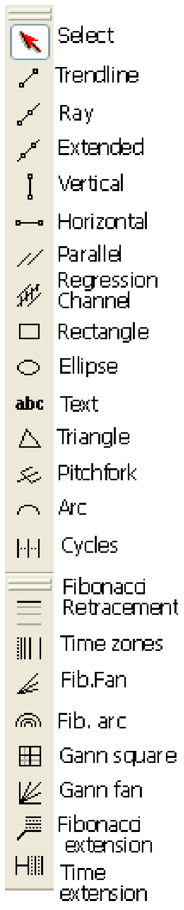
The Data Window also shows the values of all indicators defined in the formula. These values are automatically updated when cursor stops moving for a fraction of a second.


Data Window

Item	Value
Date	2009-12-17
Time	
Value	38.123
Open	40.7479
High	40.8972
Low	40.0908
Close	40.27
Volume	81475
Open Int.	0
Aux1	0
Aux2	0
Close	40.27
MA(Close,15)	40.2587
Mid MA(Clos...	38.5776
Long MA(Clo...	35.265
BBTop(Close,...	42.0268
BBBot(Close,...	38.4905
Volume()	81475

Study drawing tools

AmiBroker's study drawing tools are accessible from **Draw** / **Fibonacci & Gann** toolbars:



The following tools are available:

- trend line
- ray (new in 4.20)
- extended line (new in 4.20)
- vertical line
- horizontal line
- parallel lines (new in 4.20)
- Regression channels: Raff, standard deviation, standard error (all new in 4.20)
- Fibonacci Retracement study (enhanced in 4.20)
- Fibonacci Time zones study
- Fibonacci Extensions (new in 4.60)
- Fibonacci Time Extensions (new in 4.60)
- Fibonacci Fan
- Fibonacci arc
- Gann Square (new in 4.20)
- Gann Fan (new in 4.20)
- Ellipse tool
- Triangle tool (new in 4.30)
- Andrews' pitchfork (new in 4.30)
- Cycles tool (new in 4.60)
- Arrow tool (new in 4.70)
- Zig-zag tool (new in 4.70)
- Arc tool
- Rectangle
- text box tool

The default *Select* tool (red arrow) is used to select drawing objects and quotations on the chart. If you want to draw given study just switch on appropriate button and start drawing on the chart by pointing the mouse where you want to start the drawing and click-and-hold left mouse button. Then move the mouse. Study tracking line will appear. Release left mouse button when you want to finish drawing. You can also cancel study drawing by pressing ESC (escape) key. For beginners' guide to charting check Tutorial: Charting guide

Trend line, Ray, Extended, Vertical, Horizontal

These tools give different flavours of basic trend line. Trend line gives a line segment, Ray gives right-extended trend line, Extended gives trend line that is extended automatically from both left- and right-sides. Vertical and Horizontal are self-explaining.

Arrow

Similar to Trend line but ends with an arrow

Zig-zag

Draws a series of connected trend lines. To end drawing press ESC key.

Parallel

This tool allows to draw a series of parallel trend line segments. First you draw a trend line as usual, then a second line parallel to the first is automatically created and you can move them around with the mouse. Once you click on the chart it is placed in given position. Then another parallel line appears that can be placed somewhere else. And again, and again. To stop this please either press ESC key or choose "Select" tool.

Regression channels

AmiBroker allows to draw easily 3 kinds of regression channels:

- Raff regression channel
- Standard error channel
- Standard deviation channel

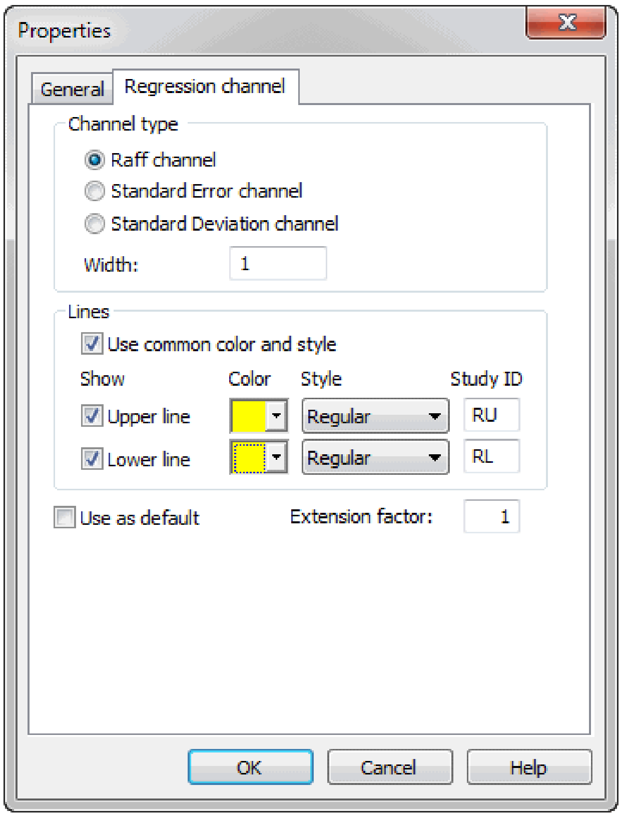
All these channels are based on linear regression trend line.

The Regression Channel is constructed by plotting two parallel, equidistant lines above and below a Linear Regression trendline. The distance between the channel lines to the regression line is the greatest distance that any one high or low price is from the regression line.

Standard Error Channels are constructed by plotting two parallel lines above and below a linear regression trendline. The lines are plotted a specified number of standard errors away from the linear regression trendline.

Standard Deviation Channels are constructed by plotting two parallel lines above and below a linear regression trendline. The lines are plotted a specified number of standard errors away from the linear regression trendline.

You can choose the type of channel by double clicking on the channel study (or choosing **Properties** from right mouse button menu)



Properties

General | Regression channel

Channel type

(●) Raff channel
( ) Standard Error channel
( ) Standard Deviation channel

Width: 1

Lines

[✓] Use common color and style

Show	Color	Style	Study ID
[✓] Upper line	🟨	Regular	RU
[✓] Lower line	🟨	Regular	RL

[ ] Use as default   Extension factor: 1

If **Use common color and style** box is marked channel lines use the same style and color as regression (middle) line. If it is not marked you can set separate colors and style for upper and lower channel line. You can also switch off completely upper and lower channel lines by unticking **Show Upper line** and **Show Lower line** boxes.

"**Study ID**" column defines study identifier that can be used in your custom formulas to detect crossovers. You can change these IDs if required by simple editing these fields. For more information on Study IDs check Tutorial: Using studies in AFL formulas
More information on regression channels is available from Technical analysis guide.

Ellipse and Arc drawing tools

These new drawing tools are connected to the date/price coordinates (as trend lines) rather than to the screen pixels so they can change the visual shape when displayed at various zoom factors or screen sizes.

To see the properties of these elements you should double-click on the clock-like 3, 6, 9 or 12 hour positions.

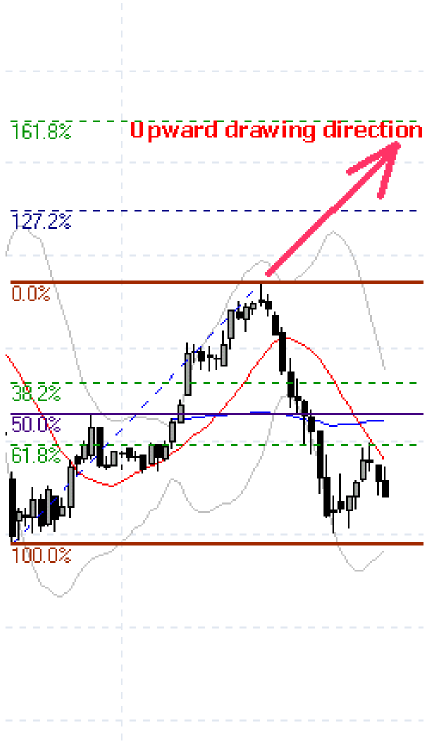
Fibonacci arc

This new drawing tool generates standard Fibonacci-arcs that are controlled by the trend line drawn with a dotted style. To see the properites of the arcs click on the controlling trend line.

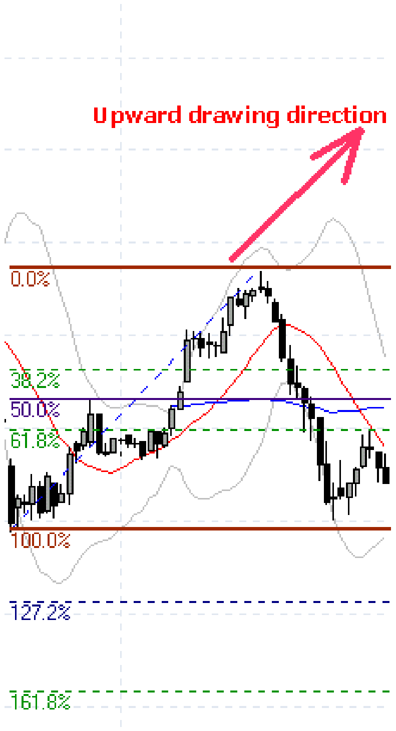
Note that arc radius and central point are relative to the controlling trendline and because Fibonacci arcs must be circular regardless of screen size/resolution and zoom factor the position of the arcs may move in date/price domain.

Fibonacci retracement

First please note that Fibonacci tool works differently depending on the direction of drawing and "show extensions" flag. See the pictures below.

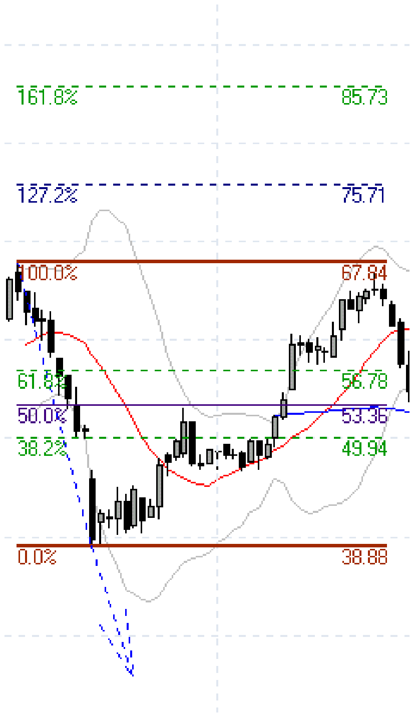


Upward drawing direction
Show Extensions ON

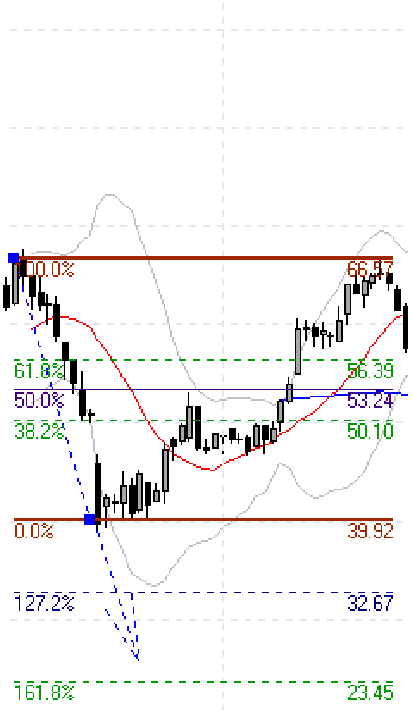


Upward drawing direction
Show Extensions OFF

as you can see it shows both retrace levels (38.2, 50, 61.8) and extension levels (127.2, 161.8). If "show extensions" box is OFF the tool shows ONLY retrace levels. It works in a similar way when controlling trend line is drawn downwards.

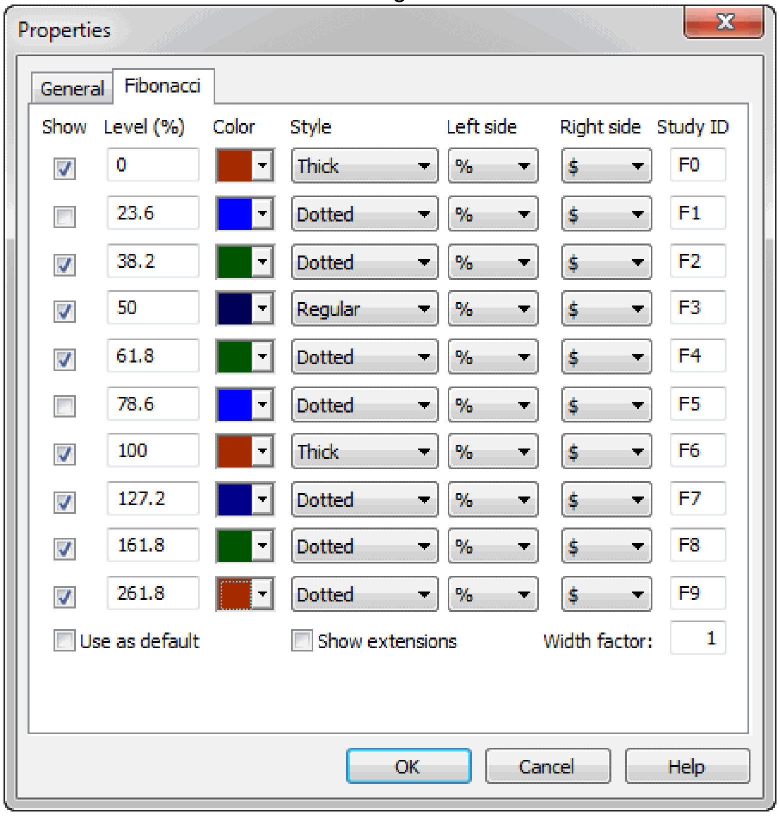


Downward drawing direction
Show Extensions ON



Downward drawing direction
Show Extensions OFF

Now more about Fibonacci settings window:



Show	Level (%)	Color	Style	Left side	Right side	Study ID
✓	0	🟥	Thick	%	$	F0
☐	23.6	🟦	Dotted	%	$	F1
✓	38.2	🟩	Dotted	%	$	F2
✓	50	🟦	Regular	%	$	F3
✓	61.8	🟩	Dotted	%	$	F4
☐	78.6	🟦	Dotted	%	$	F5
✓	100	🟥	Thick	%	$	F6
✓	127.2	🟦	Dotted	%	$	F7
✓	161.8	🟩	Dotted	%	$	F8
✓	261.8	🟥	Dotted	%	$	F9

☐ Use as default ☐ Show extensions Width factor: 1

First column "**Show**" switches particular line ON/OFF
Second column "**Level (%)**" defines percentage level. 100 and 0 represent Y-coordinate

of begin and end points of controlling trend line.
Third column "**Color**" defines color of the line, Fourth column "**Style**" allows
to choose between regular, thick and dotted styles.

Fifth and Sixth columns "**Left side**" and "**Right side**" control display of text
that appear on the left and right side of the Fibonacci level line. Empty - means no text,
% - means percentage level, $ - means dollar (point) level.

Seventh column "**Study ID**" defines study identifier that can be used in your custom formulas
to detect crossovers. Each Fibonacci level has a separate ID be default F0... F9.
You can change these IDs if required by simple editing these fields.

As described in User's Guide: Tutorial: **Using studies in AFL formulas**

you can easity write the formula that checks for penetration of particular Fibonacci level.
In this example we will detect if the closing price drops F2 (38.2% retracement) level line. The formula is very simple:

```
      sell = cross( study( "F2" ), close );
```

Note that study() function accepts two arguments: the first is StudyID two letter code that corresponds to one given in properites dialog; the second argument is chart ID - by default it is 1 (when it is not given at all) and then it references the studies drawn in the main price pane. For checking studies drawn in other panes you should use the codes given above (in the table describing study() function).
Please note that this formula is universal - it will use appropriate level from any symbol that has Fibonacci lines drawn.
This is so because AmiBroker keeps data of all studies drawn in its database.
When you scan using above code - AmiBroker checks if Fibonacci levels are drawn for symbol being currently scanned,
if it finds one - it looks what F2 study is - it finds that this is a fibonacci line 38.2% located (for example for particular symbol) at $29.06
so AmiBroker internally substitutes study("F2") by $29.06 (caveat: this is simplification - in fact it internally generates array that represents a trend line) and checks for cross.

"**Extension factor**" decides how far lines are right-extended (in X-axis direction). If you enter 2 you will get lines extended twice as much as default '1'. If you enter 0 Fibonacci level lines will end where controlling trend line ends.

"**Use as default**" - if you check this box and accept the settings by clicking **OK -** all Fibonacci drawings that you will draw later will use these settings.
When using text box tool just type the text in the box, when you want to finish click outside the text box. You can also cancel editing by pressing ESC key.

Fibonacci Extensions

The Fibonacci Extensions tool is similar to the Fibonacci Retracements tool. The Fibonacci Extensions tool requires a third point. The extensions and retracement levels are drawn from this third point, but based upon the distance between the first two points. A common use of this tool is to first connect two points that represent the endpoints of a major trend (or wave). Then choose the third point to be the endpoint of a retracement of that trend. Extensions are then drawn in the direction of the initial trend, from the third point, using the distance between points one and two as a basis for the extension levels.

The Fibonacci Extensions toolbar button and drawing tool work much like the Andrew's Pitchfork drawing tool. First, click on the Fibonacci Extension button on the toolbar. Then, click three times, once on each of the points that are involved in the Fibonacci Extension. The first click should be on the starting point of the initial trendline. The second click should be on the ending point of the initial trendline. The third click should be on the ending bar of the retracement period.

As with Fibonacci Retracements, there is a great deal of flexibility via Fibonnacci settings tab available after clicking on the study with a right mouse and selecting "Properties" from the context menu.

Fibonacci Time Extensions

Fibonacci Time Extensions tool is used to specify vertical lines at date/time levels which are determined to be probable values of changes in trend based on the market’s previous date/time range and a third extension point.

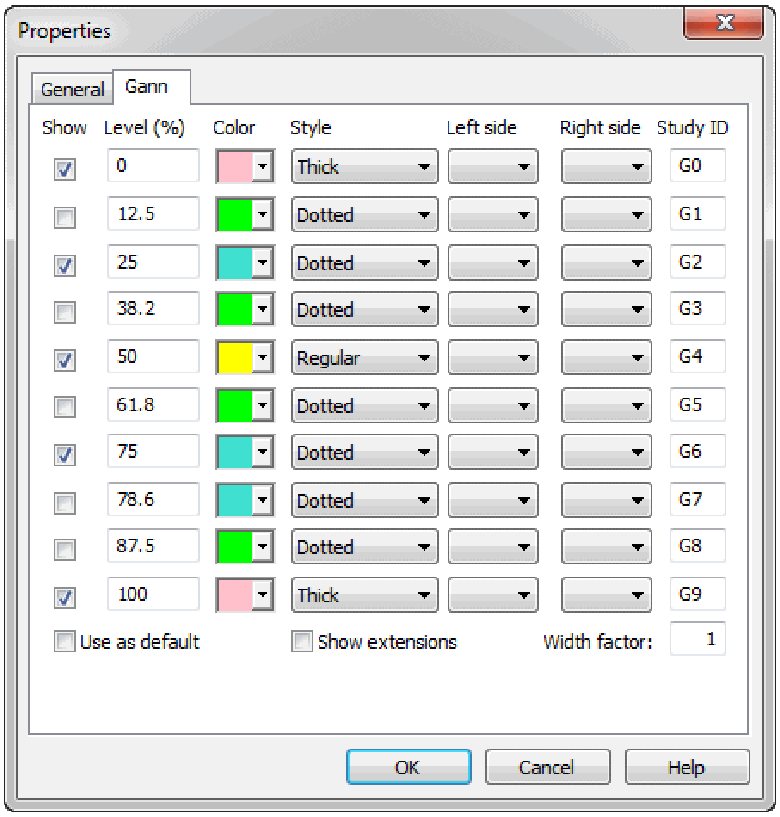
The time extension tool should be used as follows. First, click on the Fibonacci Time extension button on the toolbar. Then select the first range point (typically a major top or bottom of a market) by clicking on the chart where you want the range to begin, then move the mouse pointer to select the second range point by again clicking on the chart where you want the range to end. Extension lines now will be drawn onto the future bars.

As in Fibonacci Price retracement and extensions tools you have complete control over which percentages are used in the Time Extensions tool, and the colours of each of the extension values via Properites dialog.

Gann square and Gann Fan

Gann Squares indicate possible time and price movements from important highs and lows. To draw a Gann Square on a chart move the cursor on the chart to the starting point. The starting point is generally an important High or Low on the chart. Then drag the mouse to the right until a desired ending point is reached. The start and end points will be the corners of the square. The ending point is often to the right of the chart bars. Watch for trends to change directions at the Gann Square levels. As the Gann Square is drawn to the screen the angle of the controlling trend line is shown in the status bar.

Properties Window



Show	Level (%)	Color	Style	Left side	Right side	Study ID
✓	0	🟥	Thick			G0
	12.5	🟩	Dotted			G1
✓	25	🟦	Dotted			G2
	38.2	🟩	Dotted			G3
✓	50	🟨	Regular			G4
	61.8	🟩	Dotted			G5
✓	75	🟦	Dotted			G6
	78.6	🟦	Dotted			G7
	87.5	🟩	Dotted			G8
✓	100	🟥	Thick			G9

Use as default Show extensions Width factor: 1

The properties window is used to change the square levels, color, style, thickness, and defaults. Click on any of the Gann Square **Show** entries to add or remove lines. Click in the square **color** box to change the line color. Click on **style** combo boxes to change the line style. Check the **Use as Default** box to save the settings as the default for all subsequent Gann Squares that are drawn. "**Left side**" and "**Right side**" columns control display of text that appear on the left and right side of the Gann lines. Empty - means no text, % - means percentage level, $ - means dollar (point) level. "**Study ID**" column defines study identifier that can be used in your custom formulas to detect crossovers. You can change these IDs if required by simple editing these fields. For more information on Study IDs check Tutorial: Using studies in AFL formulas

Triangle tool

Triangle tool is self-explaining. Drawing a triangle is easy: left-click at the first point, hold down and drag to the second point, then release mouse button and drag to the third point and click once. The controlling triangle will become the pitchfork.

Andrews' Pitchfork

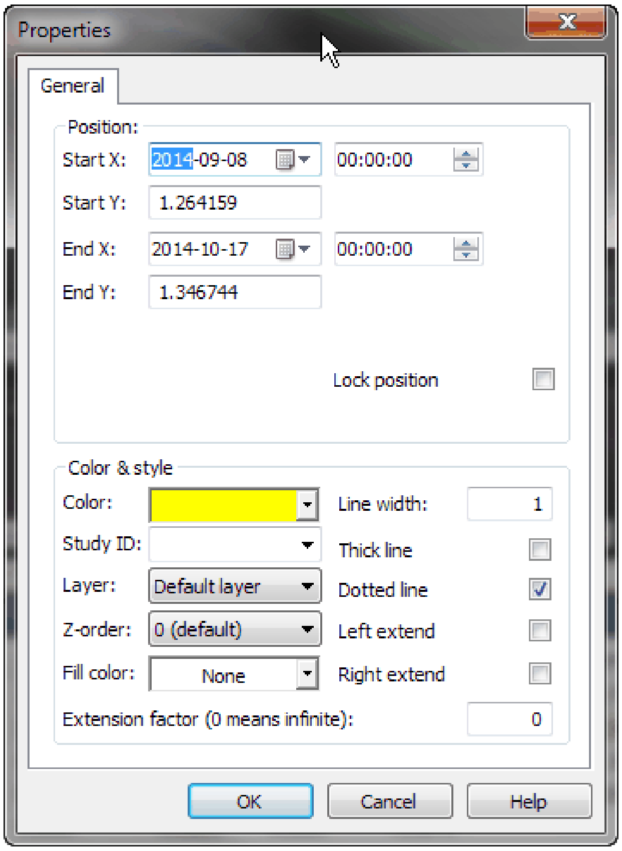
Andrews pitchfork is a study using parallel trendlines. In constructing the study, starting points are chosen. The first is a major peak or trough on the left side of the chart display. The second and third starting points are chosen to be a major peak and a major trough to the right of the first point. After all starting points have been decided, AmiBroker draws a trendline from the first point (the most left) so that it passes directly between the right most points. This line is called the handle of the pitchfork. The second and third trend lines are drawn by AmiBroker beginning at the starting points and parallel to the handle. Dr. Andrews suggested that prices make it to the median line (or handle) about 80% of the time while the price trend is in place. This means that while the basic long term price trend remains intact, Dr. Andrews believed that the smaller trends in price would gravitate toward the median line while the larger price trend remained in tact. When that does not occur, it may be evidence that a reversal in the larger price trend may be in progress or provides evidence of a stronger bias at work in market. When price fails to make it to the medial line from either side, it is often an expression of the relative enthusiasm of buyers and sellers and may predict the next major direction of prices. If prices fail to reach the median line while above the median line, it is a bullish and failing to reach the median line from below is bearish.

Operating Andrews' Pitchfork tool is similar to drawing triangle. Left-click at the first point, hold down and drag to the second point, then release mouse button and drag to the third point and click once. The controlling triangle will become the pitchfork.

Cycles tool

To use time cycles tool, click on the cycles drawing tool button in the toolbar then click at the starting point of the cycle and drag to the end of the cycle. These two control points control the interval between the cycle lines. When you release the mouse button you will get a series of
parallel lines with equal interval in between them.

Line study properties window



Position:
Start X: 2014-09-08 00:00:00
Start Y: 1.264159
End X: 2014-10-17 00:00:00
End Y: 1.346744
Lock position ☐

Color & style
Color: 🟨
Line width: 1
Study ID:
Thick line ☐
Layer: Default layer
Dotted line ✓
Z-order: 0 (default)
Left extend ☐
Fill color: None
Right extend ☐
Extension factor (0 means infinite): 0

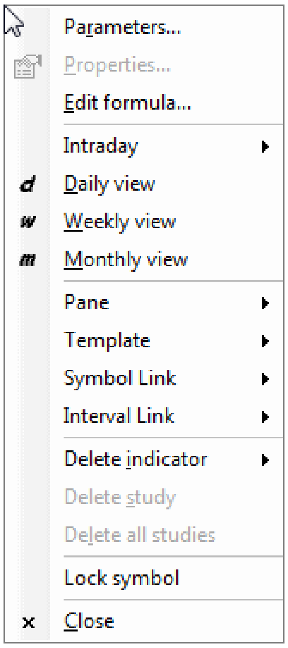
In the study properites window you can select start and end coordinates as well as line colours and styles. You can also enable automatic left- or right- line extension so that line will be extended when new quotes will be available.

There are following fields available:

- **Start X, Start Y, End X, End Y** - study start and end coordinates
- **Third X, Third Y -** visible only for TRI-POINT studies like triangle, pitchfork - the coordinates of 3rd control point of the study
- **Lock position** - if this field is marked it's impossible to chenge the position of the study with use of mouse
- **Color** - allows you to change the study color
- **StudyID** - defines Study ID which allows you to refer to the study from AFL formula.
  The detailed information is available in Using studies in your AFL formulas chapter.
- **Layer** - indicates the layer that the study is placed on.
  To learn more about layers read Working with layers.
- **Z-order** - defines the Z-order of the line. Lines, plots and graphics can be ordered in "Z" direction using Z-order. Learn more about this in Using Z-order tutorial.
- **Line width** - (new in 5.90) specifies line width in pixels. Default line width is 1 pixel.
- **Thick -** doubles the width of the line. The width is defined by **Line width** parameter. Turning this on makes the line twice as wide, so actual pixel width would be 2 * lineWidth.
- **Left / Right Extend** - you can choose whether line is extended
- **Extension factor -** (new in 5.90) - decides how far line is extended to left/right. Lines are extended in the direction of "X-axis" (i.e. date/time axis). 0 (zero) means infinite extension, one unit represents

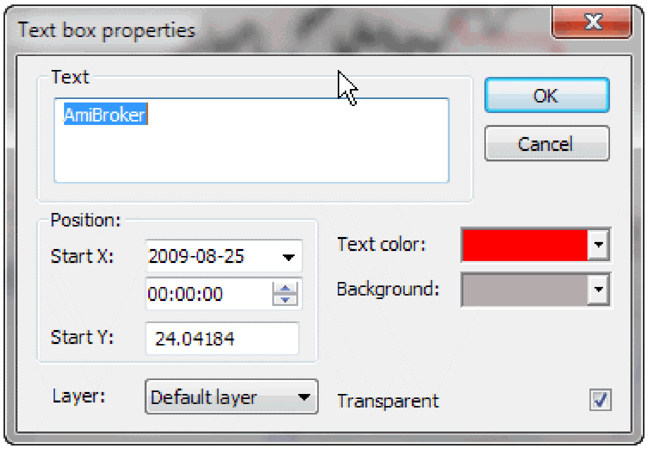
X-axis distance between study end and start points. Fractional values are allowed. Allowable range **0**...**25.5**.

Line study properites window is accessible from chart window's right mouse button menu. When you click on a study line with a right mouse button the following menu appears:



Simply choose **Properties** to show the line study window.

Text box properties window



Text: AmiBroker

Position:

Start X: 2009-08-25 00:00:00

Start Y: 24.04184

Text color: 🟥

Background: ▨

Layer: Default layer

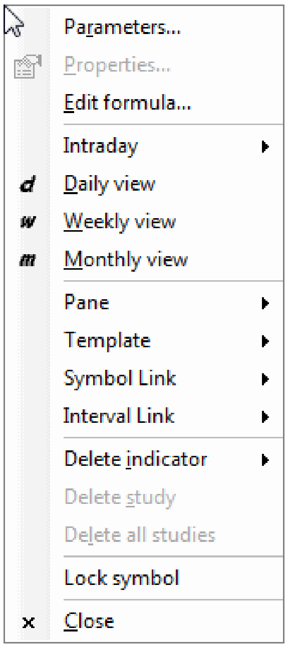
Transparent ✓

In the text box properites window you can change the text displayed in the box, select start co-ordinates as well as text and background colours and transparent style.

There are following fields available:

- **Start X, Start Y** - text coordinates
- **Color** - allows you to change the color of the text
- **Background Color** - allows you to change the color of the background
- **Layer** - indicates the layer that the text is placed on. To learn more about layers read Working with layers.

Text box properites window is accessible from right mouse button menu. When you click on a text box with a right mouse button the following menu appears:



Simply choose **Properties** to show the text box properties window.

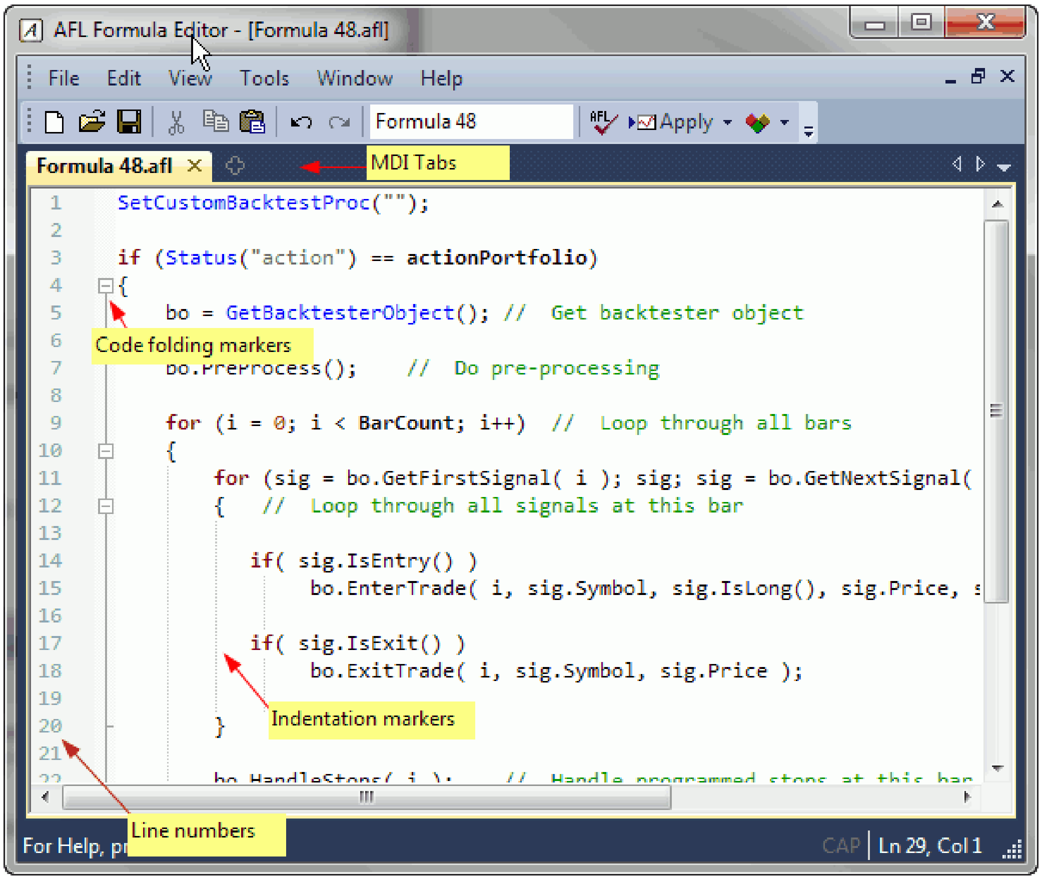
In the text box properites window you can change the text displayed in the box, select start co-ordinates as well as text and background colours and transparent style.

Formula Editor

A new AFL Formula Editor features:

- Syntax highlighting (improved in 5.80)
- Automatic brace matching/highlighting (NEW in 5.80)
- Auto indentation (NEW in 5.80)
- Indentation markers (NEW in 5.80)
- Enhanced auto-complete in two modes (immediate (NEW in 5.80) and on-demand)
- Parameter information tooltips
- Line numbering margin and selection margin (NEW in 5.80)
- Code folding (NEW in 5.80)
- In-line Error Reporting (NEW in 5.80)
- New tabbed user interface with ability to work in both MDI and separate floating frame mode, can be moved behind main AmiBroker screen and brought back (Window->Toggle Frame) (NEW in 5.80) or kept on top (Window->Keep on top)
- Rectangular block copy/paste/delete (Use mouse and hold down left **Alt** key to mark rectangular block) (new in 5.80)
- Auto capitalisation (change case)
- Virtual space (new in 5.80)
- Enhanced printing (with syntax highlighting and header/footer)
- Code snippets (new in 5.80)
- Integrated Visual Debugger (NEW in 6.10)
- Bookmarks (NEW in 6.10)
- Find-in-Files (NEW in 6.10)
- Block comment/uncomment (NEW in 6.10)
- Clickable links in comments (NEW in 6.10)

These features greatly simplifies writing formula and provides instant help so time needed to write formula decreases significantly.

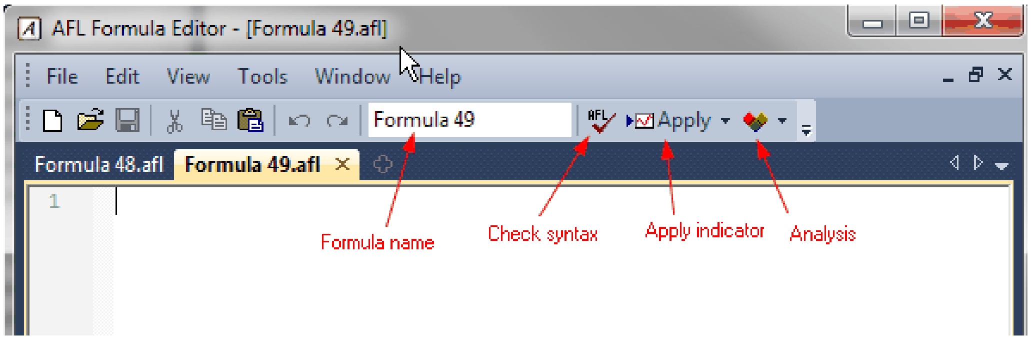


```
SetCustomBacktestProc("");

if (Status("action") == actionPortfolio)
{
    bo = GetBacktesterObject(); //  Get backtester object

    bo.PreProcess();    //  Do pre-processing

    for (i = 0; i < BarCount; i++)  //  Loop through all bars
    {
        for (sig = bo.GetFirstSignal( i ); sig; sig = bo.GetNextSignal(
        {   //  Loop through all signals at this bar

            if( sig.IsEntry() )
                bo.EnterTrade( i, sig.Symbol, sig.IsLong(), sig.Price,

            if( sig.IsExit() )
                bo.ExitTrade( i, sig.Symbol, sig.Price );

        }

        bo.HandleStops( i );    //  Handle programmed stops at this bar
```

Menu

Formula Editor menu options are described in detail in Menus: Formula Editor chapter of the guide.

Toolbar



The Formula Editor toolbar provides the following buttons:

- **New** - clears the formula editor window
- **Open** - opens the formula file
- **Save** - saves the formula under current name
- **Print** - prints the formula
- **Cut** - cuts the selection and copies to the clipboard
- **Copy** - copies the selection to the clipboard
- **Paste** - pastes current clipboard content in the current cursor position

- **Undo** - un-does recent action (multiple-level)
- **Redo** - re-does recent action (multiple-level)
- **Formula Name** - an EDIT field that allows to modify the formula file name, once you change the name here and press **Save** button the formula will be saved under new name and the change will be refleced in editor CAPTION BAR and in the STATUS BAR (Status bar shows full path).
- **Check syntax** - checks current formula for errors
- **Apply indicator** - saves the formula and applies current formula as a chart/indicator ONCE
- **Analysis -** saves the formula and selects it as current formula in Automatic Analysis window and repeat most recently used Analysis operation (i.e. Scan or Exploration or Backtest or Optimization)

Usage

Typical use of Formula Editor is as follows:

- open Formula Editor
- type the formula
- type meaningful name that describes the purpose of you code into **Formula Name** field
- click **Apply indicator** button (if you have written indicator code)
  .. or..
  click **Analysis** button to display Automatic Analysis window (when you have written exploration/scan or trading system)

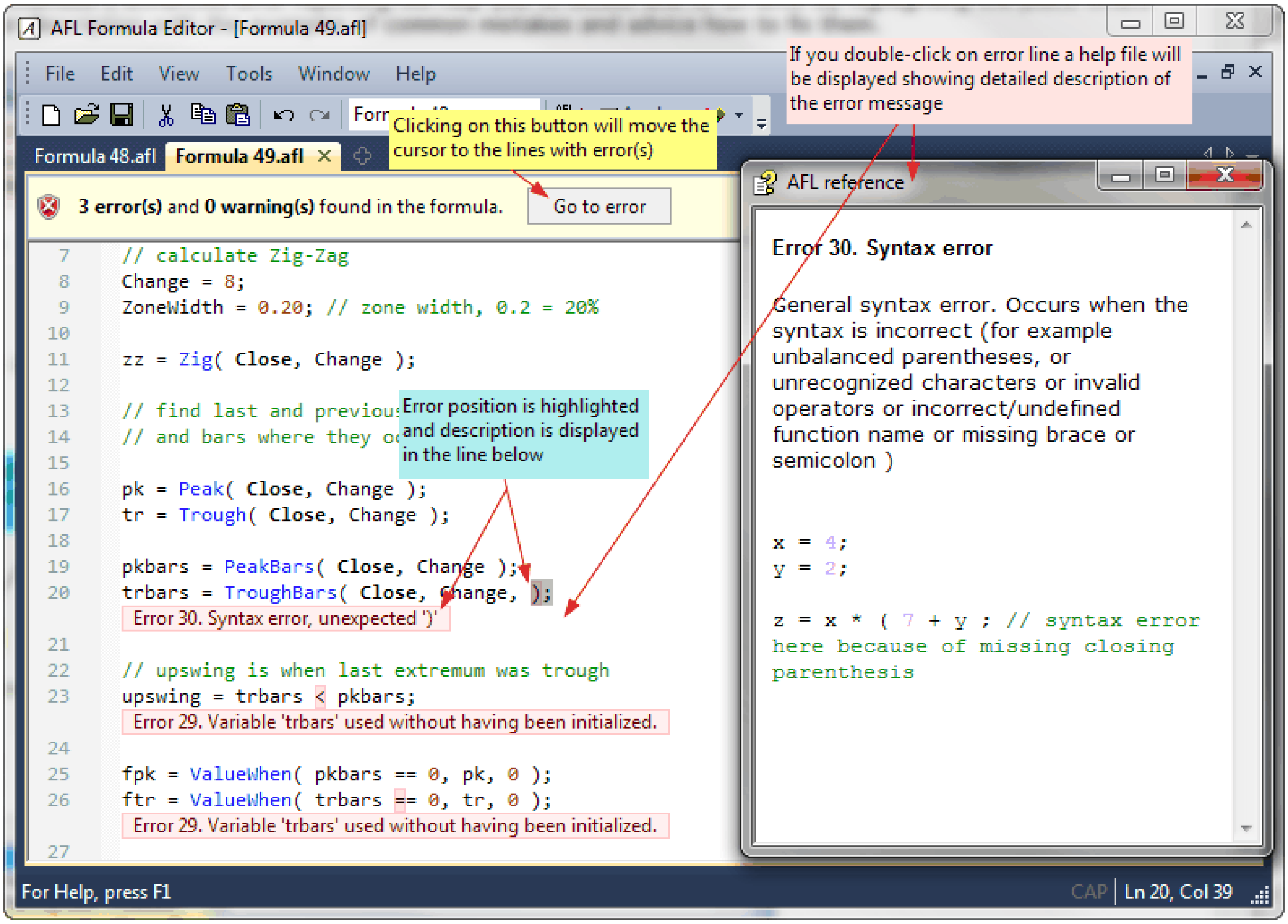
Syntax highlighting

AmiBroker's AFL editor features user-definable syntax highlighting that automatically applies user-defined colors and styles to different language elements like functions and reserved variable names, strings, numbers, comments, etc. This feature greatly simplifies code writing. You can modify coloring scheme in Preferences window.

Enhanced error reporting

When you make an error in your formula, AmiBroker's enhanced error reporting will help you to locate and fix an error by highlighting the place where error occured and displaying extended error description with the examples of common mistakes and advice how to fix them. In version 5.80 description of errors are displayed in-line with the code.

A message bar displays total number of errors and/or warnings. If you press "**Go to error**" button the editor will move the caret to the relevant line with the error, if you press it again, it will move to the next error and so on. If you close the message bar with the "X" button all error messages will be cleared (hidden) from the view. You can use **Edit->Clear Error Message** menu (**Ctrl+E**) to clear individual error message (in the current line).



If you double-click on error line a help file will be displayed showing detailed description of the error message

Clicking on this button will move the cursor to the lines with error(s)

3 error(s) and 0 warning(s) found in the formula.

Error position is highlighted and description is displayed in the line below

```
// calculate Zig-Zag
Change = 8;
ZoneWidth = 0.20; // zone width, 0.2 = 20%

zz = Zig( Close, Change );

// find last and previous
// and bars where they oc

pk = Peak( Close, Change );
tr = Trough( Close, Change );

pkbars = PeakBars( Close, Change );
trbars = TroughBars( Close, Change, );
Error 30. Syntax error, unexpected ')'

// upswing is when last extremum was trough
upswing = trbars < pkbars;
Error 29. Variable 'trbars' used without having been initialized.

fpk = ValueWhen( pkbars == 0, pk, 0 );
ftr = ValueWhen( trbars == 0, tr, 0 );
Error 29. Variable 'trbars' used without having been initialized.
```

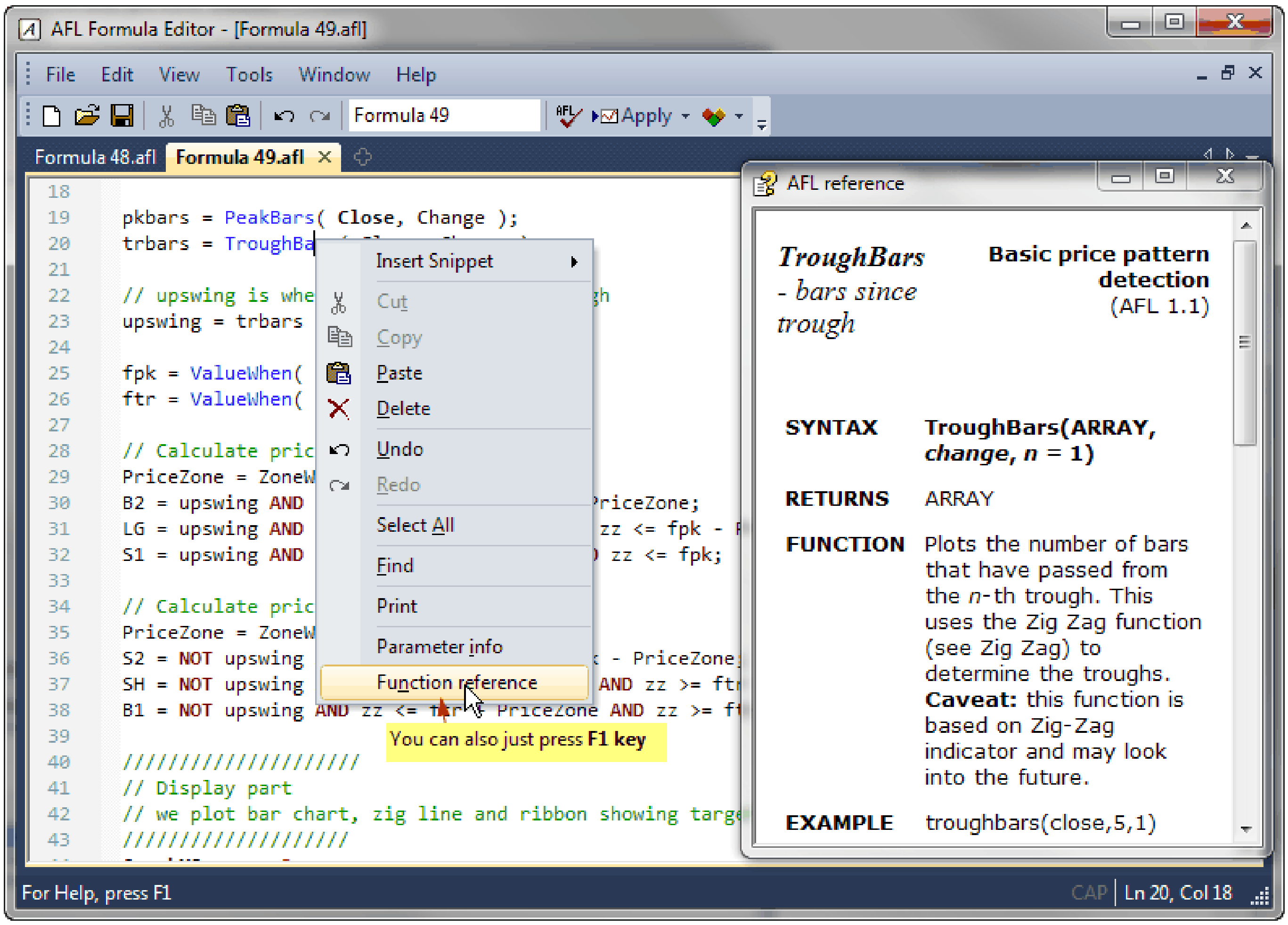
Error 30. Syntax error

General syntax error. Occurs when the syntax is incorrect (for example unbalanced parentheses, or unrecognized characters or invalid operators or incorrect/undefined function name or missing brace or semicolon)

```
x = 4;
y = 2;

z = x * ( 7 + y ; // syntax error here because of missing closing parenthesis
```

Context help

You can quickly display relevant AFL function reference page if you press **F1** key or choose "Function reference" from the context menu while the caret is inside or right after function name as shown in the picture below:



```
pkbars = PeakBars( Close, Change );
trbars = TroughBa
// upswing is whe                                  gh
upswing = trbars
fpk = ValueWhen(
ftr = ValueWhen(
// Calculate pric
PriceZone = ZoneW
B2 = upswing AND                               riceZone;
LG = upswing AND                               zz <= fpk - F
S1 = upswing AND                               zz <= fpk;
// Calculate pric
PriceZone = ZoneW
S2 = NOT upswing                               - PriceZone;
SH = NOT upswing                               AND zz >= ftr
B1 = NOT upswing AND zz <= ftr   PriceZone AND zz >= ft
/////////////////////
// Display part
// we plot bar chart, zig line and ribbon showing targe
////////////////////
```

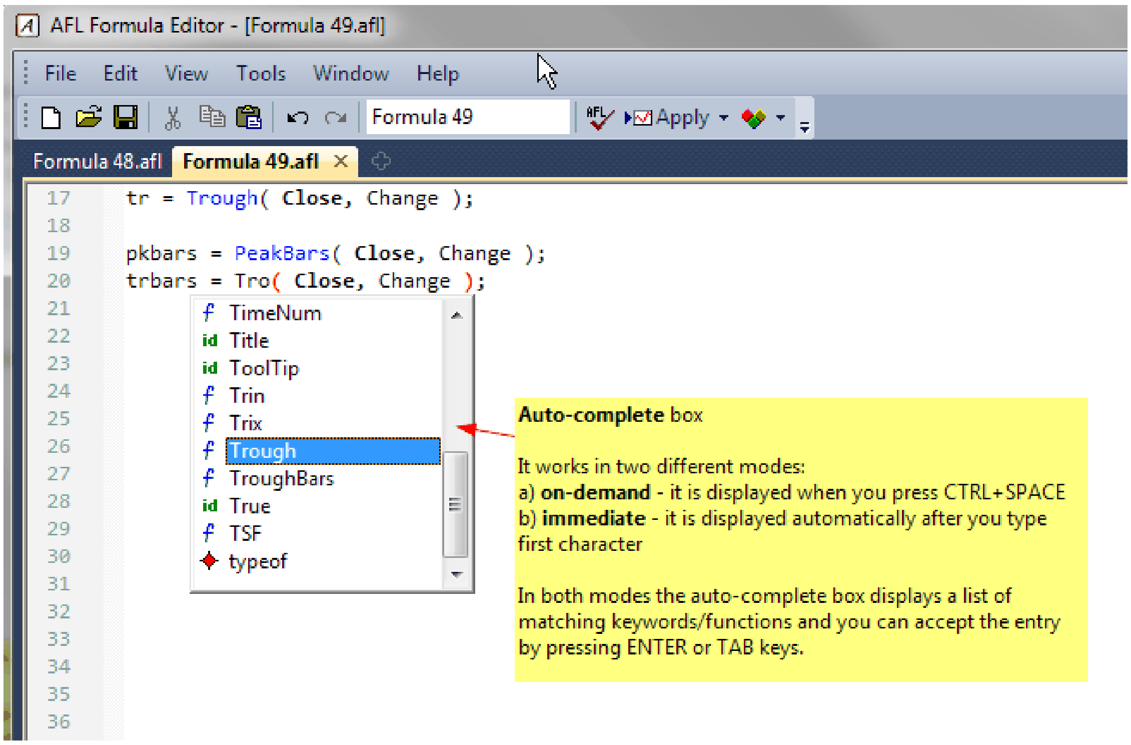
***TroughBars* - bars since trough**

Basic price pattern detection (AFL 1.1)

SYNTAX	**TroughBars(ARRAY, *change*, *n* = 1)**
RETURNS	ARRAY
FUNCTION	Plots the number of bars that have passed from the *n*-th trough. This uses the Zig Zag function (see Zig Zag) to determine the troughs. **Caveat:** this function is based on Zig-Zag indicator and may look into the future.
EXAMPLE	troughbars(close,5,1)

Automatic statement completion

The automatic completion feature (available when you press **CTRL+SPACE** key combination) finishes typing your functions and reserved variables for you, or displays a list of candidates if what you've typed has more than one possible match. You can select the item from the list using up/down arrow keys or your mouse. To accept selection press RETURN (ENTER). You can also type immediately space (for variables) or opening brace (for function) and AmiBroker will auto-complete currently selected word and close the list. To dismiss the list press ESC key.



```
tr = Trough( Close, Change );

pkbars = PeakBars( Close, Change );
trbars = Tro( Close, Change );
```

Auto-complete box

It works in two different modes:
a) **on-demand** - it is displayed when you press CTRL+SPACE
b) **immediate** - it is displayed automatically after you type first character

In both modes the auto-complete box displays a list of matching keywords/functions and you can accept the entry by pressing ENTER or TAB keys.

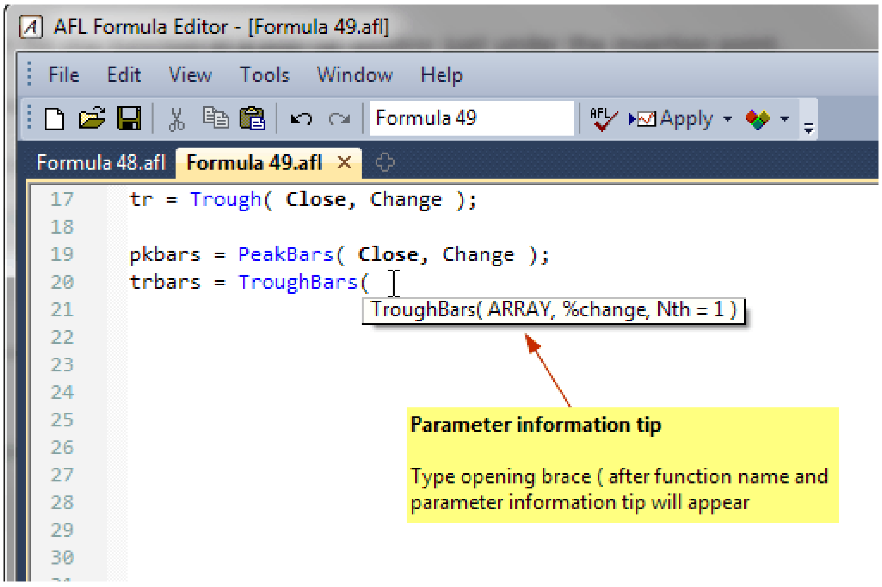
Parameter Information

When you are typing a function, you can display a Tool Tip containing the complete function prototype, including parameters. The **Parameter Info** Tool Tip is also displayed for nested functions.
With your insertion point next to a function, type an open parenthesis as you normally would to enclose the parameter list.

AmiBroker displays the complete declaration for the function in a pop-up window just under the insertion point.

Typing the closing parenthesis dismisses the parameter list.

You can also dismiss the list if you press arrow up/down key, click with the mouse or press RETURN.



AFL Formula Editor - [Formula 49.afl]

```
tr = Trough( Close, Change );

pkbars = PeakBars( Close, Change );
trbars = TroughBars(
```

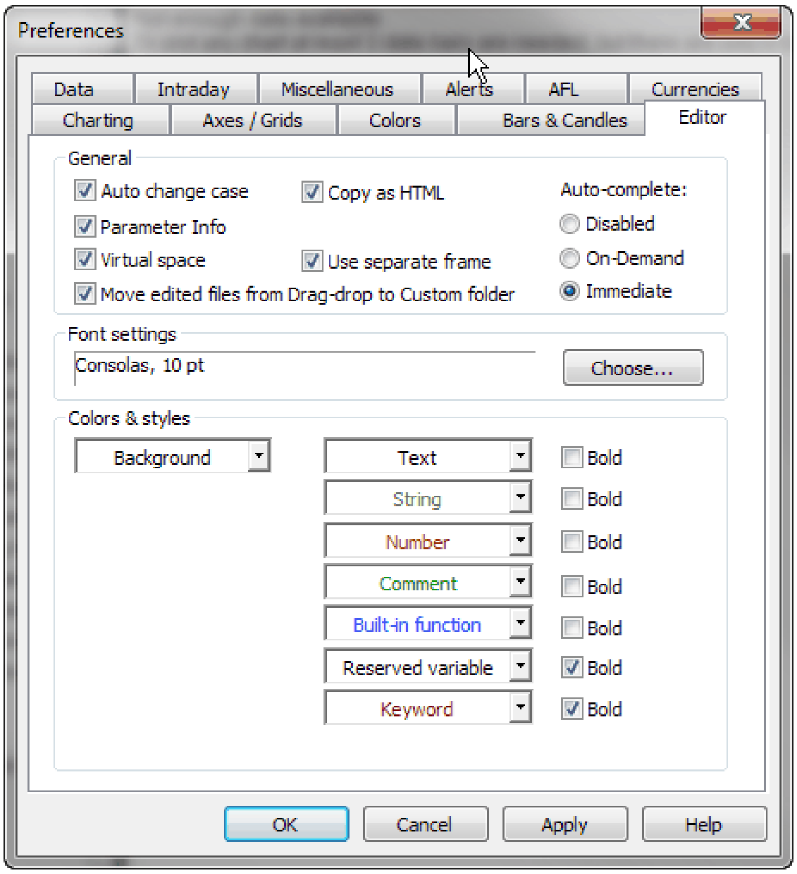
TroughBars(ARRAY, %change, Nth = 1)

Parameter information tip

Type opening brace (after function name and parameter information tip will appear

Editor configuration

The settings of the AFL editor can be changed using **Tools->Preferences**, **Editor** page:



Preferences

General
- [✓] Auto change case
- [✓] Copy as HTML
- [✓] Parameter Info
- [✓] Virtual space
- [✓] Use separate frame
- [✓] Move edited files from Drag-drop to Custom folder

Auto-complete: Disabled / On-Demand / [✓ Immediate]

Font settings: Consolas, 10 pt

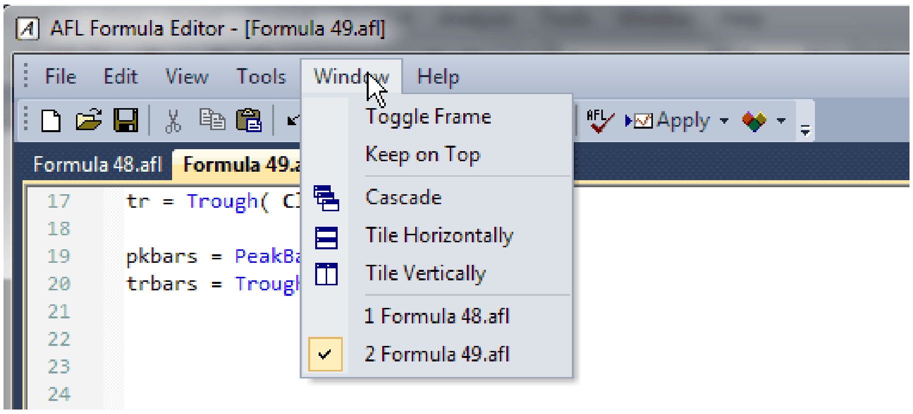
Colors & styles

Background	Text	Bold
	String	Bold
	Number	Bold
	Comment	Bold
	Built-in function	Bold
	Reserved variable	[✓] Bold
	Keyword	[✓] Bold

- Auto change case - controls whenever editor automatically changes case of reserved keywords (for example if user typed valuewhen it would change it to ValueWhen)
- Parameter info - controls whenever parameter info tips are displayed

- Virtual space - controls whenever it is possible to place the caret freely in any place after the end of a line
- Move edited files from drag-drop to custom folder - normally formulas created by drag-drop mechanism are located in hidden drag-drop folder, if you then want to edit them, you can do so in place so they remain in drag drop (hidden) folder, or you may choose to move them automatically to 'custom' folder. This switch enables automatic move to custom folder
- Copy as HTML - enables copies in HTML format so AFL code is copied with colors, without it it will be copied as plain text without formatting
- Use separate frame - if turned on it displays AFL Editor in completely separate frame that behaves like separate application, if it is turned off, then AFL editor is displayed as a MDI tab within main AmiBroker frame (along with charts, analysis windows, web, account windows and so on). By default it is turned on
- Auto-complete: in "On-demand" mode auto-complete list shows up only when you press **Ctrl+SPACE**, in "Immediate" mode auto-complete list pops up automatically as soon as you type first character (letter) of the identifier.

Window control

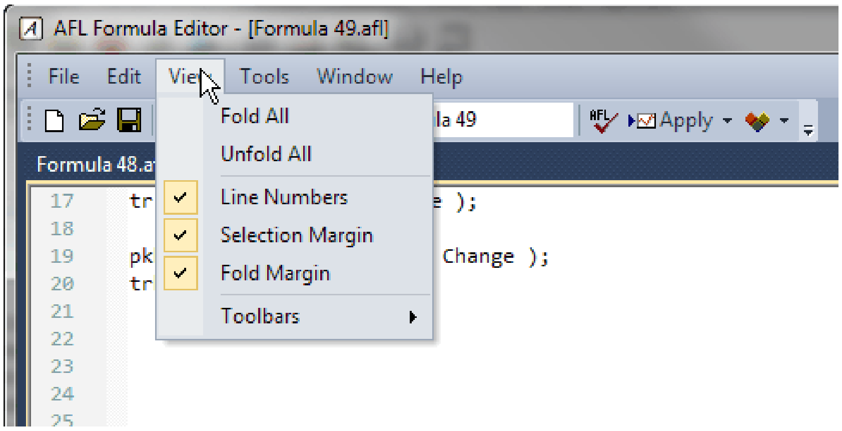


```
tr = Trough( C

pkbars = PeakB
trbars = Troug
```

AFL Editor Window as a separate frame can be brought on top or to the back as any other application window using Windows Task Bar. In addition to that there is a **Window->Toggle Frame** menu (and **Ctrl+`** shortcut, ` is the tilde key just above TAB key on most keyboards) that allows to quickly toggle between AmiBroker main frame and AFL editor frame.

The user may also turn on **Window->Keep On Top** feature that keeps editor window on top of AmiBroker main frame.

Margins

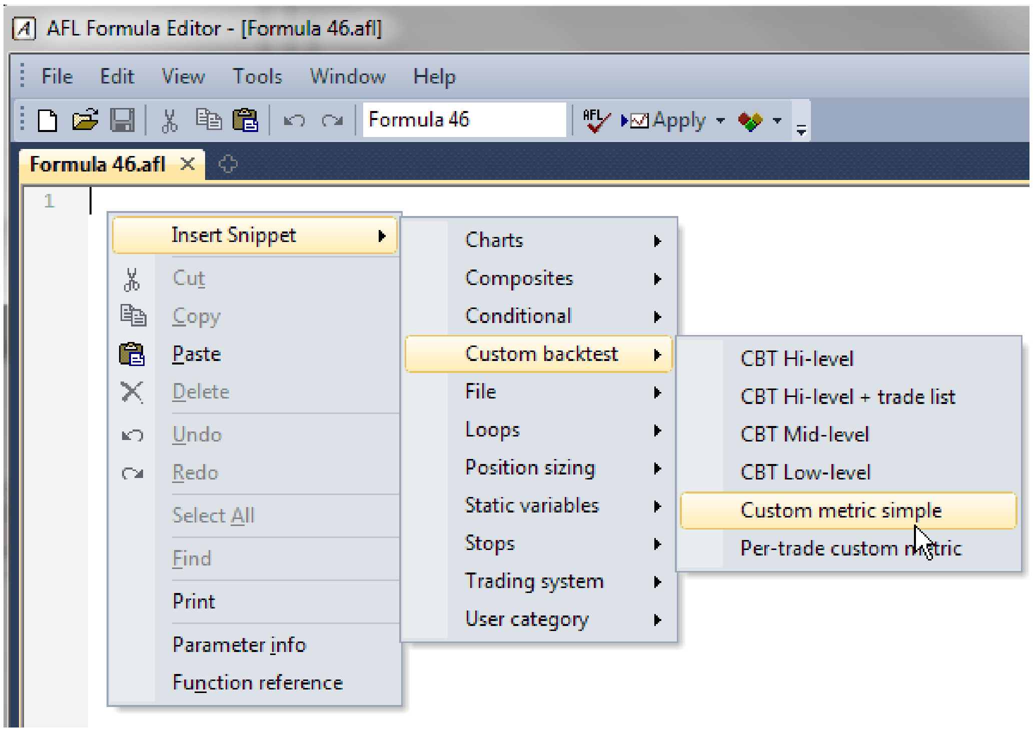


Line numbers margin, **Selection margin** and **Fold margin** can be switched on/off using **View** menu. In this menu there are also options to fold/unfold all code.

Code snippets

Code snippet is a small piece of AFL code. It can be inserted by:

- right-clicking in the AFL editor window and choosing "Insert Snippet" menu, or
- dragging a snippet from Code Snippet window, or
- typing keyboard trigger (such as @for) in the editor



For more information about Code snippets see Tutorial: Using Code-snippets

Bookmarks

A bookmark is a marker that allows to quickly jump to marked line within the formula.

To set/remove the bookmark use **Edit->Bookmark->Insert /Remove** menu or **Ctrl+F2** key. When you set the bookmark a light blue marker will be shown in the left-hand margin. Now with a couple of bookmarks set you can quickly navigate (jump) betwen them using **F2** key - goes to next bookmark, **Shift+F2 key** - goes to previous bookmark. You can clear all bookmarks using **Edit->Bookmark->Clear All.** Please note that bookmarks are saved along with debug information (breakpoints/watches) in the separate file with .dbg extension.

Find in Files

Find in Files functionality is available from **Edit->Find in Files** menu and allows you to search all files in selected folders for specified text. The search results are presented in the Output window (**Window->Output** menu) like this:

```
Filepath\Filename.afl(line_number): Line contents
```

Now if you double click on that output line AmiBroker would automatically open given file in the new editor tab.

Block comment/uncomment

The AFL editor now provides a tool to automatically comment out blocks of selected lines. The tool adds a // (double slash) comment to every line within selected area. To use this tool, select some lines using mouse or cursor + SHIFT key, then use **Edit->Line Comment.** All lines within selected range will be commented out with // (double slash). To uncommment, select the range again and choose **Edit->Line Comment** once again.

Clickable links in comments

The AFL Editor now supports clickable JavaDoc/Doxygen-style links in doc comments. To use links in comments you have to put @link command followed by path to file or URL inside JavaDoc/Doxygen style comment:

- multiline comment that begins with /** like this:

  ```
  /**
  @link readme.html
  @link http://www.amibroker.com
  */
  ```

  (note double asterisk after initial slash)
- single line comment that begins with triple slash

  ```
  /// @link readme.html
  /// @link c:\program files\amibroker\readme.html
  /// @link http://www.amibroker.com
  ```

Now when you hover the mouse over @link command you will see the underline that indicates it is clickable. It reacts to double click (not single click). When you double click it linked document will be open.

@link command can open web pages, local files (both relative and absolute paths are supported) with Windows-registered program to open given file type. So if you use:

```
/// @link something.doc
```

then MS word would be used.

```
/// @link test.xls
```

would open test.xls in Excel.

Relative paths refer to AmiBroker working directory. Html files are open with default browser, txt files are usually open with Notepad (or whatever application you use). If file does not exist then you will get an error message.

Code Snippets window

Code snippets are small re-usable pieces of AFL code, detailed information on usage of Code Snippets can be found in this tutorial.

Code Snippets window is available in new AFL editor. It can be shown/hidden using Window menu.

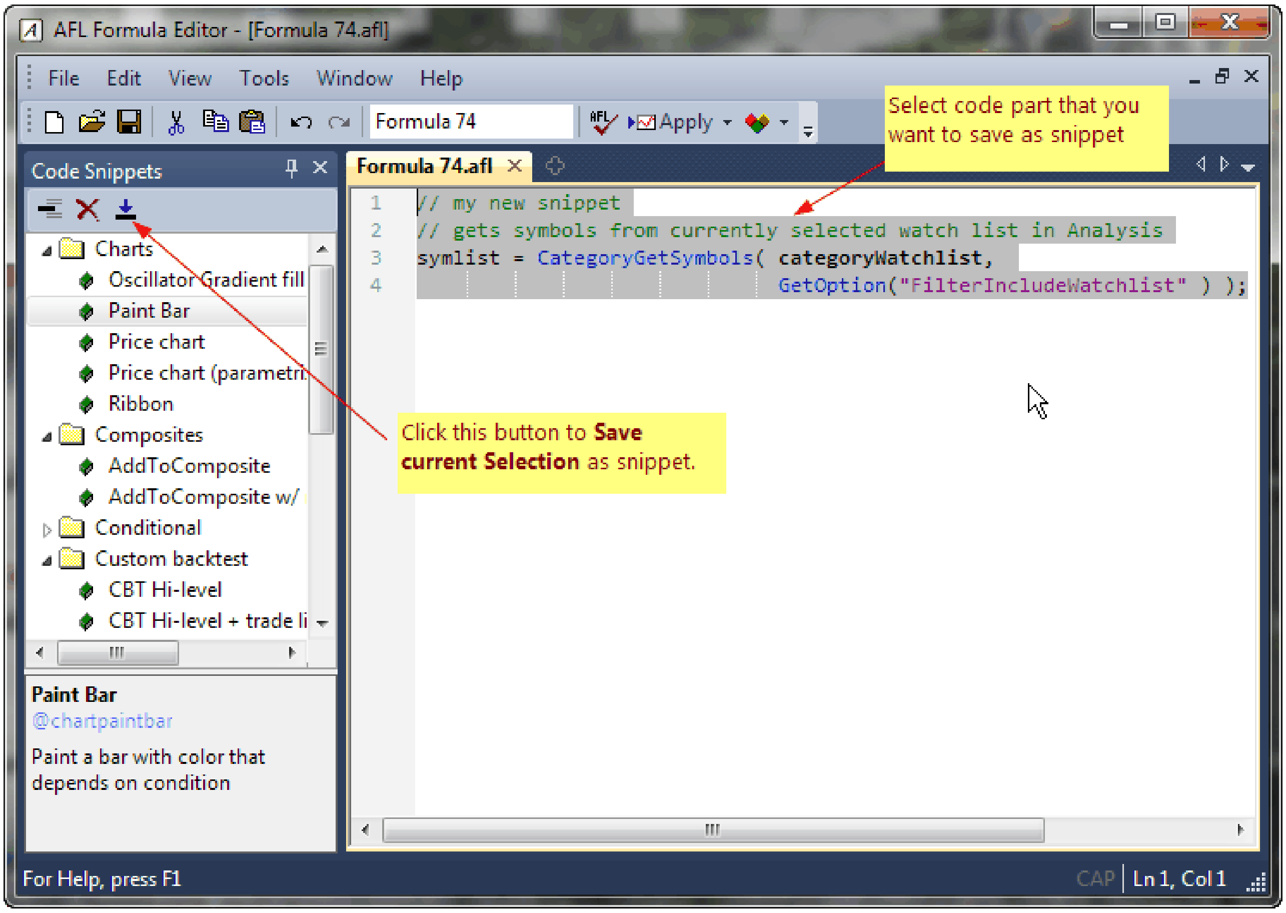
INSERTING SNIPPET

To insert an existing snippet, drag-drop the snippet from the Code Snippet list into AFL Editor, or double click on the snippet.

CREATING YOUR OWN SNIPPET

To create your own snippet, do the following:

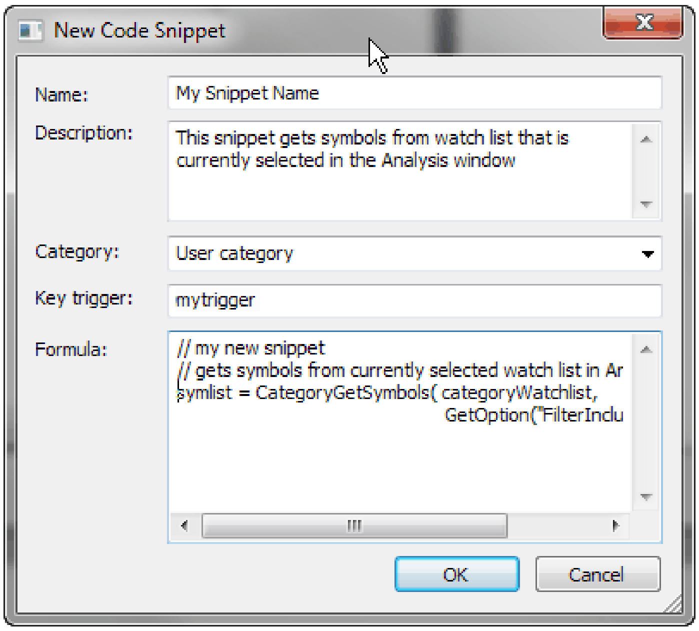
1. type the code you want
2. select (mark) the code you want to place in a snippet
3. press **Save selection** as snippet button in the Code Snippets window



```
// my new snippet
// gets symbols from currently selected watch list in Analysis
symlist = CategoryGetSymbols( categoryWatchlist,
                              GetOption("FilterIncludeWatchlist" ) );
```

Paint Bar
@chartpaintbar
Paint a bar with color that depends on condition

If you do the steps above the following dialog will appear:



Name: My Snippet Name

Description: This snippet gets symbols from watch list that is currently selected in the Analysis window

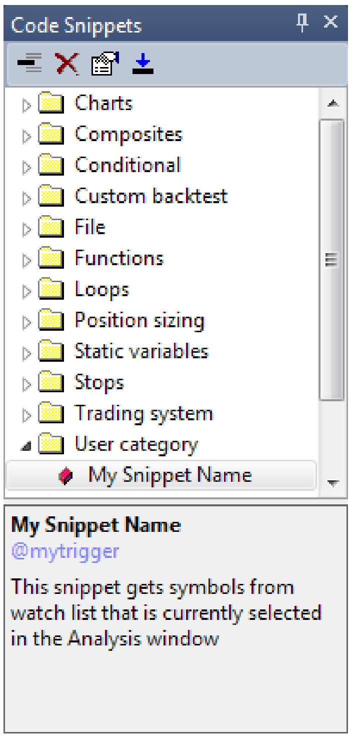
Category: User category

Key trigger: mytrigger

Formula:

```
// my new snippet
// gets symbols from currently selected watch list in Ar
symlist = CategoryGetSymbols( categoryWatchlist,
                              GetOption("FilterIndu
```

Now you need to enter the **Name** of the snippet, the **Description** and **Category**. **Category** can be selected from already existing items (using drop down box), or new category name can be typed in the category field. **Key trigger** field is optional and contains snippet auto-complete trigger (described above). The **Formula** field is the snippet code itself. Once you enter all fields and press **OK**, your new snippet will appear in the list.



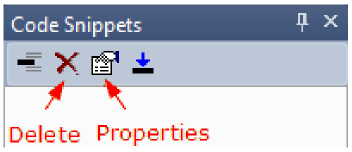
My Snippet Name
@mytrigger
This snippet gets symbols from watch list that is currently selected in the Analysis window

From then on you can use your own snippet the same way as existing snippets. Perhaps most convenient method is using drag-drop from the list to AFL editor.

As you may have noticed user-defined snippets are marked with red color box in the Code Snippets list. Only user-defined snippets can be overwritten and/or deleted.

EDITING SNIPPET

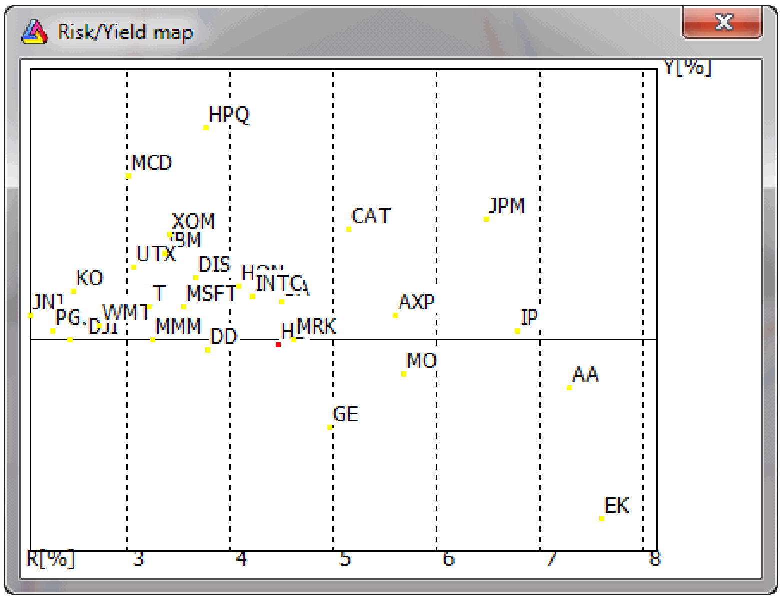
To edit existing user-defined snippet, you can either follow the steps above and give existing name. AmiBroker will ask then if you want to overwrite existing snippet, or you can simply click on **Properties** button and edit the snippet directly, without re-inserting it.



DELETING SNIPPET

To delete a snippet, select the snippet you want to delete from the list and press **Delete** (X) button in the Code Snippet window.

Risk-Yield Map window



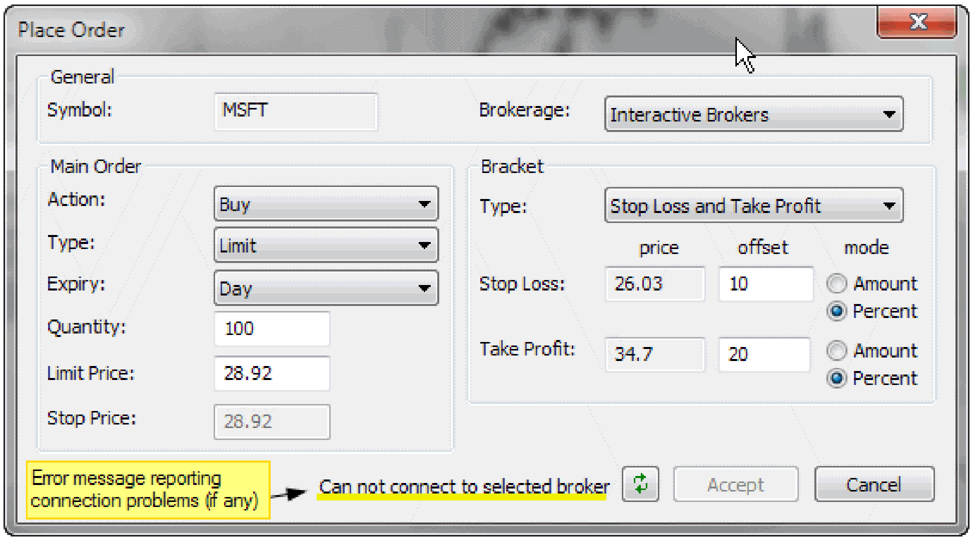
This map provides fast information about risk and possible yields. Yield is a the average weekly percentage return while Risk is a standard deviation of percentage weekly returns. On the X axis risk is presented and on Y axis - yield. Thus in the upper part of the map we have got symbols with giving best yield, with risk increasing from left to right side of the map.

Selected symbol is marked with a different color, and you can zoom the part of the map by pressing left mouse button and marking rectangle to zoom in.

Place Order dialog

NOTE: This functionality requires automated trading interface add-on that can be downloaded separately

To place order from the chart please first choose **Insert->Buy Order** or **Insert->Sell Order** menu or appropriate buttons from Order toolbar, then AmiBroker will allow you to draw a horizontal line with mouse cursor over the chart. Simply click with LEFT mouse button over chart and hold it down - you a horizontal line will show marking the price level, once you move the line to correct level, release left mouse button to place the order (the following dialog will show up), or press ESC key to cancel entire operation.


Place Order
General
Symbol: MSFT
Brokerage: Interactive Brokers
Main Order
Action: Buy
Type: Limit
Expiry: Day
Quantity: 100
Limit Price: 28.92
Stop Price: 28.92
Bracket
Type: Stop Loss and Take Profit
price offset mode
Stop Loss: 26.03 10 Amount Percent
Take Profit: 34.7 20 Amount Percent
Error message reporting connection problems (if any)
Can not connect to selected broker
Accept
Cancel

In the "**Brokerage**" currently selected trading interface is displaed. After installing Interactive Brokers automated trading interface (from http://www.amibroker.com/at/) the text "Interactive Brokers" should appear. If there is no trading interface installed the combo box will be empty. If you installed other trading interfaces they should appear in the list.

In the "**Action**" field you can choose either **Buy** or **Sell** - note that preselected is the option choosen earlier from the menu or the toolbar.

In the "**Type**" field you can choose order type (Market, Limit, Stop, StopLimit, etc), by default "Limit" order is selected.

In the "**Expiry**" field you can choose how long given order will be valid. Currently available are Day and GoodTilCanceled.

In the "**Quantity**" field you can enter the number of shares / contracts to buy/sell

In the "**Limit Price**" field you can enter the limit price for the order - AmiBroker will fill the value selected on chart by default.

In the "**Stop Price**" field you can enter stop price for Stop and Stop Limit orders.

In the "**Bracket**" group you can choose additional automatic bracket orders. Bracket orders are "child" stop loss and/or take profit orders that are connected to main "Parent" order and work as OCA (one cancel another) group (so when for example take profit is triggered, the corresponding stop loss is canceled). Bracket prices are calculated automatically from Limit price. The distance between limit price and stop loss / take profit levels is defined by appropriate "offset" fields. The distance can be expressed in amount (dollars) or percent of limit price.

All prices are subject to rounding depending on current symbol TickSize setting (see Information window). If TickSize is not defined (i.e. is equal to zero), then AmiBroker assumes 0.01 (one cent).

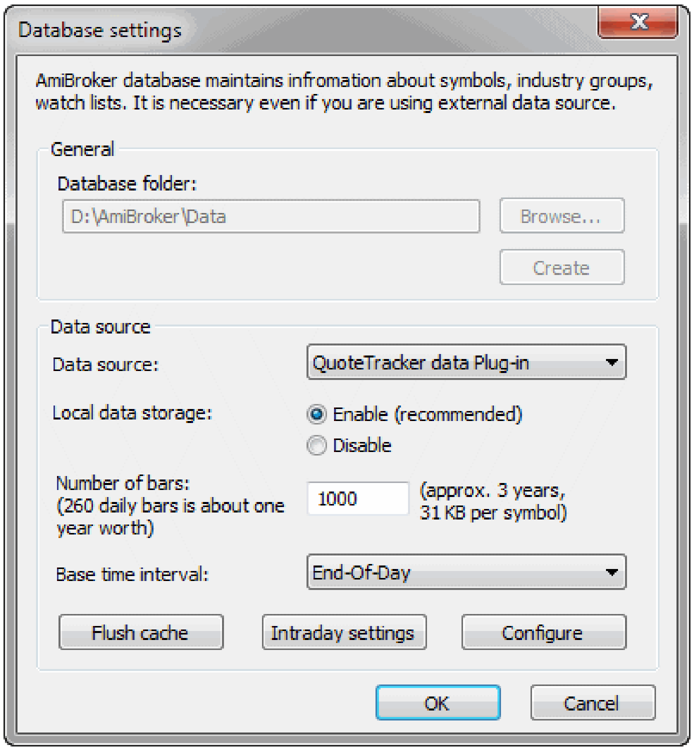
Status field (highlighted in yellow) - shows the connection status between AmiBroker and trading interface. Any connection error will be displayed here and in case of an error AmiBroker will disable "**Accept**" button will attempt to reconnect every 5 seconds. You can also manually trigger reconnection attempt by pressing button with two green arrows.

When status field shows "Connected" then **Accept** button is enabled and you can press it to place order. Note that currently the interface places orders with Transmit flag set to FALSE. This means that orders are NOT actually transmitted to exchange but await manual transmit in the TWS. This is safety measure.

Once dialog is closed by pressing Accept, the horizontal line showing the limit price entered will stay on chart. You can not move it by default, but you can delete it by selecting it and pressing "DEL" (Delete) key.

Database Settings

This window allows you to define per-database settings. It is accessible via File->Database Settings menu.


Database settings

AmiBroker database maintains infromation about symbols, industry groups, watch lists. It is necessary even if you are using external data source.

General

Database folder: D:\AmiBroker\Data

Data source

Data source: QuoteTracker data Plug-in

Local data storage: [✓ Enable (recommended)] Disable

Number of bars: (260 daily bars is about one year worth) 1000 (approx. 3 years, 31 KB per symbol)

Base time interval: End-Of-Day

IMPORTANT: These per-database settings in this window take precedence over default values definable in Preferences window. See explanation in Tutorial: Understanding database concepts.

The database settings window is divided into two parts: *General* and *Data source*

General settings part are enabled **only at the database creation time** (**File->New database**), once database is created these controls become disabled.

- **Browse... -** allows to browse for folder where new database should be created.
- **Create** - clicking on this button creates the database inside the folder specified in **Database folder** edit field.

For more details about creating new database working with particular data source please check Tutorial section.

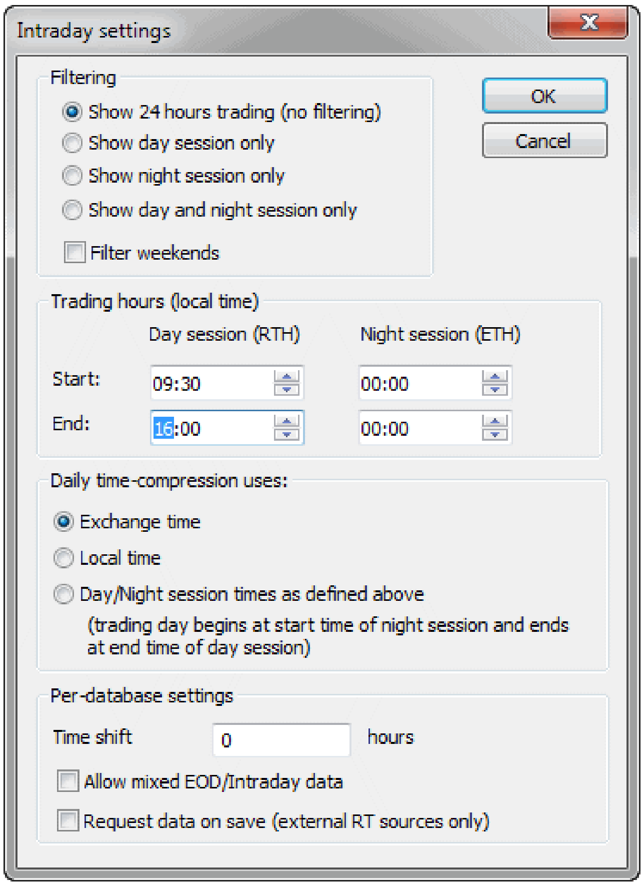
Data source part becomes enabled once database is created and it can be used to modify settings for already existing databases (via **File->Database Settings** menu). The following controls are available:

- **Data source:** defines data souce, this can be either
  - ♦ (local) - it means that no external source is used and data are maintained by AmiBroker itself. Such database can be updated either using AmiQuote (Tools->Auto-update quotes) or using ASCII import - Import Wizard, Metastock importer, or script.
  - ♦ external data source (one of: eSignal, myTrack, QuoteTracker, Quotes Plus, TC2000/TCNet, FastTrack, Metastock) - it means that data are retrieved directly from external database / data source. Such database is updated automatically via plugin and does not require any user

action in AmiBroker. For example if you use TC2000 as a data source all data that are present in TC2000 system become automatically available in AmiBroker. For more details please read Tutorial: Understanding database concepts.

- **Local data storage:** decides if data from **external data source** should be stored/cached also in AmiBroker's own files. If "Enabled" then external data are cached in local files. If "Disabled" then local files do not store external data. Switching this to "Enabled" is **required** for most **real-time** data sources as eSignal, myTrack, QuoteTracker. This setting has no effect if data source is set to (local).
- **Number of bars to load -** defines how many bars should be loaded from **external data source** and kept in AmiBroker. Examples: 10-years EOD: 2600, 60-days intraday 1-minute: 30000 (approx). This setting has no effect if data source is set to (local).
- **Base time interval** - defines what 'base' bar interval is used in this database. For real-time data sources this should be set once at the database creation time. This is so because real-time sources need to collect RT ticks and pack them (time-compress) into interval bars. This setting defines the minimum 'grain'. For EOD sources it is (End-of-day (daily). For real-time sources this should be 1-minute or higher. For some real-time sources (like eSignal) this can be also set to tick, 5-sec or 15-sec.
  Please note also that you **won't be able to use intraday charting and/or analysis** until base time interval is set to something below end-of-day interval (it can be 1-minute for example). For more details please read Tutorial: Basic charting guide.
- **Flush cache -** allows to force cache flushing and force retrieving fresh data from the plugin
- **Configure -** allows to display data source specific configuration dialog see Tutorial section for details on configuring various data sources.
- **Intraday settings** - allows to define per-database settings for intraday databases (see below)

Intraday Settings window



Filtering
- [✓ Show 24 hours trading (no filtering)]
- Show day session only
- Show night session only
- Show day and night session only
- Filter weekends

Trading hours (local time)

	Day session (RTH)	Night session (ETH)
Start:	09:30	00:00
End:	16:00	00:00

Daily time-compression uses:
- [✓ Exchange time]
- Local time
- Day/Night session times as defined above (trading day begins at start time of night session and ends at end time of day session)

Per-database settings

Time shift 0 hours
- Allow mixed EOD/Intraday data
- Request data on save (external RT sources only)

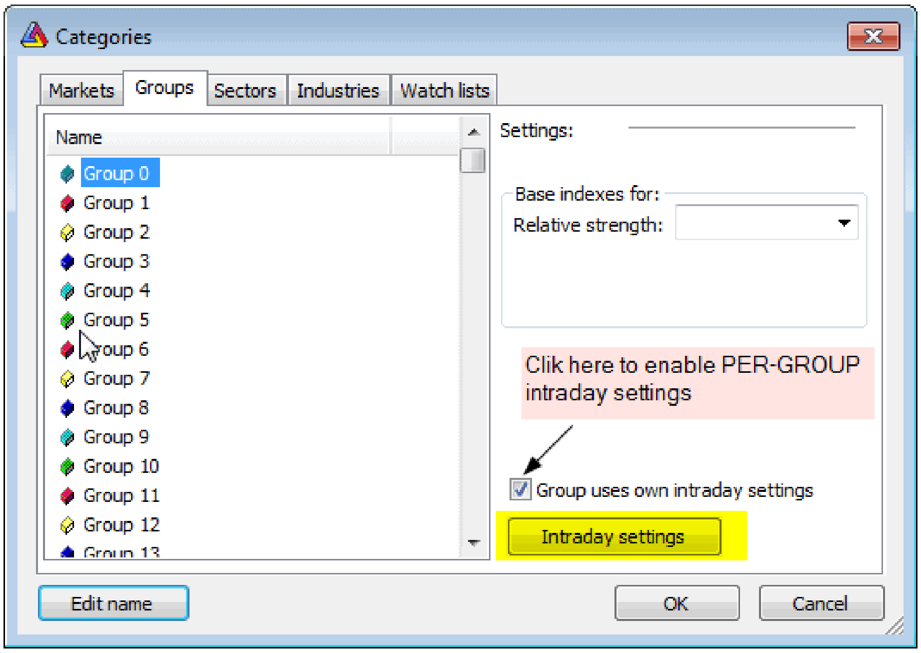
- **Filtering -** this provides control over the display of intraday data. AmiBroker collects all the data but displays only those data which are inside selected trading hours start-end time. Please note that this affects all charts and windows except Quote Editor that always displays all available data.

  Show 24 hours trading (no filtering) - all data are displayed (no filtering at all)
  Show day session only - only the data between day session (RTH) start and end times are displayed
  Show night session only - only the data between night sesison (ETH) start and times are displayed
  Show day and night session only - only the data between either day session start/end time or night session start/end time are displayed

  Filter weekends - when checked AmiBroker collects but does not display data from weekends. When unchecked those data are collected and displayed.
- **Trading hours Start / End** - defines trading hours start and end times for day (RTH) and night (ETH) sessions separately (see above). Please note that the times should be specified in your local time zone.
- **Daily time-compression uses -** this decides how AmiBroker performs intraday to daily time compression

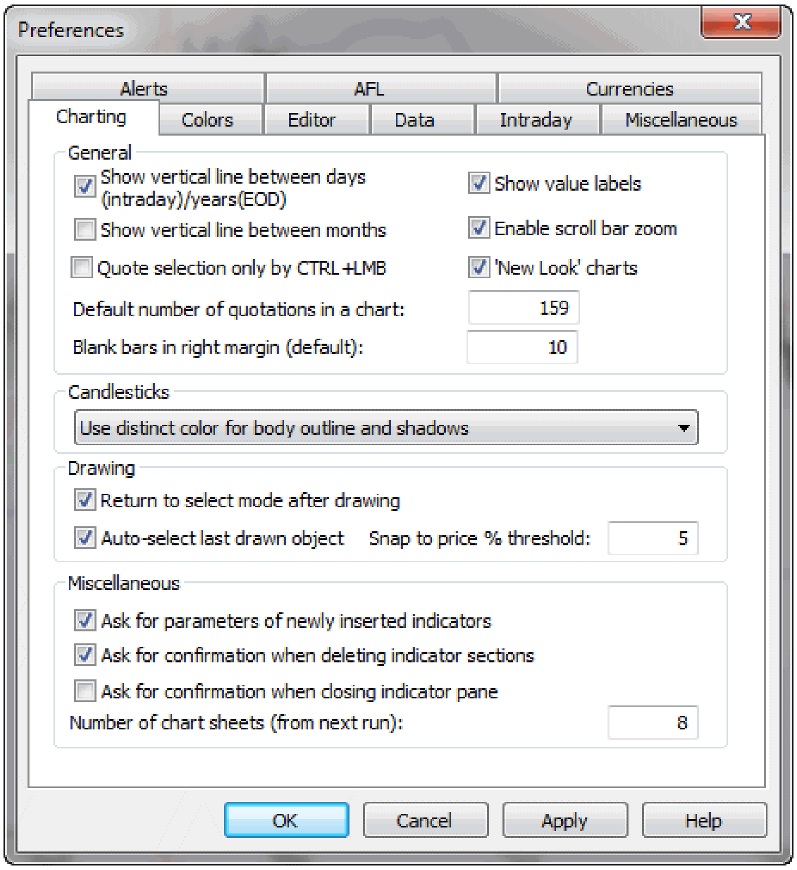
  Exchange time - daily data are constructed from intraday bars starting from 00:00 and ending at 23:59 in the EXCHANGE (or data source) TIME ZONE
  Local time - daily data are constructed from intraday bars starting from 00:00 and ending at 23:59 in the LOCAL (computer) TIME ZONE
  Day/Night session times as defined above - daily data are constructed from the intraday bars that start at the start time of night session (previous day) and end at the end time of day session)
- **Time shift -** is the time difference (in hours) between your local time zone and the exchange time zone
- **Allow mixed EOD/Intraday data** - it allows to work with database that has a mixture of intraday and EOD data in one data file. If this is turned on then in intraday modes EOD bars are removed on-the-fly and in daily mode EOD bars are displayed instead of time compressed intraday or if there is no EOD bar for corresponding day then intraday bars are compressed as usual.
  This mode works in conjunction with new versions of plugins that allow mixed data. As of June 2008 Mixed mode is now supported by IQFeed plugin, eSignal plugin (1.7.0 or higher) plugins only. Mixed mode allows intraday plus very long daily histories in one database.

Note that **Intraday Settings** available from **Database Settings** dialog are **PER-DATABASE**. There is however also an option to define **PER-GROUP** intraday settings. To use PER-GROUP intraday settings you have to open Symbol->Categories window, switch to **Groups** tab and
check "Group uses own intraday settings" box as shown in the picture below



Then you can click on **Intraday Settings** button to display per-group settings. Please note that each group in the category list can have its own individual settings so you can easily setup groups so they contain instruments traded in different hours. You can move symbols between groups using Symbol->Organize assignments dialog.

Preferences window



General
- [✓] Show vertical line between days (intraday)/years(EOD)
- [ ] Show vertical line between months
- [ ] Quote selection only by CTRL+LMB
- [✓] Show value labels
- [✓] Enable scroll bar zoom
- [✓] 'New Look' charts

Default number of quotations in a chart: 159
Blank bars in right margin (default): 10

Candlesticks
Use distinct color for body outline and shadows

Drawing
- [✓] Return to select mode after drawing
- [✓] Auto-select last drawn object

Snap to price % threshold: 5

Miscellaneous
- [✓] Ask for parameters of newly inserted indicators
- [✓] Ask for confirmation when deleting indicator sections
- [ ] Ask for confirmation when closing indicator pane

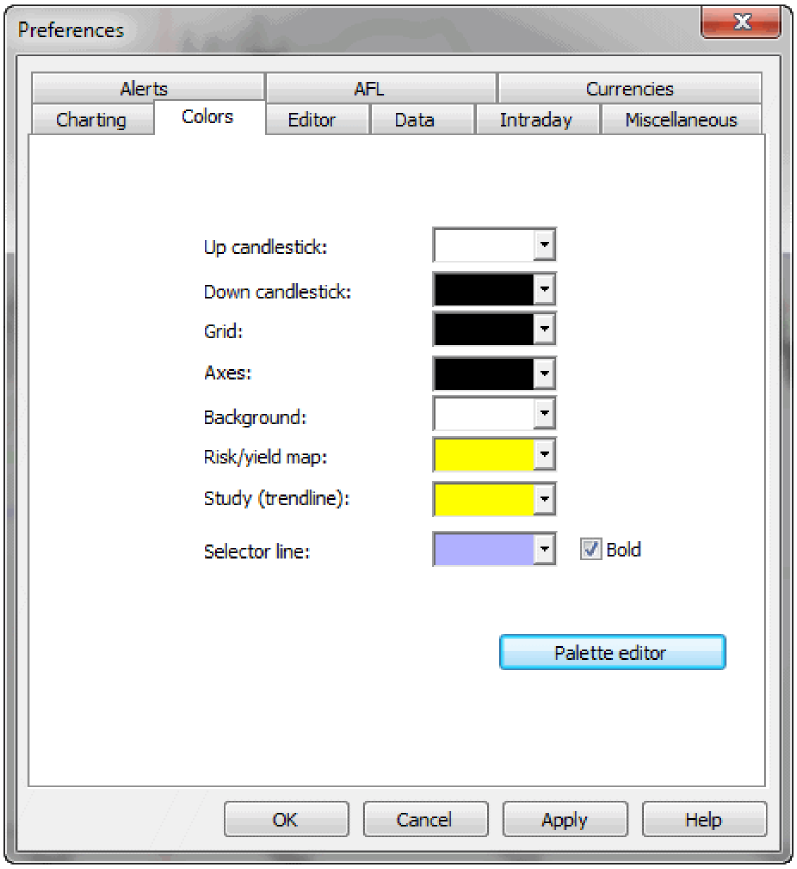
Number of chart sheets (from next run): 8

Charting tab - allows you to modify charting options

- **Default number of quotations in a chart -** this sets the amount of bars initially displayed in the chart. (in other words it defines "normal" zoom range)
- **Blank bars in right margin** - defines how many blank bars are added in the right margin (past the last available quote). This blank margin allows you to project studies (trend lines for example) into the future
- **Quote selection only by CTRL+LMB** - this decides how vertical selection line is invoked. When this box is unchecked - single click on the chart causes display of the selection line, when this box is checked you have to hold down CTRL key while clicking to get the selection line
- **Show vertical line between days (intraday)/years(EOD)** - this decides if dotted vertical line is displayed on the chart to mark day (in intraday mode) or year (in EOD mode) boundaries
- **Show value labels -** this decides if value lables for indicator / price chart lines should be displayed. See basic charting guide for explanation what value label is.
- **Candlesticks -** this setting provides detailed control over the appearance of candlesticks. The distinct color may be used to draw part of the candlestick or entire candle may be drawn in the same color as its interior.
- **Drawing**
    - **Return to select mode after drawing** - when checked current tool is deactivated after drawing and select mode is entered, when unchecked currently selected drawing tool remains active after drawing (allows to plot one study after another, note that the same effect can be achieved even if this box is checked - it is enough to hold down SHIFT key while drawing and the tool will remain active)

- ♦ **Auto-select last drawn object** - this useful feature automatically selects recently drawn object. This allows to hit ALT+ENTER to display properties box immediately without need to click on the study, and allows to Copy the study via CTRL+C also without additional click
- ♦ **Snap to price % threshold** - defines how far price 'magnet' works, it will snap to price when the mouse is nearer than % threshold from H/L/C price

- **Miscellaneous**
    - ♦ **Ask for parameters of newly inserted indicators** - when checked AmiBroker will automatically display Parameters window each time you insert new indicator or overlay one indicator over another.
    - ♦ **Ask for confirmation when deleting indicator sections** - when checked AmiBroker will ask you to confirm deletion of any overlaid indicator section (applies to indicators created via drag-and-drop). Please note that deletion of indicator section modifies the underlying formula. More on this in Tutorial: Drag-and-drop
    - ♦ **Max number of chart sheets** - defines how many chart sheets (tabs) should be available. More information on chart sheets is in the Tutorial section here. Note that this setting will take effect after restart.
-

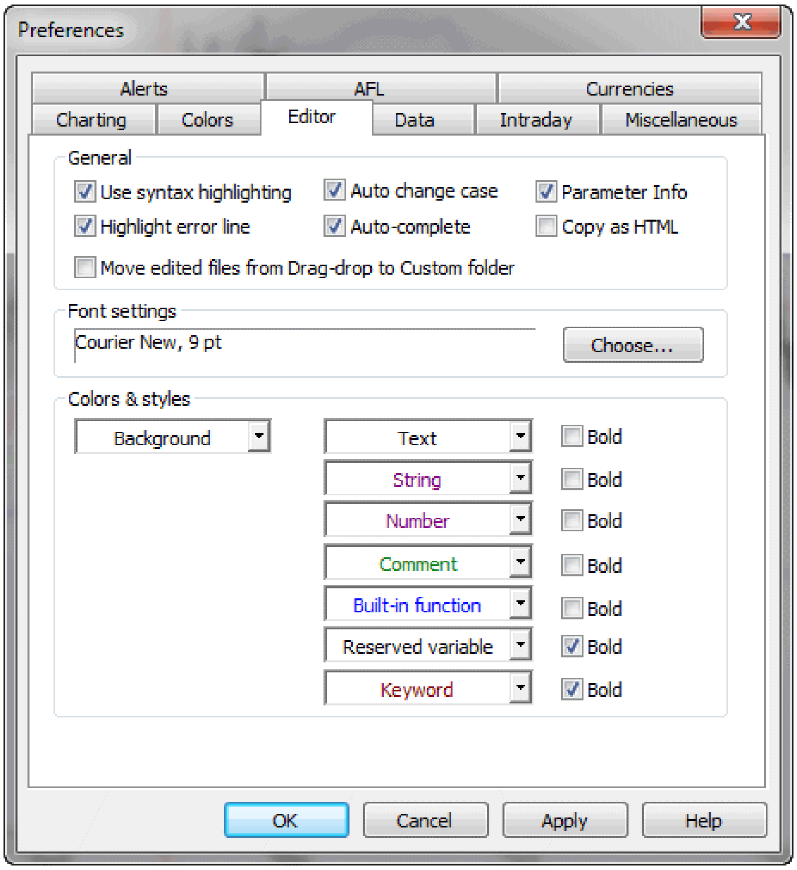


Up candlestick:
Down candlestick:
Grid:
Axes:
Background:
Risk/yield map:
Study (trendline):
Selector line:
☑ Bold

Color tab - allows to define colors for particular chart element.

The controls provide user definable color selection for charts, grid & background.

Palette editor - allows to modify custom colors that can be referenced later via colorCustom0..colorCustom15 constants



General
- [✓] Use syntax highlighting
- [✓] Auto change case
- [✓] Parameter Info
- [✓] Highlight error line
- [✓] Auto-complete
- [ ] Copy as HTML
- [ ] Move edited files from Drag-drop to Custom folder

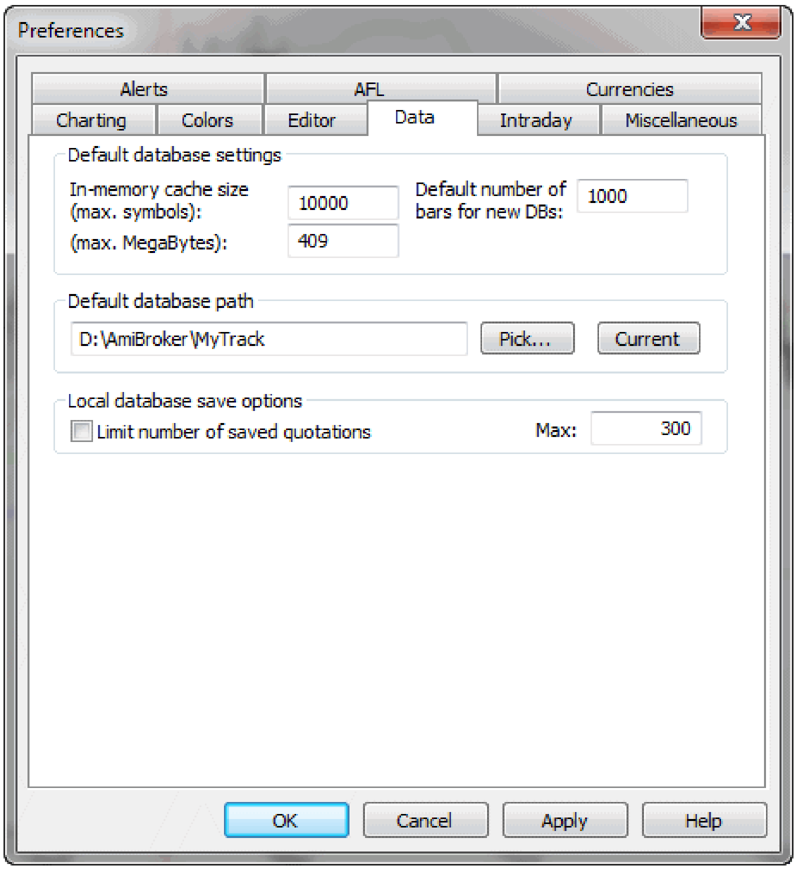
Font settings
Courier New, 9 pt
Choose...

Colors & styles

Background	Text	[ ] Bold
	String	[ ] Bold
	Number	[ ] Bold
	Comment	[ ] Bold
	Built-in function	[ ] Bold
	Reserved variable	[✓] Bold
	Keyword	[✓] Bold

Editor tab - controls the appearance and features of AFL editor.

- **Use syntax highlighting** - when checked editor automatically colorizes your code (different colors/styles for functions, constants, numbers, etc)
- **Auto-change case -** when checked the function and reserved variable names are automatically capitalised so if you type bARSSince, editor will change it to BarsSince
- **Auto-complete** - when checked you will be able to use auto-completion feature (CTRL+SPACE will auto-complete the word)
- **Parameter info** - when checked the editor will display parameter information tooltip when you type a function name and opening brace
- **Highlight error line -** when checked the formula editor marks the line of code that contains an error with a yellow background (Windows 2000 and XP only)
- **Copy As HTML -** when checked the AFL editor on Edit->Copy / Cut command puts not only plain text and RTF formats to the clipboard but also HTML and DwHTML (Dreamweaver HTML) formats allowing pasting syntax-colorized code to Macromedia Dreamweaver and other HTML-aware applications. Note: rarely (on very few machines) turning this on may cause problems with pasting to Outlook.
- Move edited files from Drag-drop to Custom folder - when checked the editor will automatically move manually edited formulas created by drag-and-drop mechanism inside hidden 'Drag-drop' subfolder to 'Custom' subfolder.
- **Font settings** - allows you to define AFL editor font face and size
- **Colors and styles** - allows you to define what colors and styles will be used to mark certain language elements when syntax highlighting is ON.



Default database settings

In-memory cache size (max. symbols): 10000

(max. MegaBytes): 409

Default number of bars for new DBs: 1000

Default database path

D:\AmiBroker\MyTrack

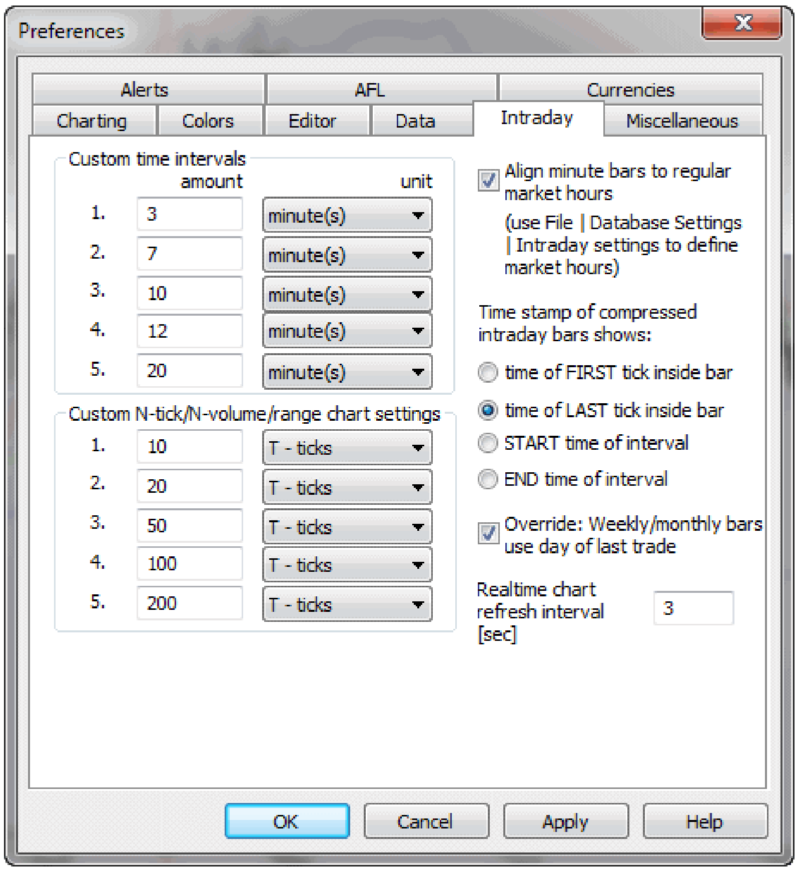
Local database save options

☐ Limit number of saved quotations Max: 300

Data tab - allows you to define default, global values for all databases.

IMPORTANT: some of these settings may **get overwritten by PER-DATABASE settings in File->Database Settings window**. See explanation in Tutorial: Understanding database concepts.

- **Data source:** defines default data souce (for databases that do not specify other source in File->Database Settings)
- **Local data storage:** default setting for **external** databases (this setting gets overwritten by File->Database Settings). If "Enabled" then external data are cached in local files. If "Disabled" then local files do not store external data.
- **In-memory cache (max. symbols)** - defines how many symbols data should be kept in RAM (for very fast access) - this works together with the next setting
- **In-memory cache (max. MegaBytes)** - defines how many MB of RAM should be used for temporary data cache (for very fast access)
- **Number of bars to load -** default setting for **external** databases (this setting gets overwritten by File->Database Settings). Defines how many bars should be loaded from external data souce and kept in AmiBroker. Examples: 10-years EOD: 2600, 60-days intraday 1-minute: 30000 (approx)
- **Limit number of saved quotations** - if this option is ON AmiBroker will save database with limited number of quotations. This prevents the database from growing too much
- **Max. number of saved quotations** - this is the limit itself. Preferable 300 or higher for EOD databases, 3000 or higher for intraday
- **Default database path** - this defines the path to the **database that is loaded on startup. If such database does not exist it will be re-created at startup time.**



Custom time intervals

	amount	unit
1.	3	minute(s)
2.	7	minute(s)
3.	10	minute(s)
4.	12	minute(s)
5.	20	minute(s)

Custom N-tick/N-volume/range chart settings

1.	10	T - ticks
2.	20	T - ticks
3.	50	T - ticks
4.	100	T - ticks
5.	200	T - ticks

[✓] Align minute bars to regular market hours

(use File | Database Settings | Intraday settings to define market hours)

Time stamp of compressed intraday bars shows:

- time of FIRST tick inside bar
- [✓ time of LAST tick inside bar]
- START time of interval
- END time of interval

[✓] Override: Weekly/monthly bars use day of last trade

Realtime chart refresh interval [sec] 3

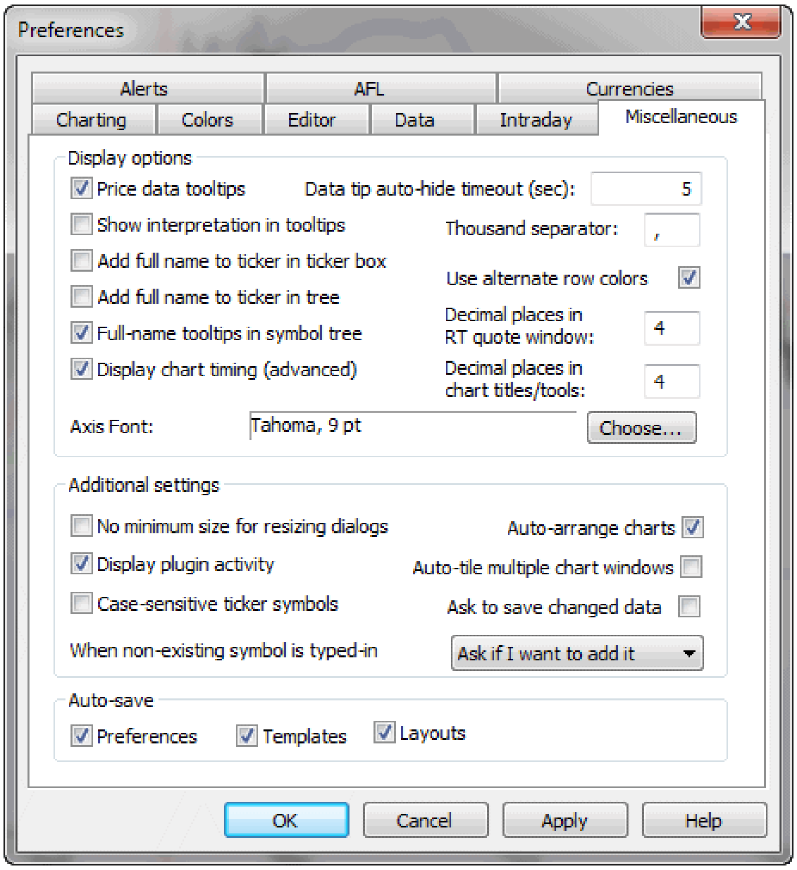
Intraday tab - provides settings for intraday charting

- **Custom time intervals** - allow to define your own N-minute or N-hours intervals (available later from View->Intraday menu)
- **Custom N-tick chart settings** - allow to define your own N-tick charts (available later from View->Intraday menu)
- **Align custom minute bars to regular market hours** - when checked AmiBroker will trim pre-market custom interval bar so new bar will begin exactly when trading hours start. Trading hours can be set per-database in File->Database Settings->Intraday settings. Let's say that we have 45-minute bars. Without this setting we would bet bars starting at 9:00, 9:45, 10:30, 11:15 etc. When this is turned on and trading starts at 9:30 we have guarantee that bars will be aligned to 9:30: 8:45, 9:30, 10:15, 11:00
- **Time compressed bars shows:**
    - ♦ **time of FIRST tick inside bar -** when selected the bar gets the time stamp of the very first trade inside given time slot (bar)
    - ♦ **time of the LAST tick inside bar -** when selected the bar gets the time stamp of the very last trade inside given time slot (bar)
    - ♦ **START time of the interval -** when selected the bar is time-stamped with start time of the time slot (bar). Lets say that 30 minute bar covers 9:00:00..9:29:59. When this is selected AmiBroker will display time of this bar as 9:00
    - ♦ **END time of the interval -** when selected the bar is time-stamped with start time of the time slot (bar). Lets say that 30 minute bar covers 9:00:00..9:29:59. When this is selected AmiBroker will display time of this bar as 9:29:59
- **Realtime chart refresh interval** - defines interval between automatic chart refreshes in real-time mode. By default charts are refreshed every 3 seconds but in very volatile market you may prefer to set it to 1, so charts are refreshed every second in real-time mode.

New in 4.90: To enable 'every tick' chart refresh in **Professional** Edition, go to Tools->Preferences, Intraday tab and enter ZERO (0) into "Intraday chart refresh interval" field. (note Standard Edition won't allow to do that).

Once you enter zero, AmiBroker will refresh all charts with every new trade arriving provided that the formulasyou use execute fast enough. If not, it will dynamically adjust refresh rate to maintain maximum possible refresh rate without consuming more than 50% of CPU (on average). So for example if your charts take 0.2 sec to execute AmiBrokerwill refresh them on average 2.5 times per second.

Note: built-in Windows Performance chart shows cumulated CPU consumption for all processes, to display PER-PROCESS CPU load use SysInternals free software http://www.sysinternals.com/Utilities/ProcessExplorer.html



Display options

- [✓] Price data tooltips
- Data tip auto-hide timeout (sec): 5
- [ ] Show interpretation in tooltips
- Thousand separator: ,
- [ ] Add full name to ticker in ticker box
- Use alternate row colors [✓]
- [ ] Add full name to ticker in tree
- Decimal places in RT quote window: 4
- [✓] Full-name tooltips in symbol tree
- [✓] Display chart timing (advanced)
- Decimal places in chart titles/tools: 4
- Axis Font: Tahoma, 9 pt Choose...

Additional settings

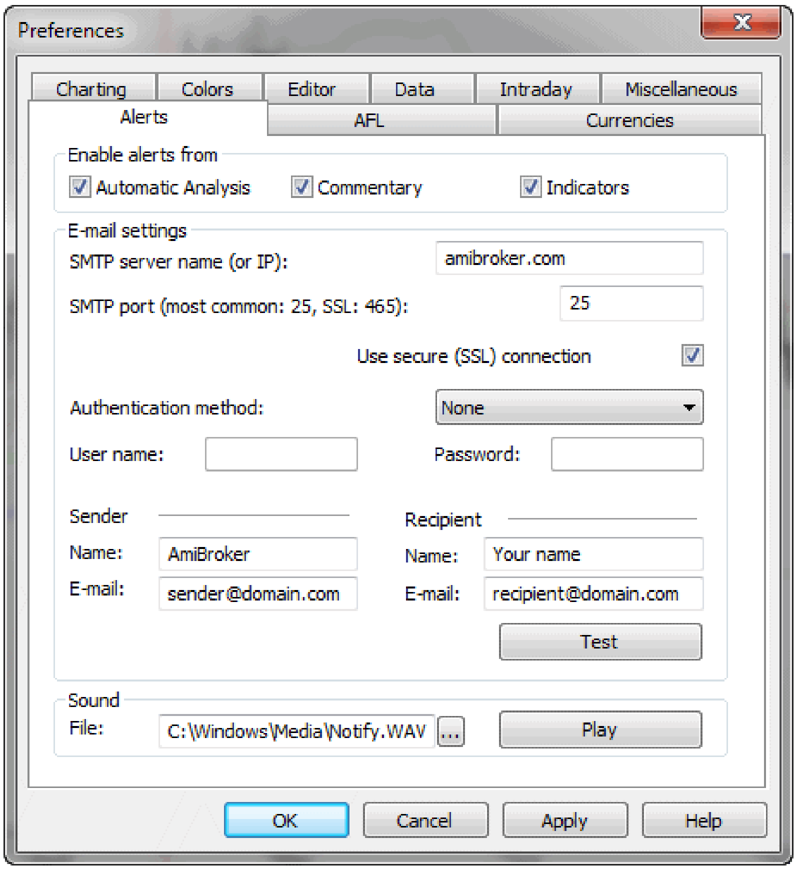
- [ ] No minimum size for resizing dialogs
- Auto-arrange charts [✓]
- [✓] Display plugin activity
- Auto-tile multiple chart windows [ ]
- [ ] Case-sensitive ticker symbols
- Ask to save changed data [ ]
- When non-existing symbol is typed-in: Ask if I want to add it

Auto-save

- [✓] Preferences [✓] Templates [✓] Layouts

- **Price data tooltips**
  if checked small tooltips will appear when you hover over the chart displaying selected bar date, prices / indicator values
- **Show interpretation in tooltip**
  if checked data tooltips will include also interpretation text that is normally displayed in the Interpretation window.
- **Data tip auto-hide timeout**
  defines how many seconds data tooltip should remain on the screen if you don't move your mouse.
- **Add full name to ticker in the ticker box**
  when checked the ticker box displays not only symbol but also full name of the issue

- **Add full name to ticker in the tree**
  when checked the workspace tree displays not only symbol but also full name of the issue
- **Full-name tooltips in symbol tree**
  when checked then full name of symbol is displayed in the tooltip that appears when you move your mouse over symbol in the symbol tree.
- **Data-tip auto-hide timeout**
  defines time in seconds how long data tooltip (that shows values of indicators) will be displayed when mouse cursor does not move
- **Thousand separator**
  defines thousand separator for number displayed on charts and all list-views.
- **Decimal places in RT quote window**
  defines how many decimal places should be displayed in Real Time quote window.
- **Axis font**
  defines font face and size to be used for chart axis and text tool
- **No minimum size for resizing dialogs**
  when checked it allows to size dialogs below the minimum size (so some controls become invisible)
- **Display plugin activity**
  when checked AmiBroker displays information about accessing data plugin in the status bar
- **Case sensitive ticker symbols**
  when checked ticker symbols are case sensitive. In other words INTC and Intc and iNTc are considered DIFFERENT. This is required for some Canadian symbols for example. Please use with caution. If your exchange do not use case-sensitive tickers please make sure it is UNCHECKED.
- **Auto-arrange charts**
  if this option is on chart windows are scaled and arranged to fit the screen after every opening/closing chart window.
- **Auto-tile multiple chart windows**
  when checked multiple chart windows are always tiled **vertically** on every resize of the main application window.
- **Ask to save changed data**
  when checked AmiBroker asks if you want to save modified data on exit. When unchecked AmiBroker saves modified data without asking.
- **Save on exit: Preferences, Templates, Layouts**
  controls which settings should be saved automatically on exit



Enable alerts from
[✓] Automatic Analysis [✓] Commentary [✓] Indicators

E-mail settings
SMTP server name (or IP): amibroker.com
SMTP port (most common: 25, SSL: 465): 25
Use secure (SSL) connection [✓]
Authentication method: None
User name: Password:

Sender
Name: AmiBroker
E-mail: sender@domain.com

Recipient
Name: Your name
E-mail: recipient@domain.com

Sound
File: C:\Windows\Media\Notify.WAV

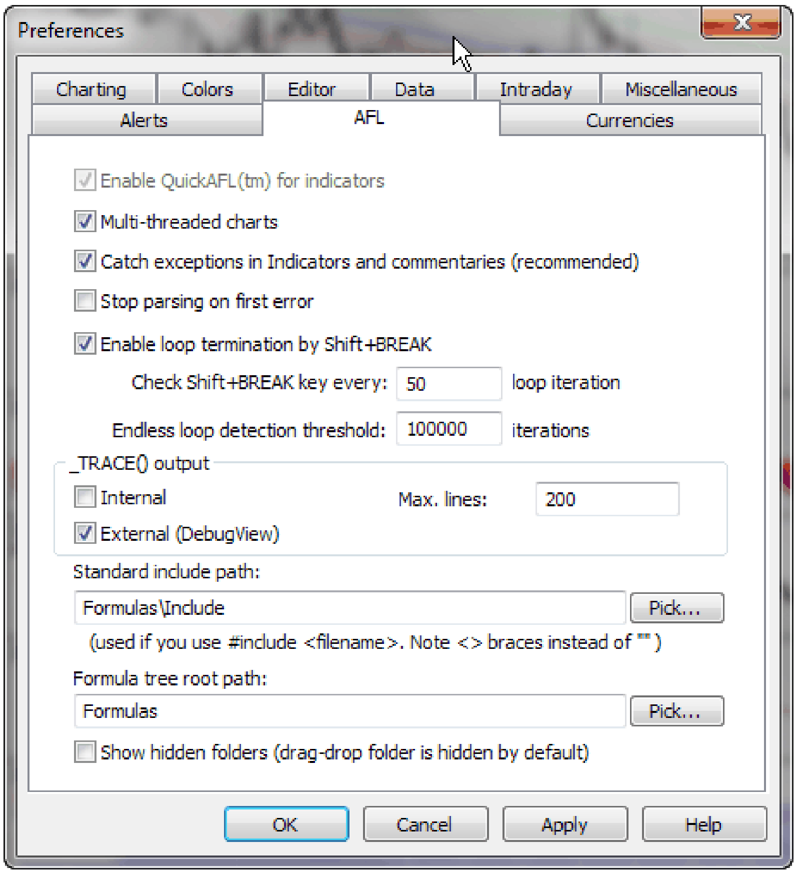
Alerts tab - It allows to define e-mail account settings, test sound output and define which parts of AmiBroker can generate alerts via AlertIF function.

E-mail settings page now allows to choose among most popular authorization schemes like: AUTH LOGIN (most popular), POP3-before-SMPT (popular), CRAM-MD5, LOGIN PLAIN. Version 5.30 allows also to use SSL (secure connection) used by GMail for example. For more information about setting up with GMail see Tutorial: Formula based alerts.

Enable alerts from checkboxes allow you to selectively enable/disable alerts generated by Automatic analysis, Commentary/Interpretation and custom indicators.

Keyboard tab

- keyboard tab has been moved to Tools->Customize dialog



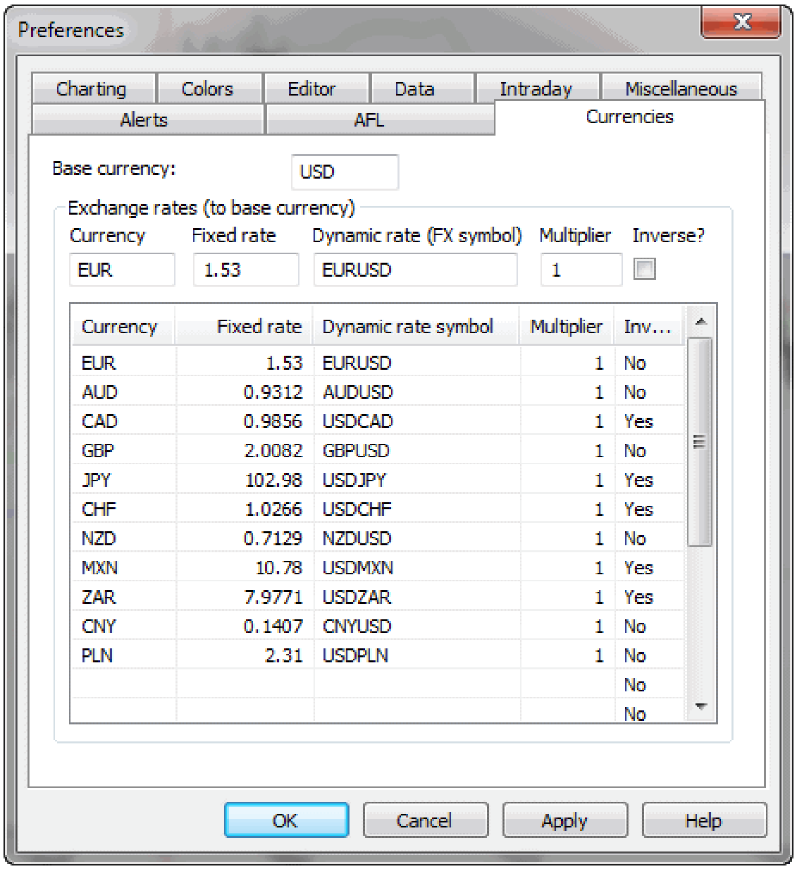
[✓] Enable QuickAFL(tm) for indicators
[✓] Multi-threaded charts
[✓] Catch exceptions in Indicators and commentaries (recommended)
[ ] Stop parsing on first error
[✓] Enable loop termination by Shift+BREAK
Check Shift+BREAK key every: 50 loop iteration
Endless loop detection threshold: 100000 iterations
_TRACE() output
[ ] Internal Max. lines: 200
[✓] External (DebugView)
Standard include path: Formulas\Include
(used if you use #include <filename>. Note <> braces instead of "")
Formula tree root path: Formulas
[ ] Show hidden folders (drag-drop folder is hidden by default)

AFL tab

- **Multi-threaded charts -** enables multi-threaded execution of AFL in charts/indicators. Multi-threading allows to maximize speed and utilisation of modern multi-core / multi-CPU computers. For example on 8-core Intel i7 CPU your charts will run upto 8 times faster than in version 5.30. In version 5.40 the AFL engine has been completely rewritten from ground up to allow multiple instances of the engine running simultaneously. This enables not only multithreading but also enhances responsiveness of entire application, as even badly-written user formula used in a chart is not able to lock or slow the rest of the program. Multi-threading is ON by default. It can be turned off by unchecking this box but it is strongly discouraged. Multi-threading should be turned ON if you want AmiBroker to operate at full speed.
- **Catch system exceptions in Indicators and commentaries** - when checked all exceptions (run-time errors) are catched by the indicator drawing code, so no Bug Recovery window appears. Instead exception information is displayed inside chart pane. It is recommended to have this turned ON especially when you use real-time data
- **Stop parsing on first error** - when checked parser stops further code analysis on first encountered error so only one (first) error is displayed in the formula editor error list. If it is unchecked then parser will list all errors found. It is recommended to turn it off.
- **Enable loop termination by Shift-BREAK** - when checked AmiBroker will allow to break any for(), while() and do-while() loop by pressing and holding down SHIFT and BREAK(PAUSE) keys on your keyboard.
- **Check Shift+BREAK key every** - defines how often keyboard state should be checked when loop is executed. Note that specifying small values will make loop execution slower.
- **Endless loop detection threshold** - defines the number of loop iterations after which AmiBroker will terminate the loop with "Possible Endless loop detected" error message. This is useful in situations when the code has infinite loop (due to mistake of the formula author) because it won't allow

AmiBroker to hang due to infinite looping

- **Standard include path** - the default path to use when #include statement uses < > braces instead of ""
- **Formula tree root path** - the root path of Formula file/directory tree displayed in the Charts tab of Workspace window
- **Show hidden folders** - determines if formula tree should show subfolders with "hidden" attribute (drag-drop folder is created as "hidden" by the setup program)



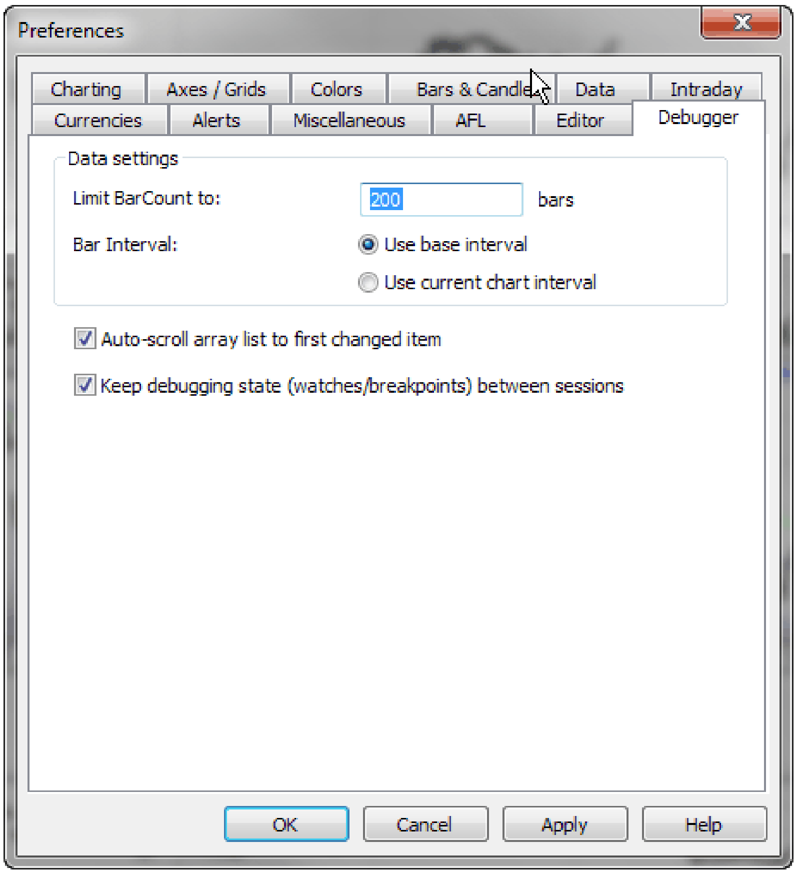
Base currency: USD

Exchange rates (to base currency)

Currency	Fixed rate	Dynamic rate (FX symbol)	Multiplier	Inverse?
EUR	1.53	EURUSD	1	☐

Currency	Fixed rate	Dynamic rate symbol	Multiplier	Inv...
EUR	1.53	EURUSD	1	No
AUD	0.9312	AUDUSD	1	No
CAD	0.9856	USDCAD	1	Yes
GBP	2.0082	GBPUSD	1	No
JPY	102.98	USDJPY	1	Yes
CHF	1.0266	USDCHF	1	Yes
NZD	0.7129	NZDUSD	1	No
MXN	10.78	USDMXN	1	Yes
ZAR	7.9771	USDZAR	1	Yes
CNY	0.1407	CNYUSD	1	No
PLN	2.31	USDPLN	1	No
				No
				No

Debugger tab



Data settings

Limit BarCount to: 200 bars

Bar Interval: [✓ Use base interval] Use current chart interval

[✓] Auto-scroll array list to first changed item

[✓] Keep debugging state (watches/breakpoints) between sessions

The Debugger settings are as follows:

- **Limit BarCount to** - defines maximum number of bars in arrays (BarCount) used during debugging (**by default it is 200 bars**)
- **Bar interval** - decides if debugger uses Base time interval or current Chart interval. The default is "**Use base time interval**" because there can be no chart open at all!
- **Auto-scroll to first changed item** - when this is ON, the "Arrays" list in the Watch window is scrolled automatically to first array item that has changed (so you don't need to locate elementsm that changed manually)
- **Keep debugging state** - when this is ON, AFL editor saves the debug state (breakpoints and watches) and bookmarks in the .dbg file along with the formula when closing the editor and restores the state when formula is reopened. Note that .dbg file is automatically deleted by AmiBroker when there are no breakpoints, no bookmarks and no watches.

Currencies tab

This page allows to define base currency and exchange rates (fixed or dynamic) for different currencies. This allows to get correct backtest results when testing securities denominated in different currency than your base portfolio currency. For more details please check Tutorial: Pyramiding and multiple-currency support in the backtester.

How does AB know whether I want the fixed or dynamic quote?

There are following requirements to use currency adjustements:
a) Symbol->Information, "Currency" field shows currency different than BASE currency

b) Appropriate currency (defined in Symbol) has matching entry in Preferences->Currencies page
c) the dynamic rate "FX SYMBOL" defined in the preferences EXISTS in your database and HAS QUOTES for each day under analysis range.

What is "INVERSE" check box for in the preferences?

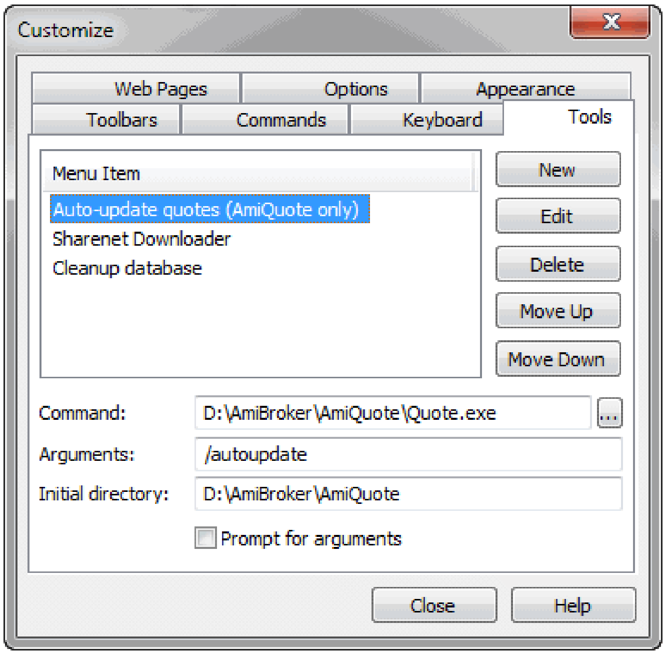
Let's for example take EURUSD.

When "USD" is your BASE currency then EUR exchange rate would be "straight" EURUSD fx (i.e. 1.3).
But when "EUR" is your BASE currency then USD exchange rate would be INVERSE of EURUSD (i.e. 1/1.3).
Opposite would be true with FX rates like USDJPY (which are already "inverse").

Customize tools window

This dialog allows you to customize the User Interface. It can be invoked from *Tools->Customize* menu.

In "Tools" tab you define custom tool menu items:



Auto-update quotes (AmiQuote only)
Sharenet Downloader
Cleanup database

Command: D:\AmiBroker\AmiQuote\Quote.exe
Arguments: /autoupdate
Initial directory: D:\AmiBroker\AmiQuote
Prompt for arguments

You can launch executable files (.exe), script files (.js, .vbs), web pages (.html) and any other registered file types from the tools menu. In order to add a new tool you should open this dialog and click "New" button. Then enter the tool name, command (by hand or using file dialog) optional arguments and initial directory. If you check "Prompt for arguments" checkbox AmiBroker will ask for program's arguments each time

Version 5.60 brings new **#import** command that allows to import ASCII files from local disk or even from remote (web) sources.

In the **Tools->Customize**, "Tools" page, you can now define custom tool that uses new #import command
Command: #import
Arguments: URL to download data from
Initial dir: path to format definition file

This functionality is used by the "Update US symbol list and categories" tool.

Other tabs provide UI customization features described in Customize UI tutorial section.

Keyboard tab

The keyboard tab allows you to define your own keyboard shortcuts.

To assign a shortcut key

On the Tools menu, click Customize, and then click the Keyboard tab.

In the Categories list, select the menu that contains the command to which you want to assign the shortcut key.

In the Commands list, select the command to which you want to assign the shortcut key.

Put the cursor in the Press New Shortcut Key box, press the shortcut key or key combination that you want, and click Assign.
If you press a key or key combination that is invalid, no key is displayed. You cannot assign key combinations with ESC, F1, or combinations such as CTRL+ALT+DEL that are already being used by your operating system.

If you press a key or key combination that is currently assigned to another command and press "Assign" the error message will appear giving you choice to either cancel or re-assign the key shortcut to new command.

To delete a shortcut key

On the Tools menu, click Customize, and then click the Keyboard tab.

On the Categories, and Commands lists, select the location for the shortcut key you want to delete.

In the Current Keys list, select the shortcut key you want to delete and click Remove.

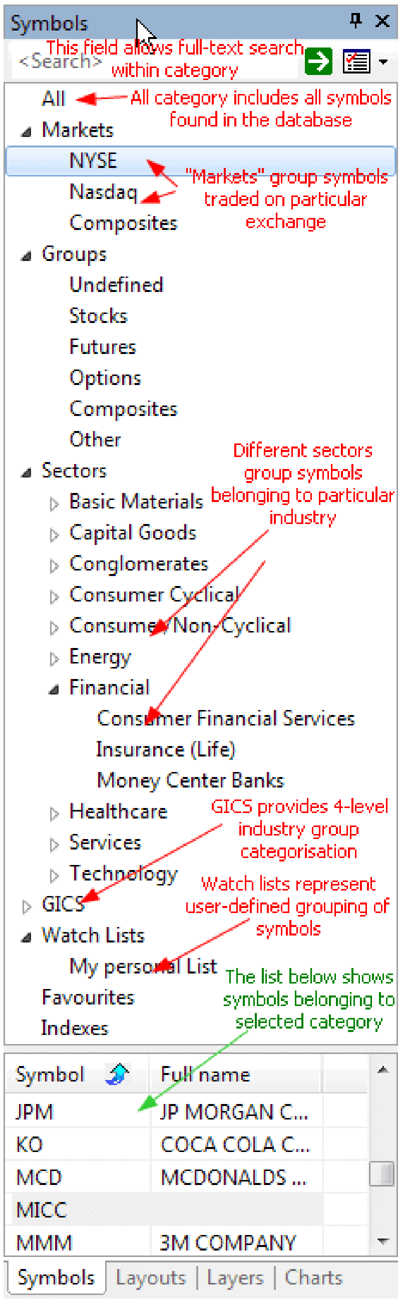
To reset all shortcut keys to their default values

On the Tools menu, click Customize, and then click the Keyboard tab.

Click Reset All.

Symbol tree window

In this windows we have got list of available symbols and categories. Selecting one of them will refresh all opened charts and update information windows. This selection is global for the program i.e. all symbol functions will reference symbol selected in this window.



Symbols window is divided into three parts:
a) search box
b) category tree
c) symbol list

The **search box** allows to perform full text searches (including wildcard matching) against symbol and full name within selected category. So for example if you select "Technology" sector and type A* (letter 'A' and wildcard character *) the symbol list will show all symbols belonging to Technology sector with symbol or full name beginning with letter 'A'. Another example would be tping *-A0-FX - this will return all forex symbols on eSignal database (those ending with -A0-FX substring).

The **category tree** (see the picture) shows different kind of categories.

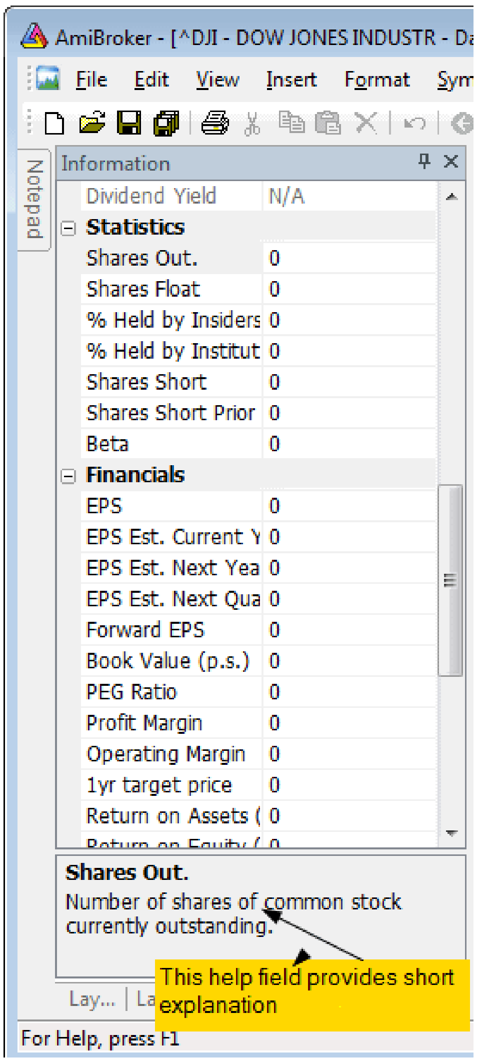
The **symbol list** (bottom part) shows the list of symbols belonging to selected category. The symbol list can be sorted by symbol or by full name. To sort just click on the header row of the list. Once you choose desired sorting order it will be kept for all subsequent category choices and searches. Also the order of columns can be changed so Full name column appears as first one. To re-arrange column, click on the column header, hold down the moust button and drag the column to desired location. Then release mouse button.

Single symbol belongs to MANY categories at the same time. For example AAPL (Apple Inc.) will belong to:

- *Stocks* group category
- *Nasdaq* market category
- *Information* sector category
- *Comp-Computer Mfg* industry category

and may also belong to several watch lists and favorites category. All at the same time. That's why one symbol will appear in many leaves of the workspace symbol tree. Now if you delete the SYMBOL it will of course disappear from ALL categories because you have deleted the symbol itself, not its assignment to category.

Information window


AmiBroker - [^DJI - DOW JONES INDUSTR - Da
File Edit View Insert Format Sym
Notepad
Information
Lay... | La
For Help, press F1

Dividend Yield	N/A
Statistics	
Shares Out.	0
Shares Float	0
% Held by Insiders	0
% Held by Institut	0
Shares Short	0
Shares Short Prior	0
Beta	0
Financials	
EPS	0
EPS Est. Current Y	0
EPS Est. Next Yea	0
EPS Est. Next Qua	0
Forward EPS	0
Book Value (p.s.)	0
PEG Ratio	0
Profit Margin	0
Operating Margin	0
1yr target price	0
Return on Assets (	0
Return on Equity (	0

Shares Out.
Number of shares of common stock currently outstanding.

This help field provides short explanation

This window allows you to display and edit preferences of the symbol.

- **Symbol**
  The short name, used in Select window and with quotation import functions. If you use them, please check if ticker given in this field is the same as used in your quotation datasource
- **Alias**
  The alternative ticker name. It will be useful if you e.g. get the realtime quotes and backfill from two separate datasources, that use different ticker names.
- **Full name**
  Official version of firm name
- **Code**
  Symbol code number
- **Web ID**
  Symbol Web ID - can be used when you define Profile view
- **Address**
  Corporation address
- **Issue**
  Total number of shares

- **Nominal value**
- **Book value**
- **Currency**

- **Market**
  Indicates which market the symbol belongs to.
- **Industry**
  Indicates which industry the symbol belongs to.
- **Group**
  Indicates which Group the symbol belongs to.

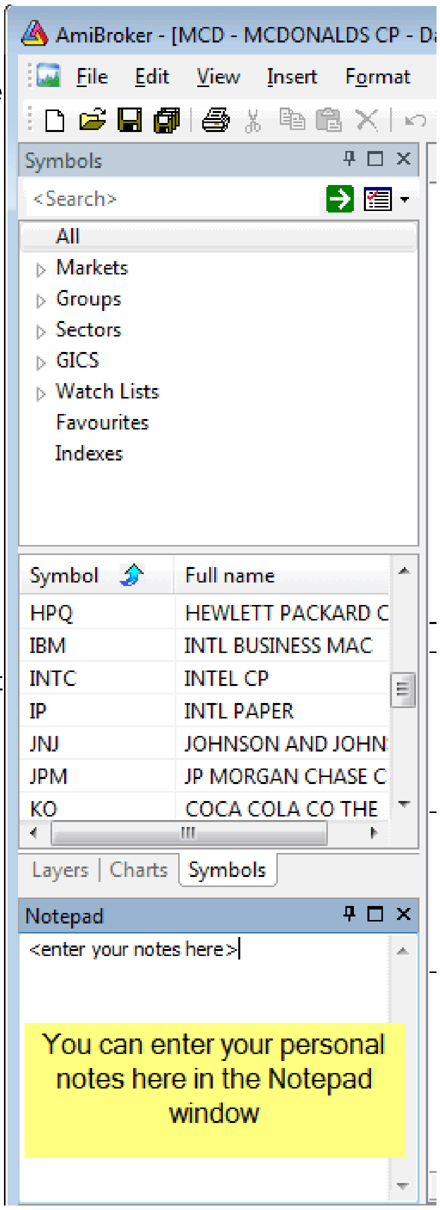
- ***Round lot size***
  Various instruments are traded with various "trading units" or "blocks". For example you can purchase fractional number of units of mutual fund, but you can not purchase fractional number of shares. Sometimes you have to buy in 10s or 100s lots. AmiBroker now allows you to specify the block size on global and per-symbol level.
  You can define per-symbol round lot size in the Symbol->Information page . The value of zero means that the symbol has no special round lot size and will use "Default round lot size" (global setting) from the Automatic Analysis settings page. If default size is set also to zero it means that fractional number of shares/contracts are allowed.
- **Tick size**
  This setting controls the minimum price move of given symbol. You can define it on global and per-symbol level. As with round lot size, you can define per-symbol tick size in the Symbol->Information page (pic. 3). The value of zero instructs AmiBroker to use "default tick size" defined in the Settings page (pic. 1) of Automatic Analysis window. If default tick size is also set to zero it means that there is no minimum price move.
  Note that the tick size setting affects ONLY trades exited by built-in stops and/or ApplyStop(). The backtester assumes that price data follow tick size requirements and it does not change price arrays supplied by the user.
  So specifying tick size makes sense only if you are using built-in stops so exit points are generated at "allowed" price levels instead of calculated ones. For example in Japan - you can not have fractional parts of yen so you should define global ticksize to 1, so built-in stops exit trades at integer levels.
- **Margin deposit** - explained in Backtesting systems for futures contracts
- **Point value** - explained in Backtesting systems for futures contracts

- **Continuous quotations**
  Enables continuous trading for this symbol (this enables candlestick charts and manual entry open/high/low/volume controls and candlestick charts), otherwise symbol is traded with price fixing
- **Index**
  Specifies if symbol belongs to *Indexes* category.
- **Favourites**
  Specifies if symbol belongs to *Favourites* category.
- **Use only local database for this symbol**
  Indicates that symbol is not updated via the plugin in real-time database. This field is checked by default if the symbol is added into realtime database as a result of import from ASCII file (also AmiQuote download). This setting allows you to keep additional symbols in the database and prevent

plugin from overwriting the imported data.

For explanation of Fundamental data fields please read "Tutorial: Using fundamental data" chapter of this guide.

Notepad window



Symbol	Full name
HPQ	HEWLETT PACKARD C
IBM	INTL BUSINESS MAC
INTC	INTEL CP
IP	INTL PAPER
JNJ	JOHNSON AND JOHN
JPM	JP MORGAN CHASE C
KO	COCA COLA CO THE

You can enter your personal notes here in the Notepad window

Notepad window (that you can show/hide using **Window->Notepad** menu) that allows to store free-text notes about particular security. Just type any text and it will be automatically saved / read back as you browse through symbols. Notes are global and are saved in "Notes" subfolder as ordinary
text files.

Notes can be also read and written to using AFL langauge NoteGet and NoteSet functions.

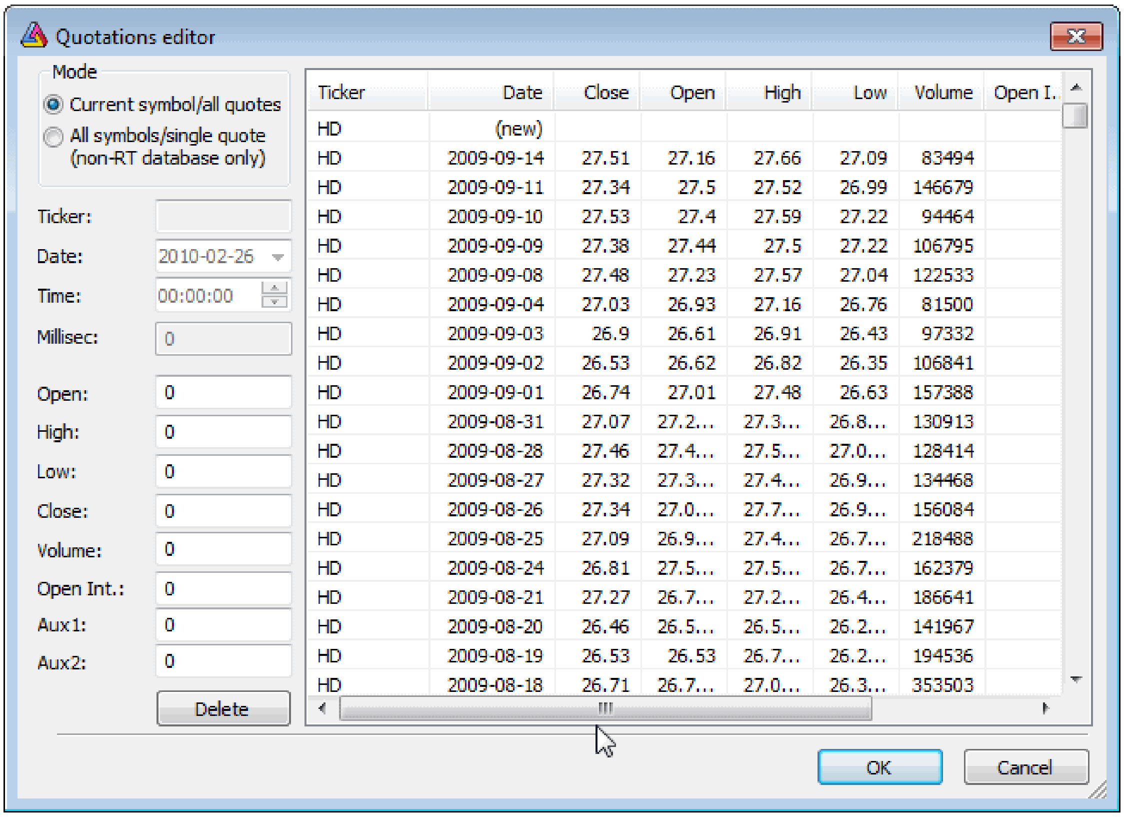
```
NoteGet( "Symbol" );
```
- retrieves note linked to "symbol". If symbol is "" (empty string) then current symbol is used

```
NoteSet( "Symbol", "Text..." );
```
- sets text of the note linked to "symbol".
If symbol is "" (empty string) then current symbol is used.

If you overwrite note from AFL level that is opened at the same time in Notepad editor the editor will ask you (when you switch the focus to it) if it should reload new text or allow to save your manually entered text.

Example:
```
NoteSet("AMD", "Jun 15, 2004: AMD will deliver its
first multi-core processors next year");
```

Quote Editor window



Mode
- [✓ Current symbol/all quotes]
- All symbols/single quote (non-RT database only)

Ticker:	
Date:	2010-02-26
Time:	00:00:00
Millisec:	0
Open:	0
High:	0
Low:	0
Close:	0
Volume:	0
Open Int.:	0
Aux1:	0
Aux2:	0

Ticker	Date	Close	Open	High	Low	Volume	Open I.
HD	(new)						
HD	2009-09-14	27.51	27.16	27.66	27.09	83494	
HD	2009-09-11	27.34	27.5	27.52	26.99	146679	
HD	2009-09-10	27.53	27.4	27.59	27.22	94464	
HD	2009-09-09	27.38	27.44	27.5	27.22	106795	
HD	2009-09-08	27.48	27.23	27.57	27.04	122533	
HD	2009-09-04	27.03	26.93	27.16	26.76	81500	
HD	2009-09-03	26.9	26.61	26.91	26.43	97332	
HD	2009-09-02	26.53	26.62	26.82	26.35	106841	
HD	2009-09-01	26.74	27.01	27.48	26.63	157388	
HD	2009-08-31	27.07	27.2...	27.3...	26.8...	130913	
HD	2009-08-28	27.46	27.4...	27.5...	27.0...	128414	
HD	2009-08-27	27.32	27.3...	27.4...	26.9...	134468	
HD	2009-08-26	27.34	27.0...	27.7...	26.9...	156084	
HD	2009-08-25	27.09	26.9...	27.4...	26.7...	218488	
HD	2009-08-24	26.81	27.5...	27.5...	26.7...	162379	
HD	2009-08-21	27.27	26.7...	27.2...	26.4...	186641	
HD	2009-08-20	26.46	26.5...	26.5...	26.2...	141967	
HD	2009-08-19	26.53	26.53	26.7...	26.2...	194536	
HD	2009-08-18	26.71	26.7...	27.0...	26.3...	353503	

Quote Editor allows editing, deleting and adding quotes.

To add new quote:

- select (new) entry
- enter date/time
- enter price data
- click on the list on the entry other than (new)

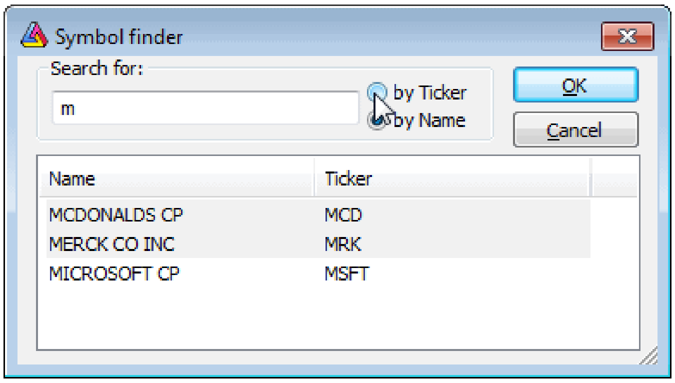
To edit existing quote:

- select quote from the list
- edit price data
- click on the list on the entry other than current

To delete existing quote(s):

- mark one or more quotes (multiple selection possible by holding down SHIFT or CTRL key)
- click "Delete" button

Symbol Finder window (F3)



Search for: m

by Ticker [✓ by Name]

Name	Ticker
MCDONALDS CP	MCD
MERCK CO INC	MRK
MICROSOFT CP	MSFT

Stock finder window allows you to quickly search the database for a symbol by typing the first letters of its full name or ticker. This feature is very useful when you don't know the ticker symbol. The symbol finder is accessible via *Edit->Find symbol*, *Symbol->Find* menus or by pressing F3 key.

To find a symbol just type one or more letters in the **Search for** box. Choose **by Name** if you want to perform full name search or choose **by Ticker** if you want to look up for the ticker. When you type the letters in the edit box appropriate symbols will appear in the list. You can click on the item to choose one or you can just press ENTER key to select the first one. Note that searching starts when the edit box contains at least 1 character - if it is empty no symbol is shown in the list.

Using Web Research window

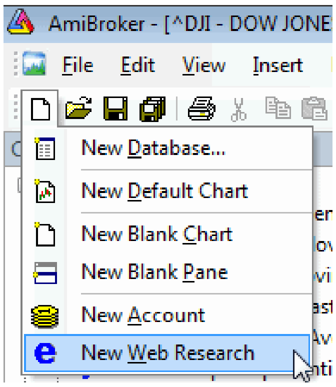
Web Research window allows you to view on-line news, research, profiles, statistics and all kind of information related to currently selected symbol available over the Internet (World Wide Web). Using Web Research instead of plain web browser has speed advantage as you don't need to type complicated/long addresses (URLs) each time you need to get desired information.

Web Research window introduced in version 4.90, replaces and enhances previously available Profile window. Now it allows unlimited number of user-definable web research (profile) pages, browsing to any web page (just type URL), tab-browsing, opening multiple pages at once, selective auto-synchronization.

Web-Reasarch uses Internet Explorer engine so you can be sure that pages are rendered with the same quality you would get from stand-alone browser.

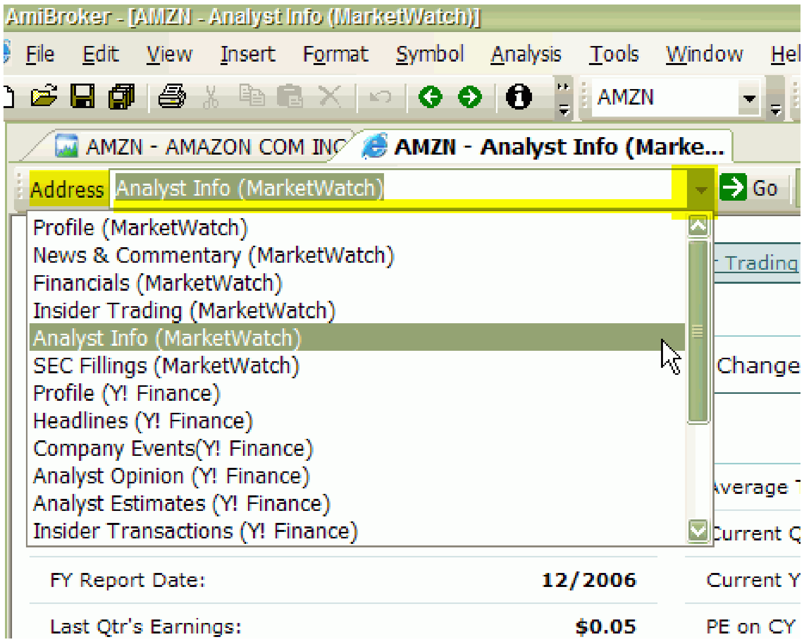
OPEN NEW WEB RESEARCH WINDOW

Use **File->New->Web Research** menu to create new web research window

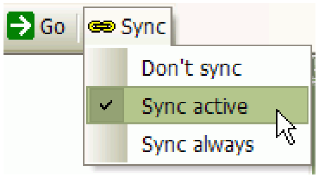


PICKING PRE-DEFINED WEB RESEARCH PAGE:

To display any pre-defined web research page, simply click on the drop down arrow in the Address combo-box and pick one item from the list. Once you do so, the web page relevant to currently selected symbol will be automatically displayed.


AmiBroker - [AMZN - Analyst Info (MarketWatch)]
File Edit View Insert Format Symbol Analysis Tools Window Hel
AMZN
AMZN - AMAZON COM INC
AMZN - Analyst Info (Marke...
Address
Analyst Info (MarketWatch)
Go
Profile (MarketWatch)
News & Commentary (MarketWatch)
Financials (MarketWatch)
Insider Trading (MarketWatch)
Analyst Info (MarketWatch)
SEC Fillings (MarketWatch)
Profile (Y! Finance)
Headlines (Y! Finance)
Company Events(Y! Finance)
Analyst Opinion (Y! Finance)
Analyst Estimates (Y! Finance)
Insider Transactions (Y! Finance)
Trading
Change
Average
Current Q
FY Report Date: 12/2006
Current Y
Last Qtr's Earnings: $0.05
PE on CY

Now you can specify if and when displayed page should change automatically if you select different symbol.

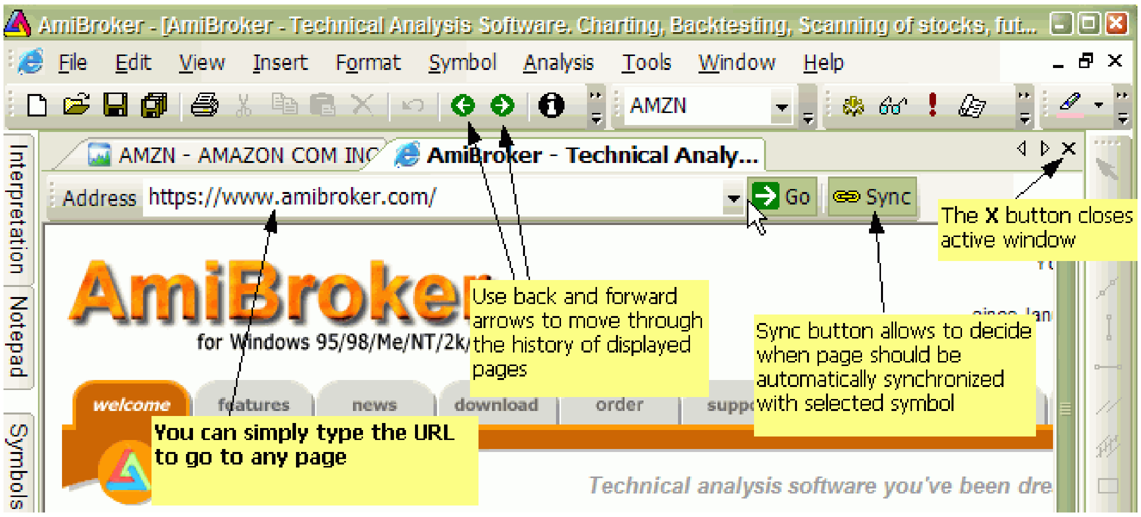


The **Sync** button allows to decide when page should be automatically synchronized with currently selected symbol.

- **Don't sync** - means that page should not be synchronized with currently selected symbol at all
- **Sync active** - means that page should be synchronized ONLY when it is currently active or becomes active (by user clicking on given tab) - this is recommended setting for web-research profiles since it conserves bandwidth and resources (not active pages are not synchronized and do not consume any bandwidth)
- **Sync always** - means that page is synchronized with currently selected symbol always, no matter if it is active or not.

NAVIGATION

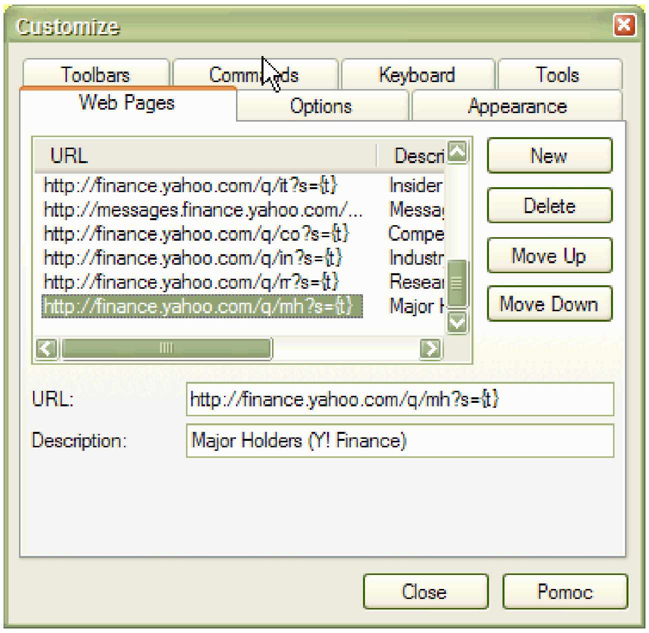
Web Research window operates in a way very similar to stand-alone browser. To display any web page just type the URL address to "Address" field and press ENTER (RETURN) key. To navigate back and forward in the history use **<-** and **->** buttons.


AmiBroker - [AmiBroker - Technical Analysis Software. Charting, Backtesting, Scanning of stocks, fut...
File Edit View Insert Format Symbol Analysis Tools Window Help
AMZN
AMZN - AMAZON COM INC
AmiBroker - Technical Analy...
Address https://www.amibroker.com/
Go
Sync
The X button closes active window
Use back and forward arrows to move through the history of displayed pages
Sync button allows to decide when page should be automatically synchronized with selected symbol
You can simply type the URL to go to any page
Interpretation
Notepad
Symbols

To close currently displayed page use regular window close **X** button as shown in the picture above

DEFINING YOUR OWN WEB RESEARCH PLACES

In addition to web-research pre-defined pages you can define any number of your own places. To do so use **Tools->Customize** menu, **Web Pages** tab.



URL	Descri
http://finance.yahoo.com/q/it?s={t}	Insider
http://messages.finance.yahoo.com/...	Messa
http://finance.yahoo.com/q/co?s={t}	Compe
http://finance.yahoo.com/q/in?s={t}	Industr
http://finance.yahoo.com/q/rr?s={t}	Resea
http://finance.yahoo.com/q/mh?s={t}	Major H

URL: http://finance.yahoo.com/q/mh?s={t}

Description: Major Holders (Y! Finance)

To add new place press **New** button, then type the URL template in the **URL** field and web page description in the **Description** field.

The URL template is the web address in that has parts that depend on selected symbol. The URL template is parsed by AmiBroker to make actual URL to the web page. For example to see Yahoo's profiles page you can use following URL template:

http://biz.yahoo.com/p/**{t0}**/**{t}**.html.

Symbols enclosed in brackets **{}** define fields which are evaluated in execution time. **{t0}** symbol is evaluated to the first character of the ticker name and **{t}** is evaluated to the whole ticker name. So if AAPL is selected AmiBroker will generate following URL from above template:

http://biz.yahoo.com/p/a/aapl.html

Then AmiBroker uses built-in web browser (Web Research window) to display the contents of the page.

Special fields encoding scheme

As shown in above example template URL can contain special fields which are substituted at run time by values corresponding to the currently selected symbol. The format of the special field is {*x*} where *x* is describes field type. Currently there are three allowable field types: ticker symbol in original case **{t}**, ticker symbol in lowercase **{s}**, ticker symbol in UPPERCASE **{S}**, alias **{a}**, web id **{i}**. You can specify those fields anywhere within the URL and AmiBroker will replace them with appropriate values entered in the Information window. You can also reference to single characters of ticker, alias or web id. This is useful when given web site uses first characters of, for example, ticker to group the html files (Yahoo Finance site does that), so you have files for tickers beginning with 'a' stored in subdirectory 'a'. To reference to single character of the field use second format style {*xn*} where *x* is field type described above and *n* is zero-based index of the character. So **{a0}** will evaluate to the first character of the alias string. To get first two characters of a ticker write simply **{t0}{t1}**. Note about web id field: this new field in Information window was added to handle situations when web sites do not use ticker names for storing profile files. I found some sites that use their own numbering system so they assign unique number to each symbol. AmiBroker allows you to use this nonstandard coding for viewing profiles. All you have to do is to enter correct IDs in Web ID field and use appropriate template URL with **{i}** keyword.

Pages stored locally

You may want to have all pages stored on your local hard disk. This has an advantage that profiles are accessible instantly but they can take significant amount of storage space and you will need to update them from time to time. To access locally stored files use the following template URL (example, C: denotes drive): file://C:\the_folder_with_profile_files\{t}.html. You are not limited to HTML files, you can use simple TXT files instead. Then create (or download) the .html (or txt) files for each symbol in the portfolio. These files should obey the following naming convention: <ticker>.html. So for example for APPLE (ticker AAPL) the profile should have the name AAPL.html (or AAPL.txt)

Web-based profiles

If you want to display the profiles from remote web pages you will need to find out how they are accessible (the URL to the web page) and how the data for different symbols are accessible. I will describe the problem on the example of Sharenet (www.sharenet.co.za) site providing the data for companies listed on Johannesburg Stock Exchanges. Sharenet provides company information that is accessible at the following address (URL):

http://www.sharenet.co.za/free/free_company_na.phtml?code=**JSECODE**&scheme=default

The problem is that database provided by Sharenet uses long ticker names and **JSECODE** is a short symbol code. For example for "Accord Technologies" company the ticker in Sharenet database is ACCORD but the code is ACR. To solve the problem we will need to use **Web ID** field in the symbol Information window. If you have Sharenet database just choose the ACCORD from the ticker list, open *Symbol->Information* window and enter ACR to the **Web ID** edit box and click OK. Then enter the following URL template to the **URL** edit box:

http://www.sharenet.co.za/free/free_company_na.phtml?code={i}&scheme=default

To be 100% sure please select the text above with a mouse. Then copy it to the clipboard (Edit->Copy, CTRL-C). Then switch to AmiBroker and click on the Profile URL edit box. Delete everything from it and press CTRL-V (this will paste the text). Type "Sharenet" into **Description** field.

Please note that we have used **{i}** special field in the template that will be replaced by AmiBroker with the text entered in the Web ID field of the symbol information window. Now please select *File->New->Web Research* and pick Sharenet from Address combo box. You should see the profile for ACCORD company.

You can also delete any entry by selecting it from the list and pressing **Delete** button. You can change the order in which pages appear in the Web Research address combo using **Move Up** and **Move Down** buttons (select the item first and then use buttons).

Configuration data are stored in webpages.cfg plain text file that holds any number of URL templates in the form of:

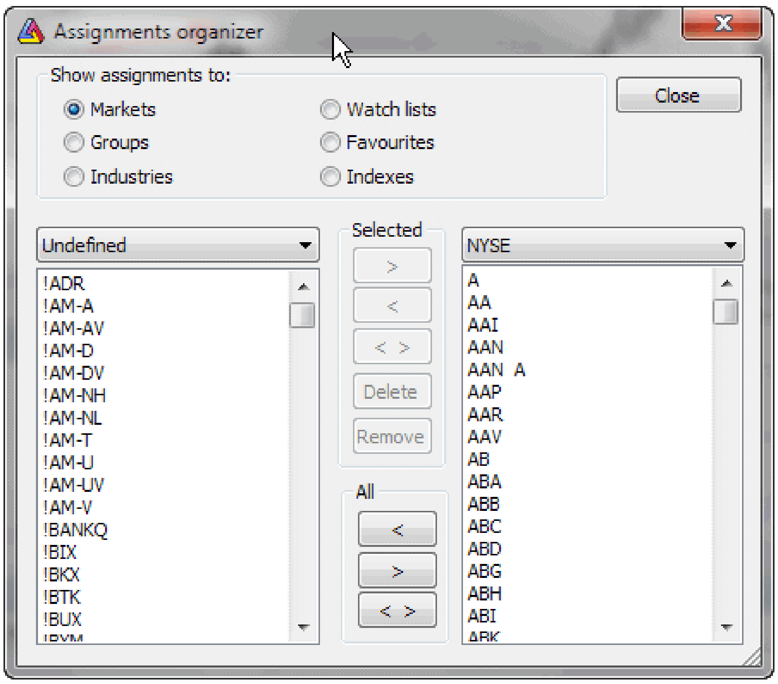
URLTemplate|Description

(each entry in separate line)

Assignment organizer window

In order to make assigning the symbols to categories simpler and faster a new assignment organizer was developed. Now you can simply mark a group of symbols and quickly move them from one category to another.

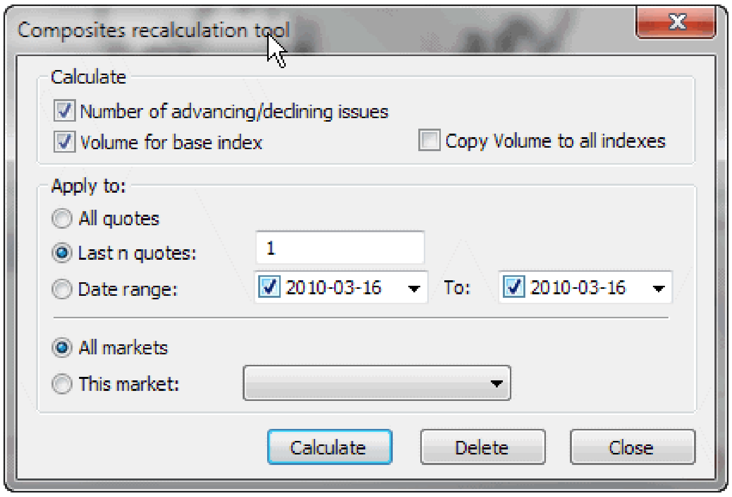
You can also delete multiple symbols using this window. To do so, select one or more symbols from the left pane and click on "Delete" button.



Show assignments to: [✓ Markets] Groups Industries Watch lists Favourites Indexes

Undefined	NYSE
!ADR	A
!AM-A	AA
!AM-AV	AAI
!AM-D	AAN
!AM-DV	AAN A
!AM-NH	AAP
!AM-NL	AAR
!AM-T	AAV
!AM-U	AB
!AM-UV	ABA
!AM-V	ABB
!BANKQ	ABC
!BIX	ABD
!BKX	ABG
!BTK	ABH
!BUX	ABI

Composite recalculation window



[✓] Number of advancing/declining issues
[✓] Volume for base index
[ ] Copy Volume to all indexes

Apply to:
( ) All quotes
(●) Last n quotes: 1
( ) Date range: [✓] 2010-03-16 To: [✓] 2010-03-16

(●) All markets
( ) This market:

This dialog allows automatic calculation of number and volume of advancing/declining/unchanged issues. Also possible in this dialog is calculation of volume numbers for indexes if imported incorrectly. Note well, that automatic recalculation of composite data makes only sense when you follow whole exchange (all symbols are included in your database) otherwise this calculation will give wrong results.

In order to calculate the composites in the database it's necessary to set the base index for the market, as it may happen that not all stocks are quoted every businness day. AmiBroker checks the 'base index' quotations dates and tries to find the corresponding quotes of all the stocks belonging to that market, to find out how many issues advanced, declined and not changed at all.

To calculate composities you need to:

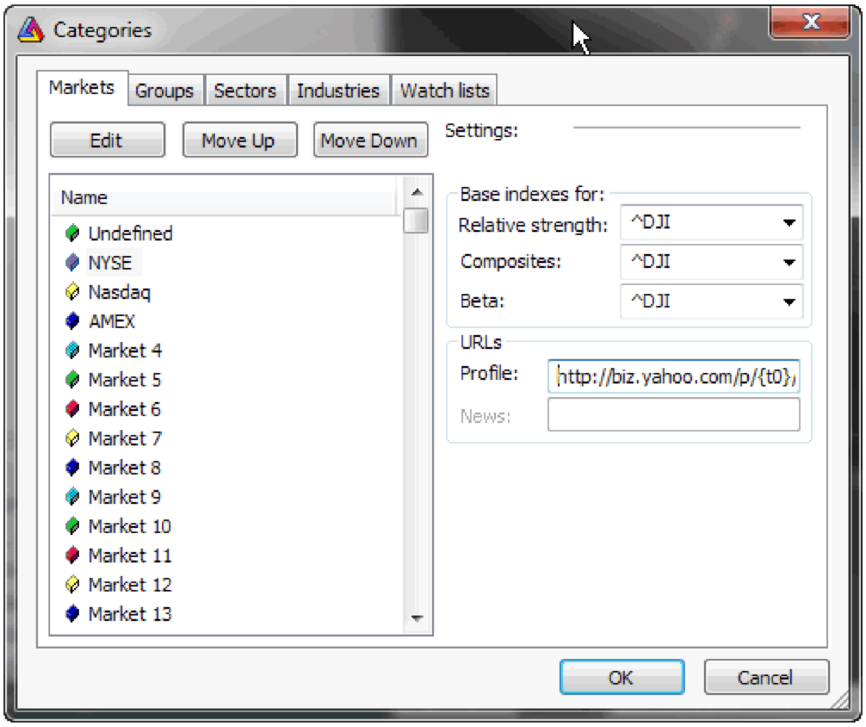
- Open Categories window using Symbol -> Categories menu item.
- Select base index for given market in Markets tab and Base indexes for - Composites combo. For example if you are following NYSE this can by ^DJI (Dow Jones Average) ^DJI must be marked as index in Symbol -> Information and must belong to the same market.
- Choose "Symbol ->Calculate composites" menu item to open the window shown below and mark : *Number of advancing/declining issues* and choose markets that you calculate composities for and the time range.
- Click Calculate.

There are also two additional fields available:

- **Volume for base index**
- **Copy volume to all indexes**

These fields are provided in case you DON'T have real volume data for index quotes. In that case AmiBroker can calculate volume for index as a sum of volumes of all stocks belonging to given market. First option assigns calculated volume only to *base index*, the second copies the volume figure to all indices belonging to given market.

Categories window



Settings:

Name
Undefined
NYSE
Nasdaq
AMEX
Market 4
Market 5
Market 6
Market 7
Market 8
Market 9
Market 10
Market 11
Market 12
Market 13

Base indexes for:

Relative strength:	^DJI
Composites:	^DJI
Beta:	^DJI

URLs

Profile:	http://biz.yahoo.com/p/{t0},
News:	

This dialog, allows you to define names of markets, groups, sectors and industries. For each market you can also define base indexes for calculating relative strength, composite data, beta or web profile URL. The detailed information about categories can be found in Understanding categories chapter of this manual.

To **Edit** the name of certain category, please select it from the list and press '*Edit*' button.

Base indexs for fields allow you to set the index used in calculation of:

- Relative Strength indicator
- Composities via Composite calculation option
- Beta

Profile field allows you to define URL-template for viewing on-line (or off-line) companies' profiles. These URL-templates are market-based, what means you can have different templates for each market. The template is then parsed to create the actual URL to the web page, which will be displayed in an embedded web browser. To learn more read How to set up the profile view chapter.

Re-arranging categories

AmiBroker version 6.10 added ability to re-arrange the order of markets/groups/sectors/industries and watch lists using "**Move Up**" / "**Move down**" buttons. This is non-trivial task as all symbols must be synchronized with the change as when the ordering of categories change then all symbols data must be re-indexed to reflect new order as symbols refer to oridinal position of category.

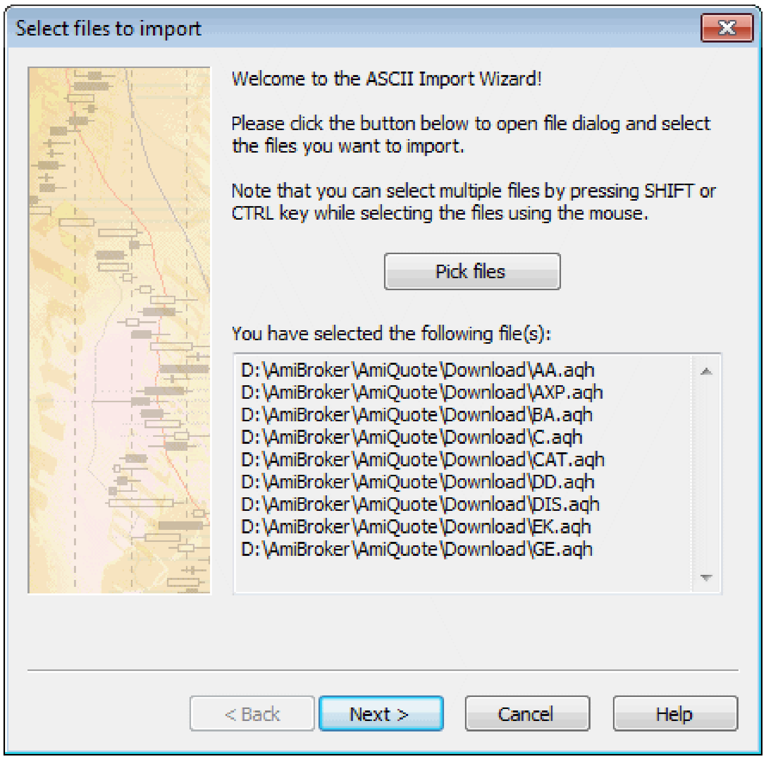
ASCII Import Wizard

ASCII Import Wizard provides an easy way to import your quotation data files as well as define your own import formats for future use. Note that wizard offers only a subset of features available in ASCII importer so it is provided for novice users only.

The wizard guides you through 3 simple steps

1. Picking the files to import
2. Defining fields
3. Additional settings

Step 1. Picking the files



Welcome to the ASCII Import Wizard!

Please click the button below to open file dialog and select the files you want to import.

Note that you can select multiple files by pressing SHIFT or CTRL key while selecting the files using the mouse.

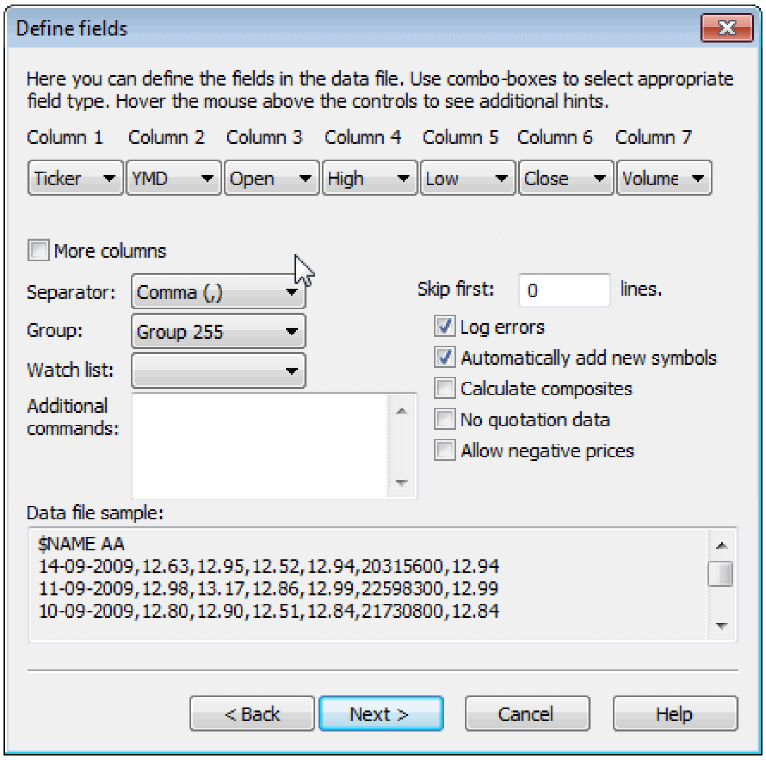
You have selected the following file(s):

```
D:\AmiBroker\AmiQuote\Download\AA.aqh
D:\AmiBroker\AmiQuote\Download\AXP.aqh
D:\AmiBroker\AmiQuote\Download\BA.aqh
D:\AmiBroker\AmiQuote\Download\C.aqh
D:\AmiBroker\AmiQuote\Download\CAT.aqh
D:\AmiBroker\AmiQuote\Download\DD.aqh
D:\AmiBroker\AmiQuote\Download\DIS.aqh
D:\AmiBroker\AmiQuote\Download\EK.aqh
D:\AmiBroker\AmiQuote\Download\GE.aqh
```

In this step you select the files you want to import. Just click on the **Pick files** button and you will see a file dialog. Browse to the folder where your data files are located and select the file(s). Please note that you can select multiple files by holding CTRL or SHIFT key while clicking on the files. After making your selection please click **Open**

A complete list of files that you have selected will be displayed in the field at the bottom of the wizard window. Please check if the list is correct, if not click "Pick files" to correct your choice.

Step 2. Defining fields


Define fields
Here you can define the fields in the data file. Use combo-boxes to select appropriate field type. Hover the mouse above the controls to see additional hints.
Column 1 Column 2 Column 3 Column 4 Column 5 Column 6 Column 7
Ticker YMD Open High Low Close Volume
More columns
Separator: Comma (,)
Group: Group 255
Watch list:
Additional commands:
Skip first: 0 lines.
Log errors
Automatically add new symbols
Calculate composites
No quotation data
Allow negative prices
Data file sample:
< Back
Next >
Cancel
Help

```
$NAME AA
14-09-2009,12.63,12.95,12.52,12.94,20315600,12.94
11-09-2009,12.98,13.17,12.86,12.99,22598300,12.99
10-09-2009,12.80,12.90,12.51,12.84,21730800,12.84
```

In this step you define the types of fields in the data file. For your convenience date file sample is shown (a few first lines of the first selected file) at the bottom of the window.

To define fields please select appropriate field types from **Column *N*** combo-boxes. For example, if the first field (column) in your data file is a symbol ticker please select "Ticker" from **Column 1** combo box. If second field in your data file is a date in Year-Month-Day format please select "YMD" from the second combo-box. You can select also DMY for Day-Month-Year dates, MDY for Month-Day-Year dates. Other field types available from the wizard are: "Open", "Close", "High", "Low" for the prices and "Volume".

Note about the dates: AmiBroker recognizes both 4 digit and 2 digit year dates. As for months both numbers and three letter codes ("Jan", "Feb", ...) are allowed. Also day, month and year may be separated by any of the following characters: / (slash), \ (backslash), - (minus sign) or may not be separated at all. All you have to do is to specify the order: DMY, MDY, YMD. For example valid YMD dates are (31th December 2000):

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

If your file has more than 7 columns please check **More columns** box and you will see additional combo-boxes.

The remaining controls here are:

Group: here you should select to which group new symbols are added

Watch list: here you should select to which watch list new symbols are added (if empty - they are not added to any watch list)

Separator: here you should select the character used as a field separator (comma is the most often)

Skip lines: this tells AmiBroker how many initial lines should be skipped (ignored) - for example a few first lines of the file should contain a comment or other information that should be ignored, and this is the place to define this

Log errors: this tells AmiBroker that it should log all errors to the file (import.log). In case of any errors this log will be displayed to the user after finishing import process.

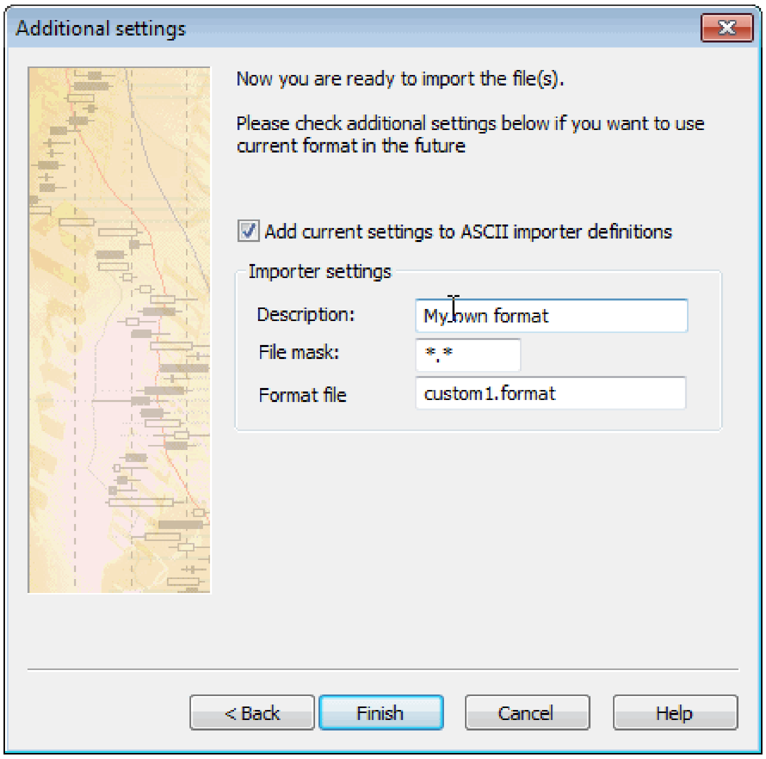
Automatically add new symbols: this tells AmiBroker to add the symbols that appear in the data file but do not exist yet in AmiBroker database.

Calculate composites: this tells AmiBroker to calculate advance/decline figures and volume for indexes after import (this requires composites to be set up properly before importing)

Allow negative prices: this tells AmiBroker to allow negative number in close, open, high, low fields. By default zero and negative values are NOT allowed.

No quotation data: allows to import data that do not contain prices. For example ticker lists and/or categories.

Step 3. Additional settings


Additional settings

Now you are ready to import the file(s).

Please check additional settings below if you want to use current format in the future

[✓] Add current settings to ASCII importer definitions

Importer settings

Description:	My own format
File mask:	*.*
Format file	custom1.format

By default the format you have defined is for single-use only. It is OK for novice users and for experimenting with the wizard.

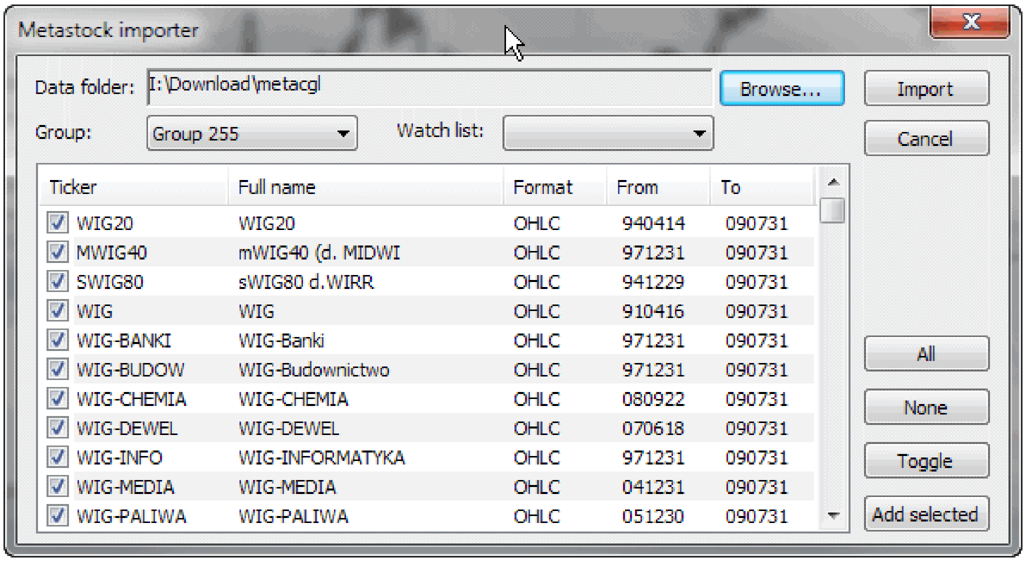
If you, however, want to make your definition permantent and available in the future via ASCII importer you should check **Add current settings to ASCII importer definitions** box. Then you should enter the **Format description**, **File mask** and **Format file name** (or you can accept automatically generated defaults). If you do so, you will be able to use the format defined in the ASCII importer window - just by selecting your own format (as typed in **Format description** field) from the "Files of type" combo of a file dialog.

Whatever you decide, you should click "Finish" button in order to start importing your data.

Metastock importer window

IMPORTANT NOTE: Metastock importer should be used ONLY if you want to import MS data to native, local AmiBroker database once. If you want AmiBroker to just read Metastock database ***DIRECTLY*** *without need to import new data over and over please set up your database WITH METASTOCK PLUGIN as described in the Tutorial.*

NOTE 2: if you setup your database with the ***MS plugin*** *you should NOT use Metastock importer, because there is no point in using it when your data are already fed by the plugin.*



Ticker	Full name	Format	From	To
✓ WIG20	WIG20	OHLC	940414	090731
✓ MWIG40	mWIG40 (d. MIDWI	OHLC	971231	090731
✓ SWIG80	sWIG80 d.WIRR	OHLC	941229	090731
✓ WIG	WIG	OHLC	910416	090731
✓ WIG-BANKI	WIG-Banki	OHLC	971231	090731
✓ WIG-BUDOW	WIG-Budownictwo	OHLC	971231	090731
✓ WIG-CHEMIA	WIG-CHEMIA	OHLC	080922	090731
✓ WIG-DEWEL	WIG-DEWEL	OHLC	070618	090731
✓ WIG-INFO	WIG-INFORMATYKA	OHLC	971231	090731
✓ WIG-MEDIA	WIG-MEDIA	OHLC	041231	090731
✓ WIG-PALIWA	WIG-PALIWA	OHLC	051230	090731

Metastock importer opens AmiBroker to very rich source of historical data. The importer supports both old Metastock 6.5 and new 7.x (XMASTER) formats.

Basically Metastock data consist of:

- MASTER/EMASTER file which holds general information about the tickers, stock names, etc.
- F1.DAT....Fxx.DAT files which hold actual quotation data

The MASTER/EMASTER file is essential because it holds the references to Fxx.DAT files. Fxx.DAT files store only quotations in either 5 field (date/high/low/close/volume), 6 or 7 field (date/open/high/low/close/volume/openinterest) format. As you see MASTER/EMASTER and Fxx.DAT files are closely connected and you need them all to import the data.

Usage

To import Metastock data you should do the following:

- Choose *Metastock import* from the menu
- Using the directory requester (***Browse***...) select the location of data in Metastock format (the directory with MASTER/EMASTER and Fxx.DAT files)
- After choosing proper directory AmiBroker will display the list of available symbols and date ranges. By default all available symbol will be marked for importing (checkmark at the beginning of the list). Now you can exclude some symbol from the import list by clicking appropriate item in the list (checkmark will toggle when you click).

- You can decide to which group and watch list the new symbols are added using **Group** and **Watch List** combos.
- After making your selections push '**Import**' button to start the process of importation.
- During the process you can cancel the operation by clicking '**Abort**' button in the progress window

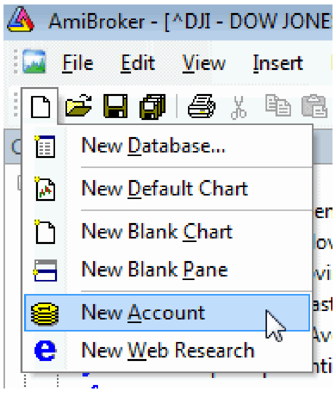
Using account manager

Account manager is a tool for keeping track of your trades and your performance. You are able to enter trades you make, deposit/withdraw funds, check the statistics and historical performance. All transactions are recorded so you will never forget what happened in the past. Account manager allows you to keep track of unlimited number of accounts.

New account manager replaces and enhances functionality provided by portfolio manager in pre-4.90 versions.

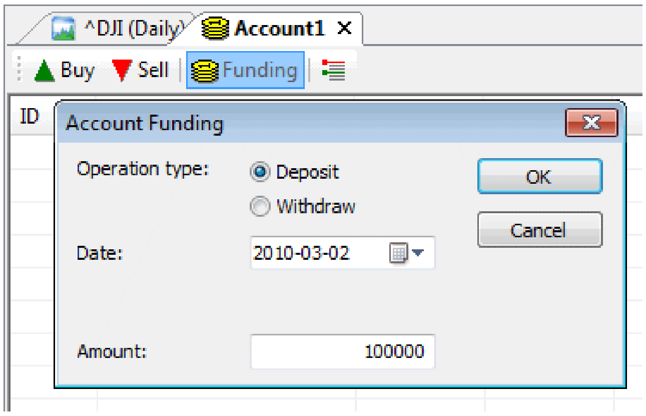
CREATE A NEW ACCOUNT

Use File->New->Account menu to create new account



FUNDING AN ACCOUNT

Before you do any trading, you have to fund your account. To do so press "FUNDING" button on the account manager toolbar, then select "Deposit" as operation type, enter the DATE when you have funded your account and enter the amount.
Note that funding date must PRECEDE any trading, as account manager won't allow you to trade prior to funding date. Initial deposit will show as "initial equity" in summary tab.

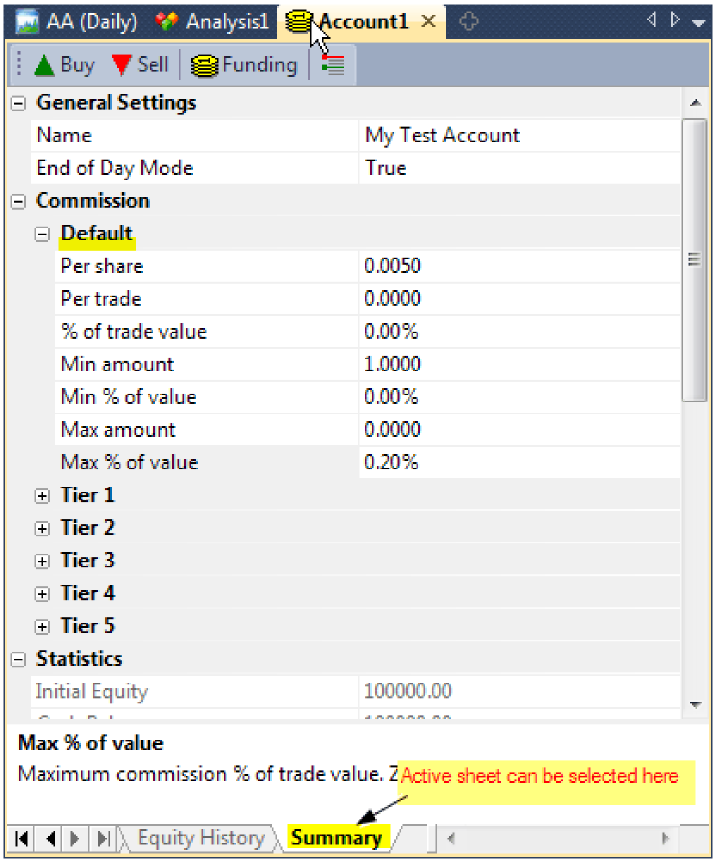


Operation type: [✓ Deposit] Withdraw
Date: 2010-03-02
Amount: 100000

THE SETTINGS

It is good idea to go to "Summary tab" and setup commissions and trading mode. If this account is used for End-of-day trading you should set "EOD Mode" to YES, otherwise (if you trade intraday) you should set "EOD

Mode" to NO. Depending on this setting Buy/Sell dialogs will allowyou to enter date and time of the trade or only date.



General Settings	
Name	My Test Account
End of Day Mode	True
Commission	
Default	
Per share	0.0050
Per trade	0.0000
% of trade value	0.00%
Min amount	1.0000
Min % of value	0.00%
Max amount	0.0000
Max % of value	0.20%
Tier 1	
Tier 2	
Tier 3	
Tier 4	
Tier 5	
Statistics	
Initial Equity	100000.00

Max % of value
Maximum commission % of trade value. Z

Commission table allows to enter both per-share (per-contract) commissions and commissions that are expressed as percent of trade value. Or a combination of both. You can also set minimums and maximums expressed in dollar amount and/or percent of trade value. For example if your broker may use 0.01$ (one cent) per share commission, then you would use PerShare = 0.01 and %OfTradeValue = 0. If your broker uses say 0.2% of trade value then you would use PerShare = 0 and %OfTradeValue = 0.2;

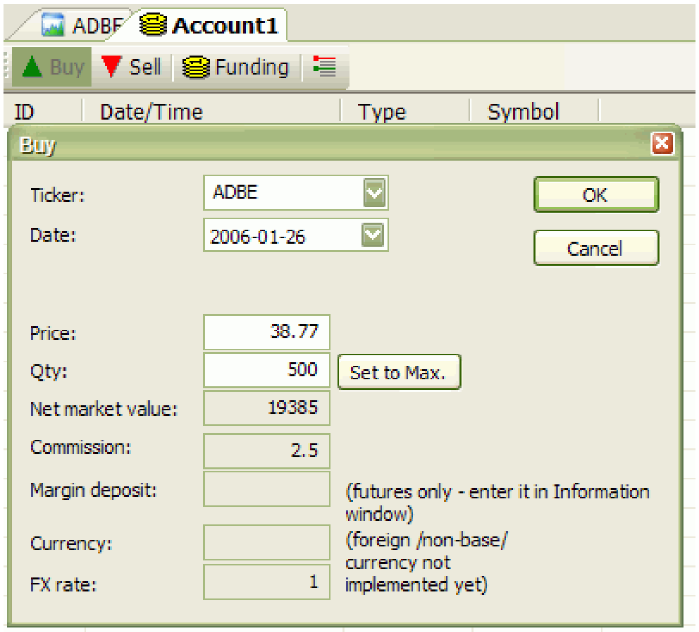
Practical example: Interactive Brokers default commission for U.S. stocks is: 0.005 per share but not less than 1 dollar and not more than 0.2% of trade value. Appropriate settings for such schedule are shown in the screenshot above.

Commission table works as follows: first sum of per-share commission and % of trade value is calculated. Then the result is checked against minimum and maximum limits and if calculated value exceeds the limit then commission is set to value of such the limit, otherwise calculated value is used without change.

Summary page contains a little bit of basic statistics as well.

ENTERING TRADES

Once you funded an account you can enter trades.To buy (enter long position or cover short position) click on "BUY" button.



Ticker:	ADBE
Date:	2006-01-26
Price:	38.77
Qty:	500
Net market value:	19385
Commission:	2.5
Margin deposit:	
Currency:	
FX rate:	1

(futures only - enter it in Information window)
(foreign /non-base/ currency not implemented yet)

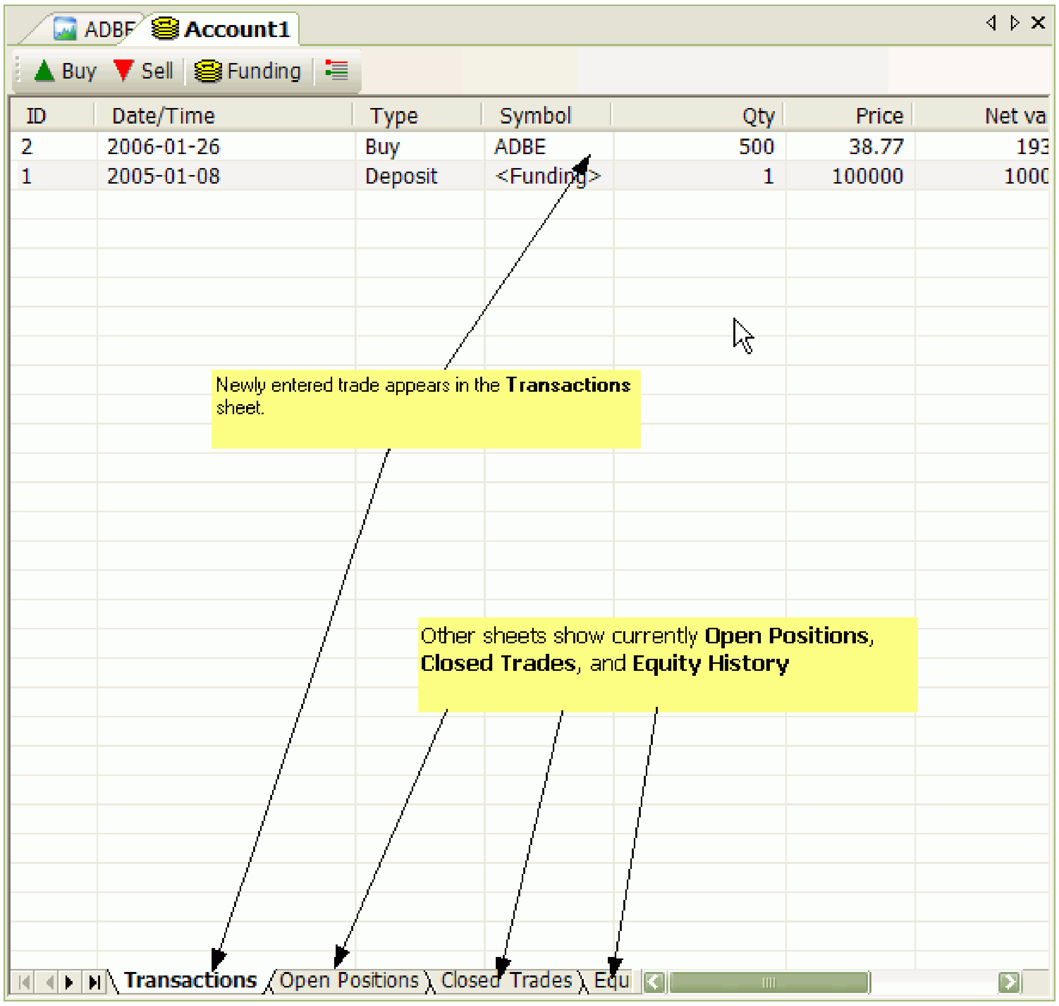
Then in the Buy dialog you need to select the symbol, the trade date/time. Once they are entered AmiBroker will display price of given symbol at the selected date/time (or preceding one if no exact match is found). It will also calculate maximum possible quantity taking price and available funds into account.

You can change the price and quantity manually.

All other values (net market valye, commission, market deposit, currency, fx rate) are calculated or retrieved automatically from Symbol->Information page. Once values are good, click OK to confirm transaction. If you made mistake, you can press UNDO (Edit->Undo) to revert last transaction.

Similar procedure is for selling (entering short positions or closing longs) with the exception that you should press "SELL" button instead.

All transactions that you made are listed in the "Transactions" sheet. All open positions are listed in "Open Positions" sheet. If you enter the trade for symbol that has position already open, AMiBroker will adjust "open positions" accordingly (perform scaling in/out). Once open position is closed it is removed from "open positions" list and moved to "Closed trades" sheet.



ID	Date/Time	Type	Symbol	Qty	Price	Net va
2	2006-01-26	Buy	ADBE	500	38.77	193
1	2005-01-08	Deposit	<Funding>	1	100000	1000

Newly entered trade appears in the **Transactions** sheet.

Other sheets show currently **Open Positions**, **Closed Trades**, and **Equity History**

After each transaction, "Equity history" sheet is updated with current account equity value and also "Summary" page is updated with basic open/long/short trade stats.(More stats are to come).

IMPORTANT

You have to remember that you must enter all transactions in chronological manner(oldest first, newest last), as account manager won't allow you to add trades out-of-order. If you make mistake, there is one-level undo that you can use to revert to state before last transaction. If you made more mistakes, the only option is to close account without saving and re-open original file.

SAVING YOUR ACCOUNT DATA

To save edits made to account use File->Save (or File->Save As to save under new name). Note that **account files are NOT encrypted now**, and it is quite easy to read the file for everyone who has the access to it. So make sure not to leave your files on some public computer. Password protection/encryption is planned but NOT implemented yet.

OPENING PREVIOUSLY CREATED ACCOUNT

To open account file, go to File->Open, in the File dialog, select "Account (*.acx)" from "Files of type" combo-box, and select the account file you want to load.

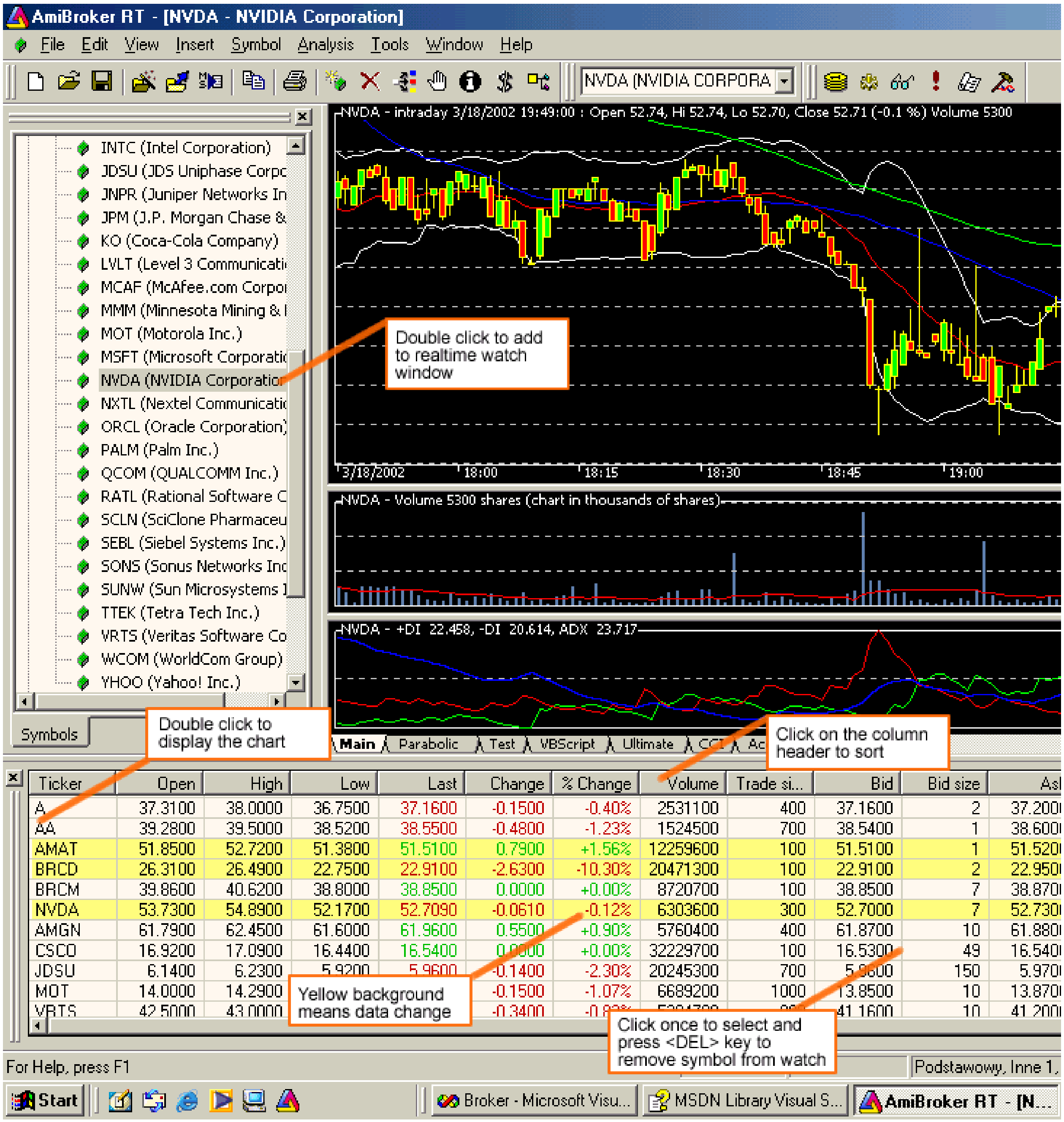
MULTIPLE ACCOUNTS

You can create/open multiple accounts at once (just use File->New->Account, File->Open many times).

Real-time quote window

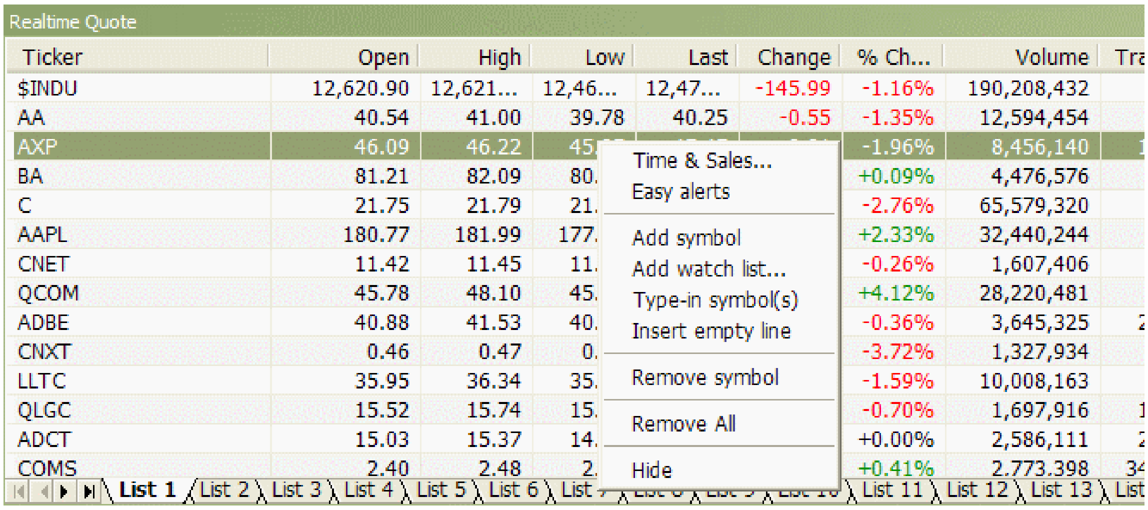
Working with real time quote window

The RT quote window provides real-time streaming quotes and some basic fundamental data. It is fairly easy to operate as shown in the picture below:


AmiBroker RT - [NVDA - NVIDIA Corporation]
File Edit View Insert Symbol Analysis Tools Window Help
NVDA (NVIDIA CORPORA
INTC (Intel Corporation)
JDSU (JDS Uniphase Corpc
JNPR (Juniper Networks In
JPM (J.P. Morgan Chase &
KO (Coca-Cola Company)
LVLT (Level 3 Communicati
MCAF (McAfee.com Corpo
MMM (Minnesota Mining & I
MOT (Motorola Inc.)
MSFT (Microsoft Corporati
NVDA (NVIDIA Corporatio
NXTL (Nextel Communicati
ORCL (Oracle Corporation)
PALM (Palm Inc.)
QCOM (QUALCOMM Inc.)
RATL (Rational Software C
SCLN (SciClone Pharmaceu
SEBL (Siebel Systems Inc.)
SONS (Sonus Networks Inc
SUNW (Sun Microsystems I
TTEK (Tetra Tech Inc.)
VRTS (Veritas Software Co
WCOM (WorldCom Group)
YHOO (Yahoo! Inc.)
Symbols
NVDA - intraday 3/18/2002 19:49:00 : Open 52.74, Hi 52.74, Lo 52.70, Close 52.71 (-0.1 %) Volume 5300
3/18/2002 18:00 18:15 18:30 18:45 19:00
NVDA - Volume 5300 shares (chart in thousands of shares)
NVDA - +DI 22.458, -DI 20.614, ADX 23.717
Main Parabolic Test VBScript Ultimate CCI Ac
Double click to add to realtime watch window
Double click to display the chart
Click on the column header to sort
Yellow background means data change
Click once to select and press <DEL> key to remove symbol from watch
For Help, press F1
Podstawowy, Inne 1,
Start
Broker - Microsoft Visu...
MSDN Library Visual S...
AmiBroker RT - [N...

Ticker	Open	High	Low	Last	Change	% Change	Volume	Trade si...	Bid	Bid size	As
A	37.3100	38.0000	36.7500	37.1600	-0.1500	-0.40%	2531100	400	37.1600	2	37.200
AA	39.2800	39.5000	38.5200	38.5500	-0.4800	-1.23%	1524500	700	38.5400	1	38.600
AMAT	51.8500	52.7200	51.3800	51.5100	0.7900	+1.56%	12259600	100	51.5100	1	51.520
BRCD	26.3100	26.4900	22.7500	22.9100	-2.6300	-10.30%	20471300	100	22.9100	2	22.950
BRCM	39.8600	40.6200	38.8000	38.8500	0.0000	+0.00%	8720700	100	38.8500	7	38.870
NVDA	53.7300	54.8900	52.1700	52.7090	-0.0610	-0.12%	6303600	300	52.7000	7	52.730
AMGN	61.7900	62.4500	61.6000	61.9600	0.5500	+0.90%	5760400	400	61.8700	10	61.880
CSCO	16.9200	17.0900	16.4400	16.5400	0.0000	+0.00%	32229700	100	16.5300	49	16.540
JDSU	6.1400	6.2300	5.9200	5.9600	-0.1400	-2.30%	20245300	700	5.9600	150	5.970
MOT	14.0000	14.2900	[illegible]	[illegible]	-0.1500	-1.07%	6689200	1000	13.8500	10	13.870
VRTS	42.5000	43.0000	[illegible]	[illegible]	-0.3400	[illegible]	[illegible]	[illegible]	41.1600	10	41.200

You can also display context menu by pressing RIGHT mouse button over RT quote window.



Ticker	Open	High	Low	Last	Change	% Ch...	Volume	Tra
$INDU	12,620.90	12,621...	12,46...	12,47...	-145.99	-1.16%	190,208,432	
AA	40.54	41.00	39.78	40.25	-0.55	-1.35%	12,594,454	
AXP	46.09	46.22	45.			-1.96%	8,456,140	1
BA	81.21	82.09	80.			+0.09%	4,476,576	
C	21.75	21.79	21.			-2.76%	65,579,320	
AAPL	180.77	181.99	177.			+2.33%	32,440,244	
CNET	11.42	11.45	11.			-0.26%	1,607,406	
QCOM	45.78	48.10	45.			+4.12%	28,220,481	
ADBE	40.88	41.53	40.			-0.36%	3,645,325	2
CNXT	0.46	0.47	0.			-3.72%	1,327,934	
LLTC	35.95	36.34	35.			-1.59%	10,008,163	
QLGC	15.52	15.74	15.			-0.70%	1,697,916	1
ADCT	15.03	15.37	14.			+0.00%	2,586,111	2
COMS	2.40	2.48	2.			+0.41%	2,773,398	34

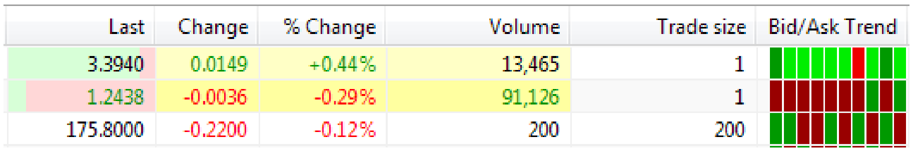
The context menu allows you to access the following options:

- **Time & Sales**
  Opens Time & Sales window that provides information about every bid, ask and trade streaming from the market.
- **Easy Alerts**
  Opens Easy Alerts window that provides way to define realtime alerts executed when bid/ask/last and other fields hit user-defined levels
- **Add Symbol**
  Adds current symbol to Real-Time Quote list
- **Add watch list...**
  Adds entire watch list to real-time quote window
- **Type-in symbols**
  Allows to type the symbols directly as comma-separated list
- **Insert empty line**
  Adds empty (separator) line - useful for grouping symbols
- **Remove Symbol**
  Removes highlighted line (symbol) from the Real-Time Quote list.
- **Remove All**
  Removes all symbols from real-time quote list
- **Hide**
  Hides Real-Time Quote list

Re-arranging symbols using drag-and-drop

Real-time quote window now (v5.10 and up) allows you to re-arrange the list of symbols by drag-and-drop mechanism. Simply click the left mouse button over the symbol, hold it down and drag to desired location then release the button.

Bid/ask trend indicator



Last	Change	% Change	Volume	Trade size	Bid/Ask Trend
3.3940	0.0149	+0.44%	13,465	1	🟩🟩🟩🟩🟩🟩🟥🟩🟩🟩
1.2438	-0.0036	-0.29%	91,126	1	🟥🟥🟥🟥🟥🟥🟥🟩🟥🟩
175.8000	-0.2200	-0.12%	200	200	🟩🟩🟥🟥🟩🟥🟩🟥🟩🟥

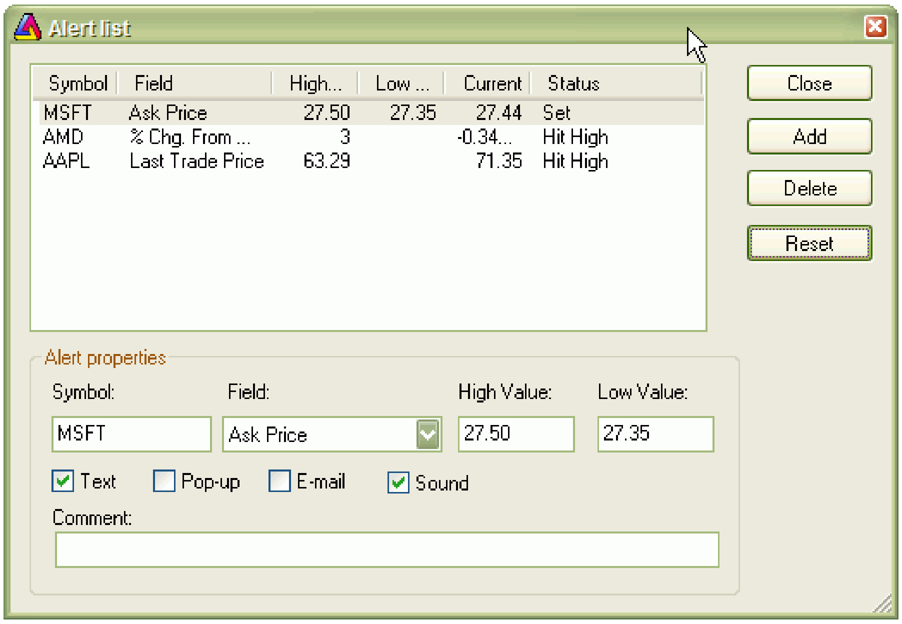
Version 5.90 adds Bid/Ask trend **-** a graphical indicator showing the direction of 10 most recent changes in real-time bid/ask prices. The right-most box is most recent and as new bid/ask quotes arrive they are shifted to the left side. Color coding is as follows:

- **Dark green:** bid > previous bid OR ask > previous ask
- **Bright green:** bid > previous bid AND ask > previous ask
- **Dark red:** bid < previous bid OR ask < previous ask
- **Bright red**: bid < previous bid AND ask < previous ask
- **Red / Green box:** ask < previous ask AND bid > previous bid
- **Green / Red box:** ask > previous ask AND bid < previous bid

If bid/ask prices don't change there is no new box. NOTE: This column works only if there are real-time quotes streaming (markets are open)

Easy alerts window

Easy alert window allows to define real-time alerts without any coding. Please note that this functionality is available ONLY if you are using real-time data plugin and is not available in end-of-day mode.



Symbol	Field	High...	Low ...	Current	Status
MSFT	Ask Price	27.50	27.35	27.44	Set
AMD	% Chg. From ...	3		-0.34...	Hit High
AAPL	Last Trade Price	63.29		71.35	Hit High

Symbol: MSFT Field: Ask Price High Value: 27.50 Low Value: 27.35

[✓] Text [ ] Pop-up [ ] E-mail [✓] Sound

Comment:

Adding new alert

- press **Add** button
- enter ticker symbol into **Symbol** field
- choose price field from **Field** combo box
- enter high trigger value and/or low trigger value
- select at least one of the **Text**/**Pop-up**/**E-mail**/**Sound** fields

Alert will be generated when selected price field (for example Ask) will become equal or greater than High value (if defined), or when selected price field will become equal or less than Low value (if defined). Alert once hit will not re-trigger until you press "**Reset**".

Modifying an alert

Select one of listed alerts and modify values in the edit fields below. If you want to modify an alert that was hit already, after doing modifications please press "**Reset**" button

Deleting alerts

Select one or more alerts from the list (multiple selection possible by pressing down SHIFT key) and then press **Delete** button.

Resetting triggered alerts

The alert that was once hit is market as "Hit high" or "Hit low" in the status field and becomes inactive (won't trigger anymore). If you want to re-activate it, select it from the list and press **Reset** button.

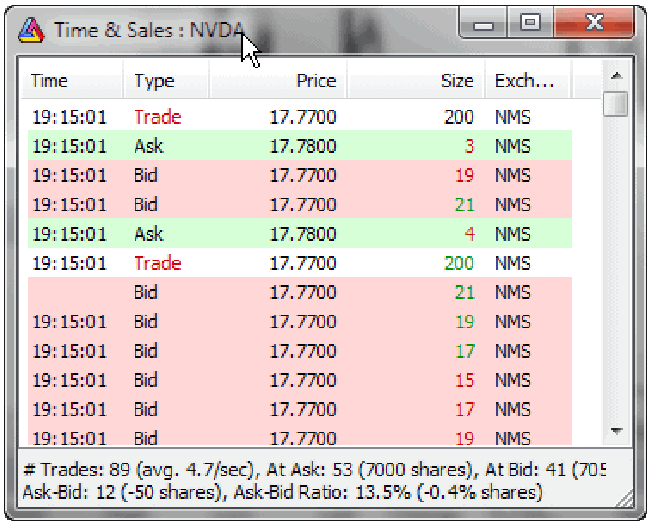
Kinds of alert output

- **Text**
  when this checkbox is marked, when alert is triggered the text defined in comment field will be displayed in Alert Output window (use Window->Alert output menu to display it)
- **Pop-up**
  when this checkbox is marked, triggered alert will display pop-up dialog box
- **E-mail**
  when this checkbox is marked, triggered alert will send an e-mail to the account defined in Preferences/Alerts.
- **Sound**
  when this checkbox is marked, triggered alert will play sound defined in Preferences/Alerts.

Time & Sales window

NOTE: Standard Edition is limited to 1 time & sales window, Professional Edition allows UNLIMITED number of time & sales window open simultaneously.

Time & Sales window that provides information about every bid, ask and trade streaming from the market. Each row displayed represents either new trade, new bid or new ask that is sent by the streaming data source.



Time	Type	Price	Size	Exch...
19:15:01	Trade	17.7700	200	NMS
19:15:01	Ask	17.7800	3	NMS
19:15:01	Bid	17.7700	19	NMS
19:15:01	Bid	17.7700	21	NMS
19:15:01	Ask	17.7800	4	NMS
19:15:01	Trade	17.7700	200	NMS
	Bid	17.7700	21	NMS
19:15:01	Bid	17.7700	19	NMS
19:15:01	Bid	17.7700	17	NMS
19:15:01	Bid	17.7700	15	NMS
19:15:01	Bid	17.7700	17	NMS
19:15:01	Bid	17.7700	19	NMS

```
# Trades: 89 (avg. 4.7/sec), At Ask: 53 (7000 shares), At Bid: 41 (705
Ask-Bid: 12 (-50 shares), Ask-Bid Ratio: 13.5% (-0.4% shares)
```

Each line in time and sales window is marked with color to make it easier to distinguish between various conditions.

Coloring rules are:

- light green background means NEW ASK
- light red background means NEW BID
- normal (white) background means NEW TRADE
- Red text for bid/ask price/size means that the value is LESS than previous value of the same category (for example bid price written in red letters mean that the new BID is lower than previous bid price, green volume field means that the volume of last trade or ask/bid size is greater than last trade volume or ask/bid size)
- Red last trade price means trade occuring on or below current bid
- Green text for bid/ask price/size means that the value is GREATER then previous value.
- Green last trade price means trade occuring on or above current ask.
- Black text for bid/ask price/size/volume means that the value is the same
- Black last trade price means trade occuring inside current bid-ask range (greater than bid and less than ask)

Time&Sales window in version 5.30 shows some "recent statistics" regarding trading namely:

- number of trades and average # of trades per second
- number of trades and shares traded at ask or above
- number of trades and shares traded at bid or below
- ask minus bid difference expressed in number of trades and shares
- ask minus bid difference expressed as percentage ratio to total trades/total volume traded

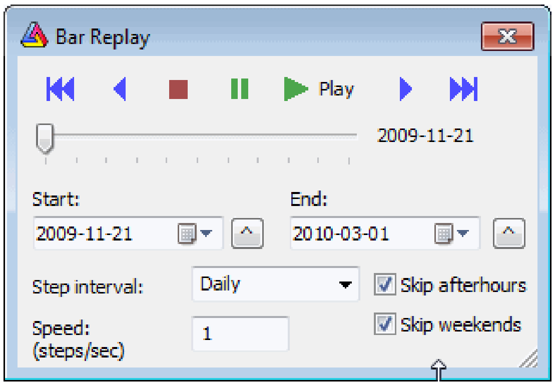
A little background:
Ask minus bid: the positive numbers represent more transactions occuring on ASK side than on BID side. This in theory may mean more buying than selling, but in practice things are largely dependent on security traded. Esp. dark liquidity pools do not show in order books and may report trades to the tape several seconds later thus invalidating relationship between bid/ask and actual trade prices.

IMPORTANT:
These are temporary, short-term stats - they cover ONLY trades displayed in the T&S window since opening of the window OR
resetting stats.

You can reset statistics using right click menu : "Reset Stats"

Bar Replay window

Bar Replay tool is available from **Tools->Bar Replay** menu. Bar Replay feature **plays back data for all symbols at once** with user-defined speed. It means that data for all symbols will end at specified "playback position". This affects all formulas (no matter if they are used in charts (indicators) or auto-analysis).


Bar Replay

Play

2009-11-21

Start: 2009-11-21

End: 2010-03-01

Step interval: Daily

[✓] Skip afterhours

[✓] Skip weekends

Speed: (steps/sec) 1

Controls description

Navigation bar:

- - **Rewind** to the beginning
- - one **Step Back**
- - **Stop** - turns bar replay OFF (chart are not affected by bar reply)
- - **Pause** - pauses current playback or enters pause mode that allows to manually drag the slider bar and affect chart display - in PAUSE mode data are internally modified so quotes past selected "playback" position are invisible to any part of AmiBroker (charts / automatic analysis), except quotation editor
- Play - **Play** - playback bars history
- - one **Step Forward**
- - **Forward** to end of selected range

Slider bar - allows to see the playback progress as well as MANUALLY move back and forward by dragging the slider thumb.

Start/End - controls provide start and end simulation dates. The playback works so that all data upto currently selected "Playback position" are visible. Data past this position are invisible. "Playback position" can change from user-defined "Start" to "End"dates. The small ^ buttons on the right side of Start / End date fields allow to set Start/End to currently selected date on the chart.

Step interval - defines interval of the step. Recommended setting is base interval of your database. So if you have 1-minute database, step interval should be 1 minute. If you have EOD database, step interval should be daily, however it is allowed to select higher step intervals. Note that chart viewing interval is independent from that. So you can playback 1 minute database and watch 15 minute bars (they will look like real - building last "ghost" bar as new data come in)

Speed parameter defines step frequency. It means how many steps will be played back within one second. Default is 1. Maximum is 5 minimum is 0.1. If you select 3 for example, AmiBroker will play one step every 0.333 sec giving total of 3 steps per second.

Skip afterhours - when turned on, playback skips hours outside regular trading hours as defined in File->Database Settings->Intraday Settings

Skip weekends - when turned on, playback skips Saturdays and Sundays

Usage

To ENTER Playback mode - press PLAY or PAUSE buttons - then data are truncated at current "playback position".

To EXIT Playback mode - press STOP button or close Bar Replay dialog - the full data set will be restored.

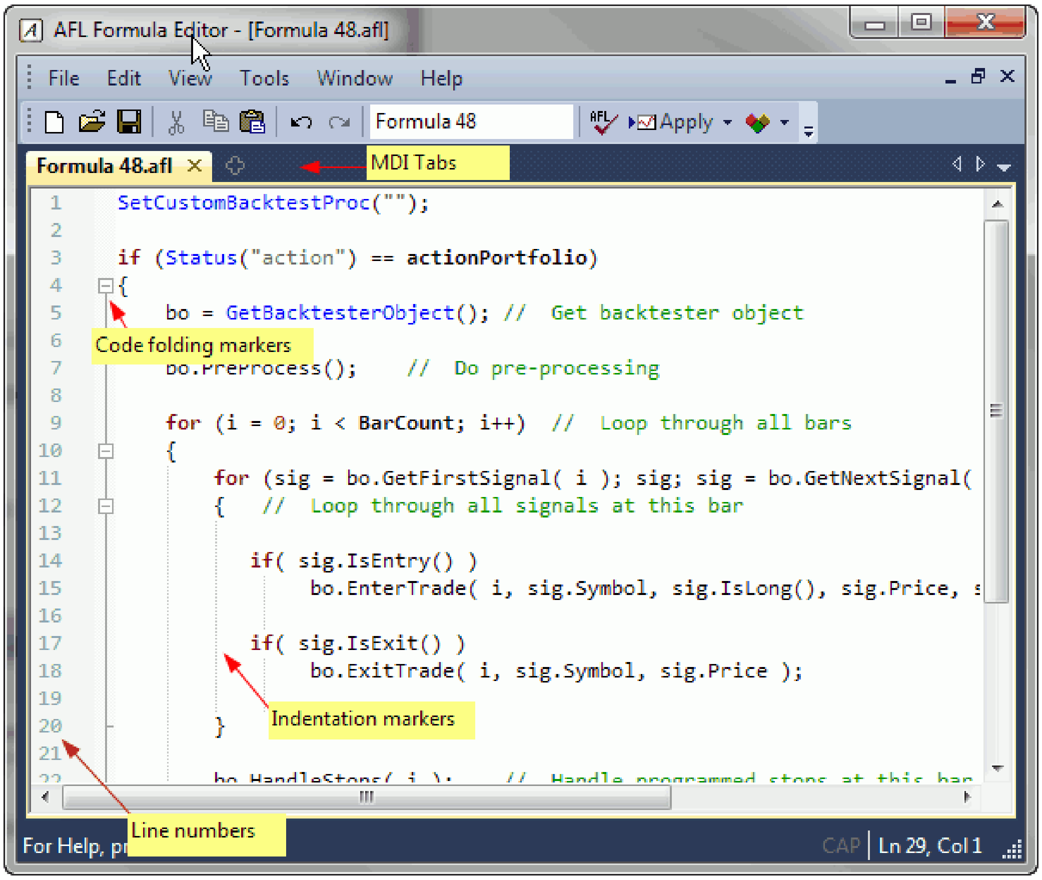
Note that playback simulation is done internally and the database is kept untouched in fact (all data are still visible in Quote Editor), so there is no risk using Bar Reply.

Formula Editor

A new AFL Formula Editor features:

- Syntax highlighting (improved in 5.80)
- Automatic brace matching/highlighting (NEW in 5.80)
- Auto indentation (NEW in 5.80)
- Indentation markers (NEW in 5.80)
- Enhanced auto-complete in two modes (immediate (NEW in 5.80) and on-demand)
- Parameter information tooltips
- Line numbering margin and selection margin (NEW in 5.80)
- Code folding (NEW in 5.80)
- In-line Error Reporting (NEW in 5.80)
- New tabbed user interface with ability to work in both MDI and separate floating frame mode, can be moved behind main AmiBroker screen and brought back (Window->Toggle Frame) (NEW in 5.80) or kept on top (Window->Keep on top)
- Rectangular block copy/paste/delete (Use mouse and hold down left **Alt** key to mark rectangular block) (new in 5.80)
- Auto capitalisation (change case)
- Virtual space (new in 5.80)
- Enhanced printing (with syntax highlighting and header/footer)
- Code snippets (new in 5.80)
- Integrated Visual Debugger (NEW in 6.10)
- Bookmarks (NEW in 6.10)
- Find-in-Files (NEW in 6.10)
- Block comment/uncomment (NEW in 6.10)
- Clickable links in comments (NEW in 6.10)

These features greatly simplifies writing formula and provides instant help so time needed to write formula decreases significantly.

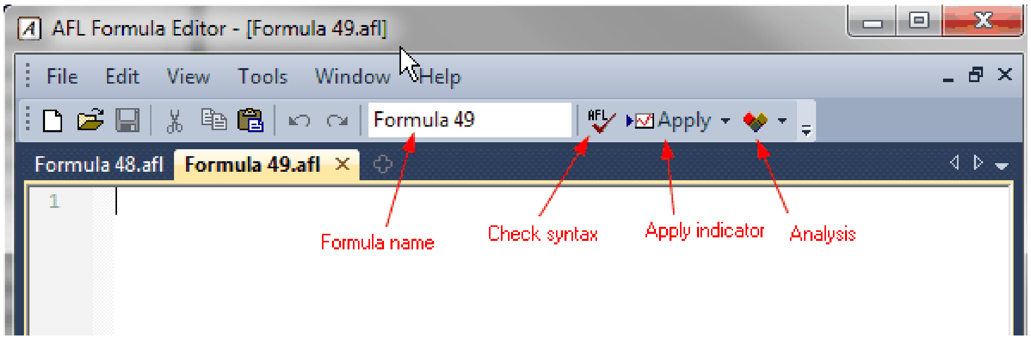


```
SetCustomBacktestProc("");

if (Status("action") == actionPortfolio)
{
    bo = GetBacktesterObject(); //  Get backtester object

    bo.PreProcess();    //  Do pre-processing

    for (i = 0; i < BarCount; i++)  //  Loop through all bars
    {
        for (sig = bo.GetFirstSignal( i ); sig; sig = bo.GetNextSignal(
        {   //  Loop through all signals at this bar

            if( sig.IsEntry() )
                bo.EnterTrade( i, sig.Symbol, sig.IsLong(), sig.Price, s

            if( sig.IsExit() )
                bo.ExitTrade( i, sig.Symbol, sig.Price );

        }

        bo.HandleStops( i );    //  Handle programmed stops at this bar
```

Menu

Formula Editor menu options are described in detail in Menus: Formula Editor chapter of the guide.

Toolbar



The Formula Editor toolbar provides the following buttons:

- **New** - clears the formula editor window
- **Open** - opens the formula file
- **Save** - saves the formula under current name
- **Print** - prints the formula
- **Cut** - cuts the selection and copies to the clipboard
- **Copy** - copies the selection to the clipboard
- **Paste** - pastes current clipboard content in the current cursor position

- **Undo** - un-does recent action (multiple-level)
- **Redo** - re-does recent action (multiple-level)
- **Formula Name** - an EDIT field that allows to modify the formula file name, once you change the name here and press **Save** button the formula will be saved under new name and the change will be refleced in editor CAPTION BAR and in the STATUS BAR (Status bar shows full path).
- **Check syntax** - checks current formula for errors
- **Apply indicator** - saves the formula and applies current formula as a chart/indicator ONCE
- **Analysis** - saves the formula and selects it as current formula in Automatic Analysis window and repeat most recently used Analysis operation (i.e. Scan or Exploration or Backtest or Optimization)

Usage

Typical use of Formula Editor is as follows:

- open Formula Editor
- type the formula
- type meaningful name that describes the purpose of you code into **Formula Name** field
- click **Apply indicator** button (if you have written indicator code)
  .. or..
  click **Analysis** button to display Automatic Analysis window (when you have written exploration/scan or trading system)

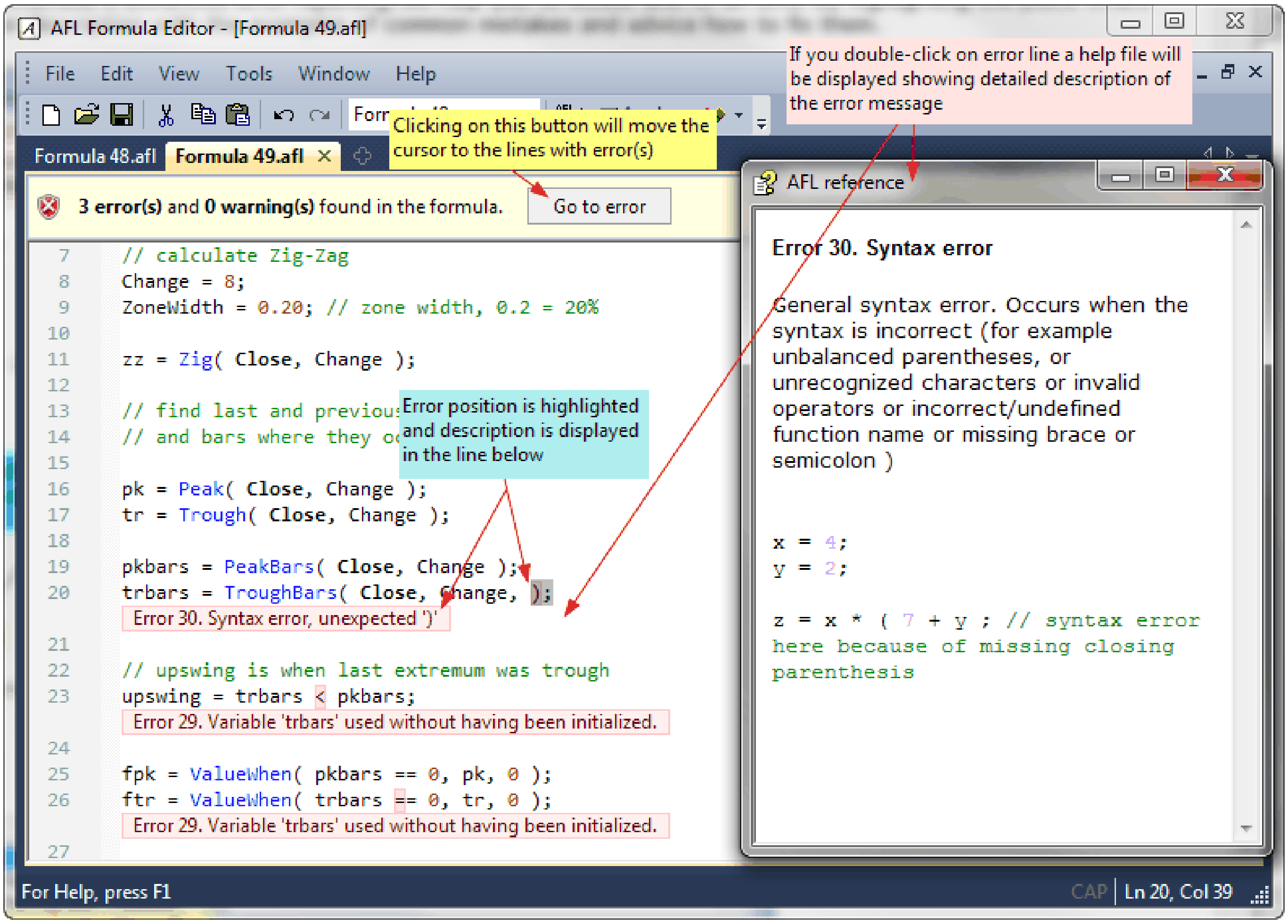
Syntax highlighting

AmiBroker's AFL editor features user-definable syntax highlighting that automatically applies user-defined colors and styles to different language elements like functions and reserved variable names, strings, numbers, comments, etc. This feature greatly simplifies code writing. You can modify coloring scheme in Preferences window.

Enhanced error reporting

When you make an error in your formula, AmiBroker's enhanced error reporting will help you to locate and fix an error by highlighting the place where error occured and displaying extended error description with the examples of common mistakes and advice how to fix them. In version 5.80 description of errors are displayed in-line with the code.

A message bar displays total number of errors and/or warnings. If you press "**Go to error**" button the editor will move the caret to the relevant line with the error, if you press it again, it will move to the next error and so on. If you close the message bar with the "X" button all error messages will be cleared (hidden) from the view. You can use **Edit->Clear Error Message** menu (**Ctrl+E**) to clear individual error message (in the current line).


AFL Formula Editor - [Formula 49.afl]
File Edit View Tools Window Help
Formula 48.afl Formula 49.afl
Go to error
AFL reference
For Help, press F1
CAP Ln 20, Col 39

If you double-click on error line a help file will be displayed showing detailed description of the error message

Clicking on this button will move the cursor to the lines with error(s)

3 error(s) and 0 warning(s) found in the formula.

Error position is highlighted and description is displayed in the line below

```
// calculate Zig-Zag
Change = 8;
ZoneWidth = 0.20; // zone width, 0.2 = 20%

zz = Zig( Close, Change );

// find last and previous
// and bars where they oc

pk = Peak( Close, Change );
tr = Trough( Close, Change );

pkbars = PeakBars( Close, Change );
trbars = TroughBars( Close, Change, );
Error 30. Syntax error, unexpected ')'

// upswing is when last extremum was trough
upswing = trbars < pkbars;
Error 29. Variable 'trbars' used without having been initialized.

fpk = ValueWhen( pkbars == 0, pk, 0 );
ftr = ValueWhen( trbars == 0, tr, 0 );
Error 29. Variable 'trbars' used without having been initialized.
```

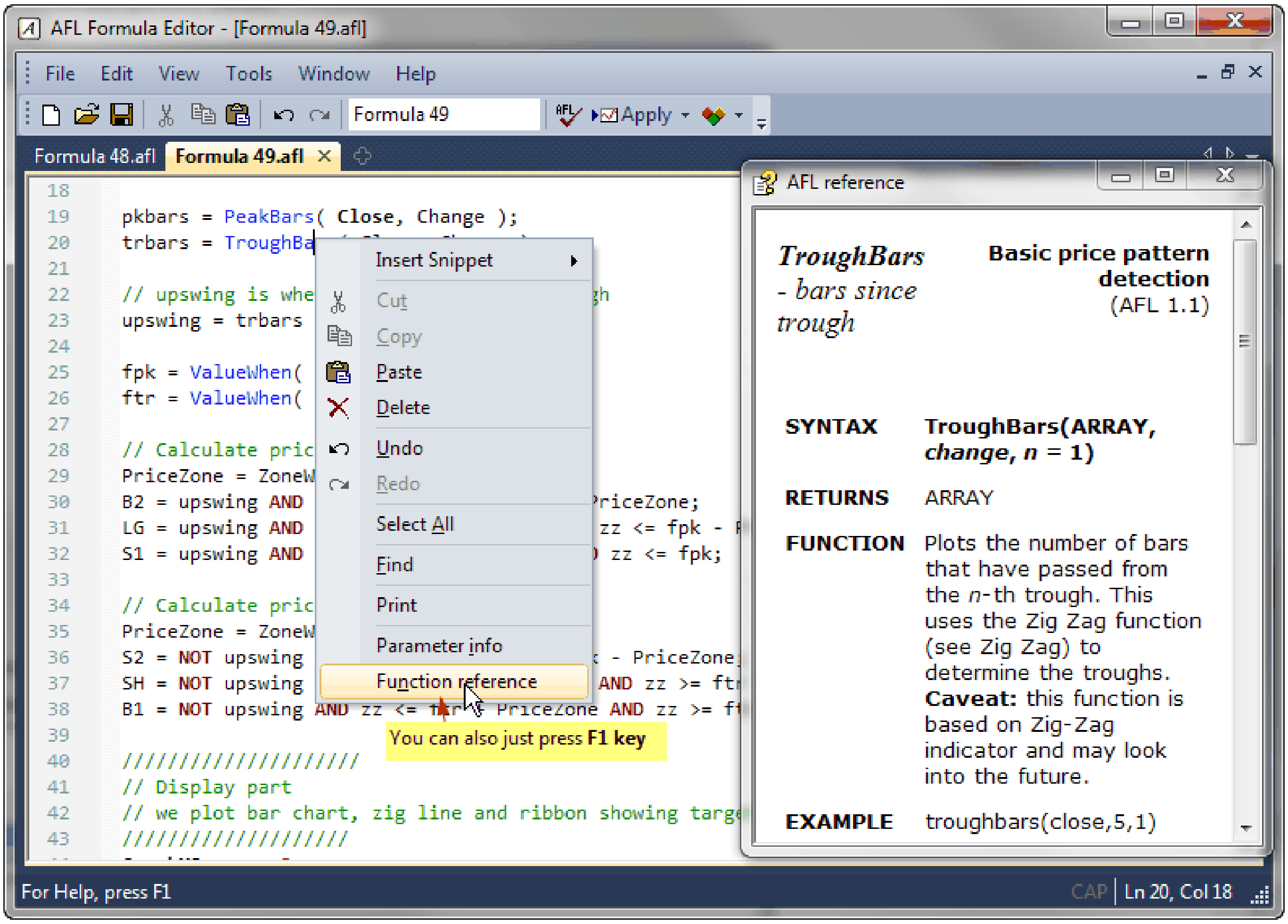
Error 30. Syntax error

General syntax error. Occurs when the syntax is incorrect (for example unbalanced parentheses, or unrecognized characters or invalid operators or incorrect/undefined function name or missing brace or semicolon)

```
x = 4;
y = 2;

z = x * ( 7 + y ; // syntax error
here because of missing closing
parenthesis
```

Context help

You can quickly display relevant AFL function reference page if you press **F1** key or choose "Function reference" from the context menu while the caret is inside or right after function name as shown in the picture below:



```
pkbars = PeakBars( Close, Change );
trbars = TroughBa

// upswing is whe                                gh
upswing = trbars

fpk = ValueWhen(
ftr = ValueWhen(

// Calculate pric
PriceZone = ZoneW
B2 = upswing AND                                 riceZone;
LG = upswing AND                                 zz <= fpk - P
S1 = upswing AND                                 ) zz <= fpk;

// Calculate pric
PriceZone = ZoneW
S2 = NOT upswing                                 - PriceZone;
SH = NOT upswing                                 AND zz >= ftr
B1 = NOT upswing AND zz <= ftr + PriceZone AND zz >= ft

/////////////////////
// Display part
// we plot bar chart, zig line and ribbon showing targe
////////////////////
```

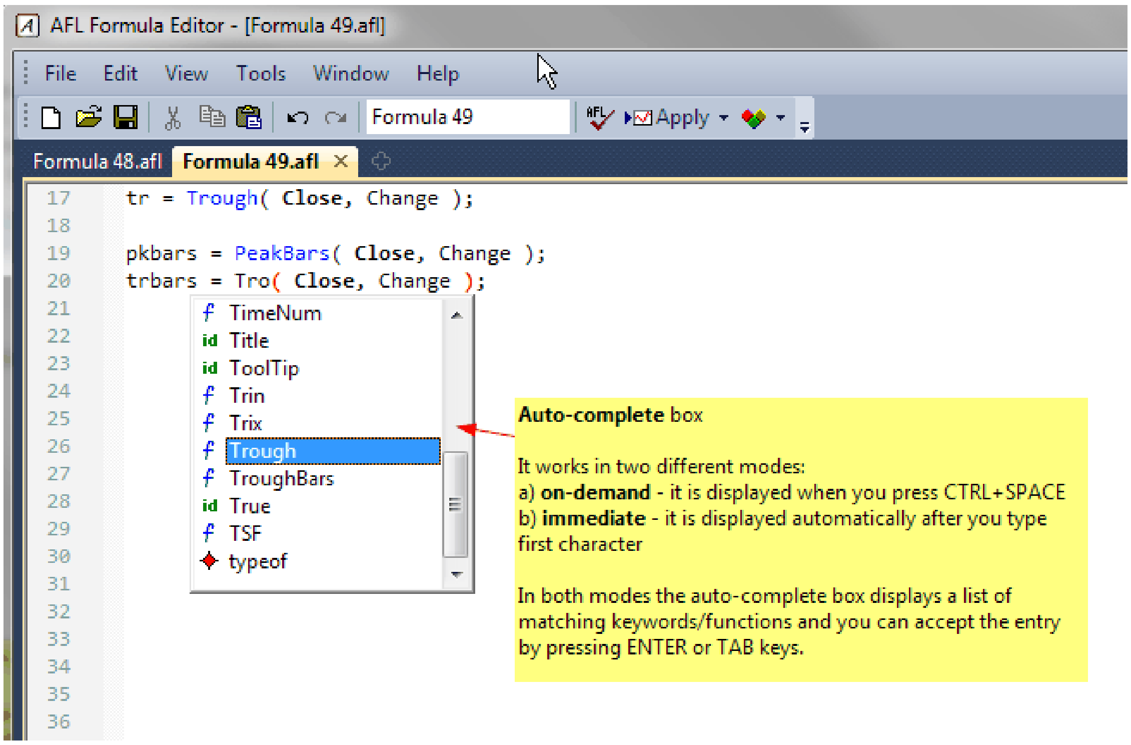
TroughBars - bars since trough

Basic price pattern detection (AFL 1.1)

SYNTAX	**TroughBars(ARRAY, *change*, *n* = 1)**
RETURNS	ARRAY
FUNCTION	Plots the number of bars that have passed from the *n*-th trough. This uses the Zig Zag function (see Zig Zag) to determine the troughs. **Caveat:** this function is based on Zig-Zag indicator and may look into the future.
EXAMPLE	troughbars(close,5,1)

Automatic statement completion

The automatic completion feature (available when you press **CTRL+SPACE** key combination) finishes typing your functions and reserved variables for you, or displays a list of candidates if what you've typed has more than one possible match. You can select the item from the list using up/down arrow keys or your mouse. To accept selection press RETURN (ENTER). You can also type immediately space (for variables) or opening brace (for function) and AmiBroker will auto-complete currently selected word and close the list. To dismiss the list press ESC key.



```
tr = Trough( Close, Change );

pkbars = PeakBars( Close, Change );
trbars = Tro( Close, Change );
```

Auto-complete box

It works in two different modes:
a) **on-demand** - it is displayed when you press CTRL+SPACE
b) **immediate** - it is displayed automatically after you type first character

In both modes the auto-complete box displays a list of matching keywords/functions and you can accept the entry by pressing ENTER or TAB keys.

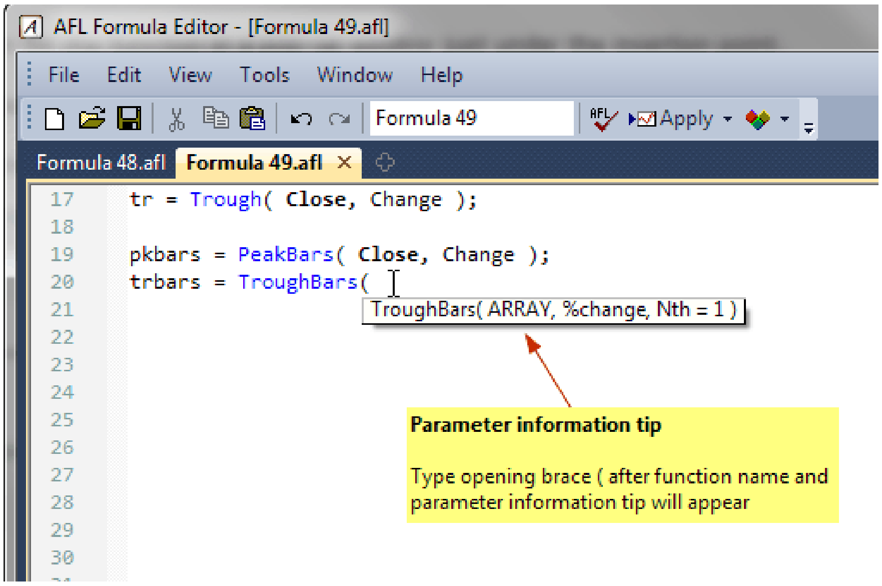
Parameter Information

When you are typing a function, you can display a Tool Tip containing the complete function prototype, including parameters. The **Parameter Info** Tool Tip is also displayed for nested functions.
With your insertion point next to a function, type an open parenthesis as you normally would to enclose the parameter list.

AmiBroker displays the complete declaration for the function in a pop-up window just under the insertion point.

Typing the closing parenthesis dismisses the parameter list.

You can also dismiss the list if you press arrow up/down key, click with the mouse or press RETURN.


AFL Formula Editor - [Formula 49.afl]
File Edit View Tools Window Help
Formula 49
Apply
Formula 48.afl
Formula 49.afl

```
tr = Trough( Close, Change );

pkbars = PeakBars( Close, Change );
trbars = TroughBars(
```

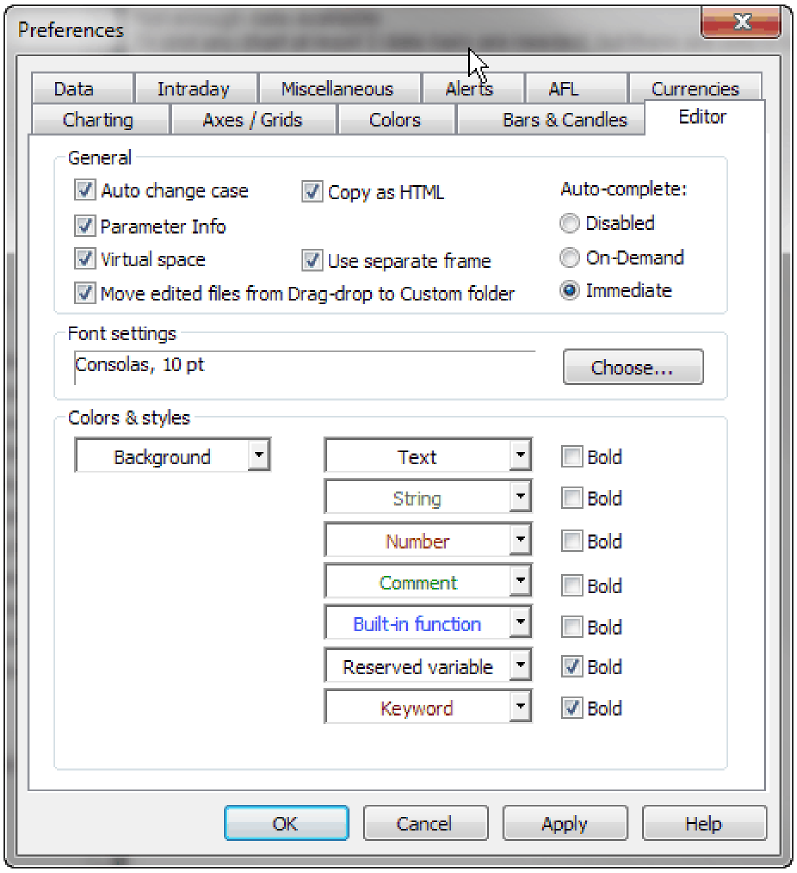
TroughBars(ARRAY, %change, Nth = 1)

Parameter information tip

Type opening brace (after function name and parameter information tip will appear

Editor configuration

The settings of the AFL editor can be changed using **Tools->Preferences**, **Editor** page:



General
- [✓] Auto change case
- [✓] Copy as HTML
- [✓] Parameter Info
- [✓] Virtual space
- [✓] Use separate frame
- [✓] Move edited files from Drag-drop to Custom folder

Auto-complete: Disabled / On-Demand / [✓ Immediate]

Font settings: Consolas, 10 pt

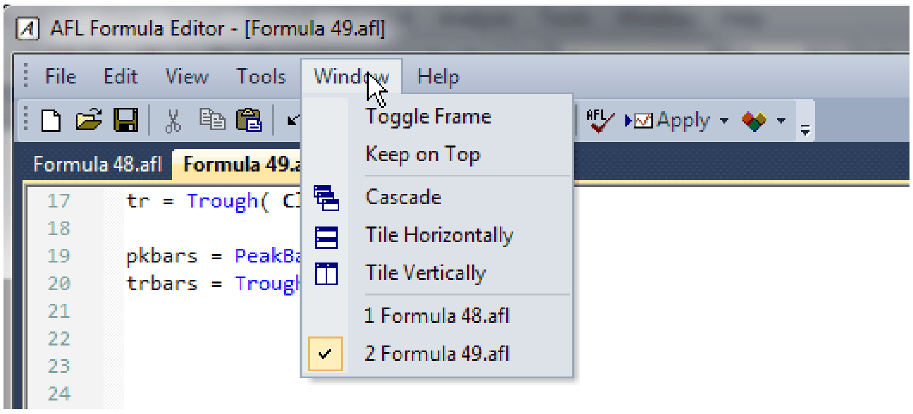
Colors & styles: Background

Text	Bold [ ]
String	Bold [ ]
Number	Bold [ ]
Comment	Bold [ ]
Built-in function	Bold [ ]
Reserved variable	Bold [✓]
Keyword	Bold [✓]

- Auto change case - controls whenever editor automatically changes case of reserved keywords (for example if user typed valuewhen it would change it to ValueWhen)
- Parameter info - controls whenever parameter info tips are displayed

- Virtual space - controls whenever it is possible to place the caret freely in any place after the end of a line
- Move edited files from drag-drop to custom folder - normally formulas created by drag-drop mechanism are located in hidden drag-drop folder, if you then want to edit them, you can do so in place so they remain in drag drop (hidden) folder, or you may choose to move them automatically to 'custom' folder. This switch enables automatic move to custom folder
- Copy as HTML - enables copies in HTML format so AFL code is copied with colors, without it it will be copied as plain text without formatting
- Use separate frame - if turned on it displays AFL Editor in completely separate frame that behaves like separate application, if it is turned off, then AFL editor is displayed as a MDI tab within main AmiBroker frame (along with charts, analysis windows, web, account windows and so on). By default it is turned on
- Auto-complete: in "On-demand" mode auto-complete list shows up only when you press **Ctrl+SPACE**, in "Immediate" mode auto-complete list pops up automatically as soon as you type first character (letter) of the identifier.

Window control

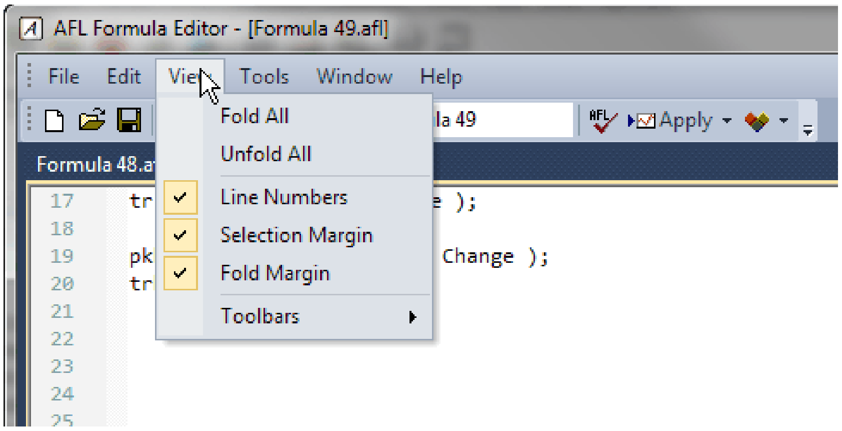


```
tr = Trough( C
pkbars = PeakB
trbars = Troug
```

AFL Editor Window as a separate frame can be brought on top or to the back as any other application window using Windows Task Bar. In addition to that there is a **Window->Toggle Frame** menu (and **Ctrl+`** shortcut, ` is the tilde key just above TAB key on most keyboards) that allows to quickly toggle between AmiBroker main frame and AFL editor frame.

The user may also turn on **Window->Keep On Top** feature that keeps editor window on top of AmiBroker main frame.

Margins

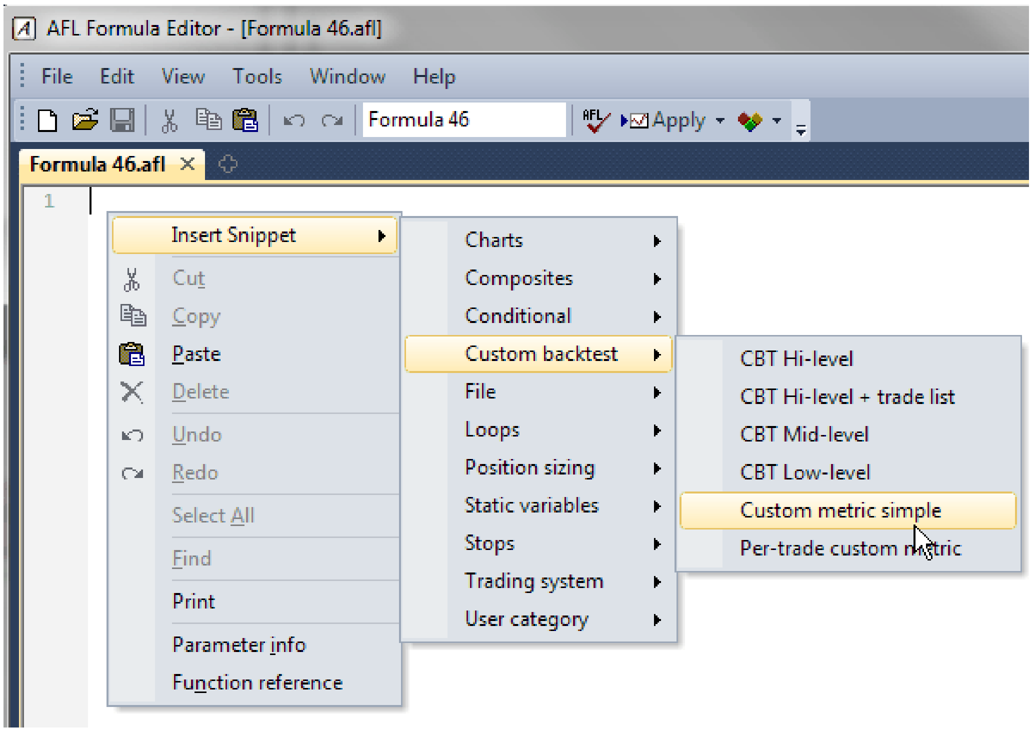


Line numbers margin, **Selection margin** and **Fold margin** can be switched on/off using **View** menu. In this menu there are also options to fold/unfold all code.

Code snippets

Code snippet is a small piece of AFL code. It can be inserted by:

- right-clicking in the AFL editor window and choosing "Insert Snippet" menu, or
- dragging a snippet from Code Snippet window, or
- typing keyboard trigger (such as @for) in the editor



For more information about Code snippets see Tutorial: Using Code-snippets

Bookmarks

A bookmark is a marker that allows to quickly jump to marked line within the formula.

To set/remove the bookmark use **Edit->Bookmark->Insert /Remove** menu or **Ctrl+F2** key. When you set the bookmark a light blue marker will be shown in the left-hand margin. Now with a couple of bookmarks set you can quickly navigate (jump) betwen them using **F2** key - goes to next bookmark, **Shift+F2 key** - goes to previous bookmark. You can clear all bookmarks using **Edit->Bookmark->Clear All.** Please note that bookmarks are saved along with debug information (breakpoints/watches) in the separate file with .dbg extension.

Find in Files

Find in Files functionality is available from **Edit->Find in Files** menu and allows you to search all files in selected folders for specified text. The search results are presented in the Output window (**Window->Output** menu) like this:

```
Filepath\Filename.afl(line_number): Line contents
```

Now if you double click on that output line AmiBroker would automatically open given file in the new editor tab.

Block comment/uncomment

The AFL editor now provides a tool to automatically comment out blocks of selected lines. The tool adds a // (double slash) comment to every line within selected area. To use this tool, select some lines using mouse or cursor + SHIFT key, then use **Edit->Line Comment.** All lines within selected range will be commented out with // (double slash). To uncommment, select the range again and choose **Edit->Line Comment** once again.

Clickable links in comments

The AFL Editor now supports clickable JavaDoc/Doxygen-style links in doc comments. To use links in comments you have to put @link command followed by path to file or URL inside JavaDoc/Doxygen style comment:

- multiline comment that begins with /** like this:

  ```
  /**
  @link readme.html
  @link http://www.amibroker.com
  */
  ```

  (note double asterisk after initial slash)
- single line comment that begins with triple slash

  ```
  /// @link readme.html
  /// @link c:\program files\amibroker\readme.html
  /// @link http://www.amibroker.com
  ```

Now when you hover the mouse over @link command you will see the underline that indicates it is clickable. It reacts to double click (not single click). When you double click it linked document will be open.

@link command can open web pages, local files (both relative and absolute paths are supported) with Windows-registered program to open given file type. So if you use:

```
/// @link something.doc
```

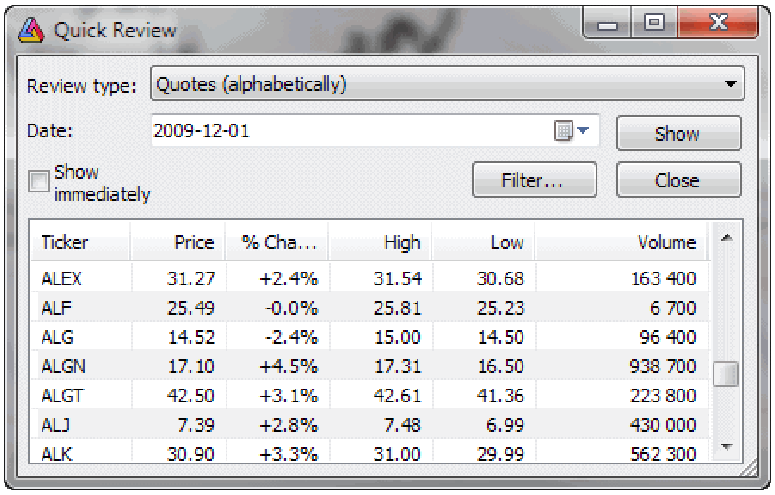
then MS word would be used.

```
/// @link test.xls
```

would open test.xls in Excel.

Relative paths refer to AmiBroker working directory. Html files are open with default browser, txt files are usually open with Notepad (or whatever application you use). If file does not exist then you will get an error message.

Quick review window



Ticker	Price	% Cha...	High	Low	Volume
ALEX	31.27	+2.4%	31.54	30.68	163 400
ALF	25.49	-0.0%	25.81	25.23	6 700
ALG	14.52	-2.4%	15.00	14.50	96 400
ALGN	17.10	+4.5%	17.31	16.50	938 700
ALGT	42.50	+3.1%	42.61	41.36	223 800
ALJ	7.39	+2.8%	7.48	6.99	430 000
ALK	30.90	+3.3%	31.00	29.99	562 300

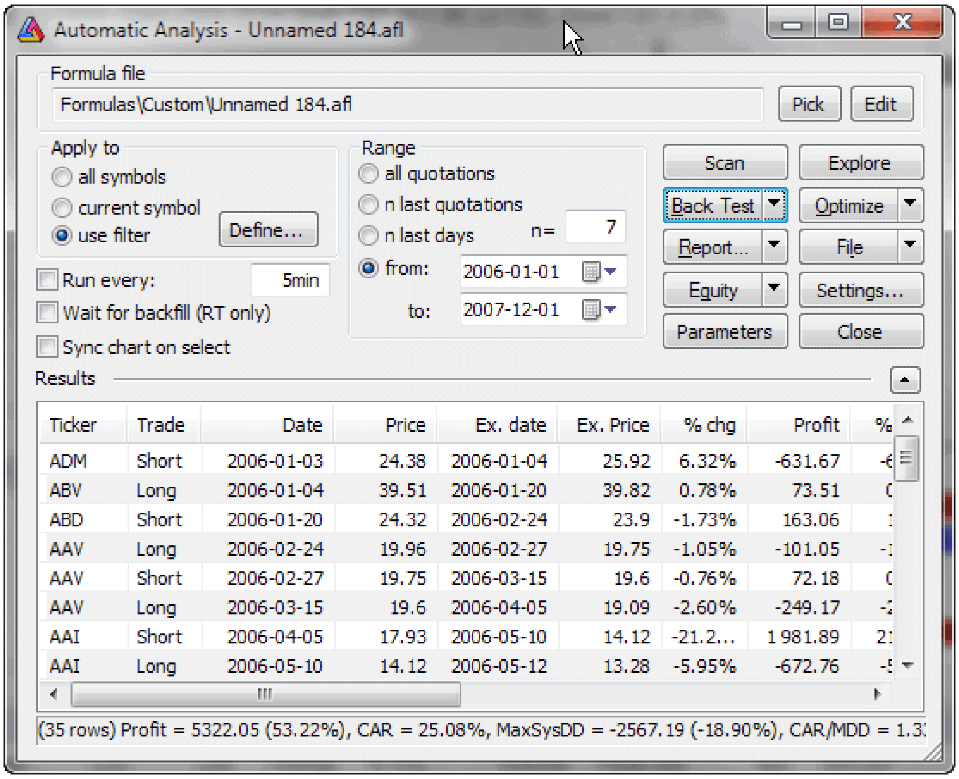
This window provides overall market information like:

- daily symbol quotes
- weekly returns comparison table
- monthly returns comparison table
- quarterly returns comparison table
- yearly returns comparison table
- Price/Earnings comparison
- Price/Book value comparison

In the **Date** field you select the base date for comparisons. For example weekly returns are calculated by dividing base day close price with the closing price one week before.

Filter button allows you to narrow down your search to symbols defined in Filter settings window.

Automatic analysis window



Formula file: Formulas\Custom\Unnamed 184.afl

Apply to: all symbols [ ] current symbol [ ] [✓ use filter]

Run every: [ ] 5min
Wait for backfill (RT only) [ ]
Sync chart on select [ ]

Range: all quotations [ ] n last quotations [ ] n last days [ ] n= 7 [✓ from:] 2006-01-01 to: 2007-12-01

Results

Ticker	Trade	Date	Price	Ex. date	Ex. Price	% chg	Profit	%
ADM	Short	2006-01-03	24.38	2006-01-04	25.92	6.32%	-631.67	-6
ABV	Long	2006-01-04	39.51	2006-01-20	39.82	0.78%	73.51	0
ABD	Short	2006-01-20	24.32	2006-02-24	23.9	-1.73%	163.06	1
AAV	Long	2006-02-24	19.96	2006-02-27	19.75	-1.05%	-101.05	-1
AAV	Short	2006-02-27	19.75	2006-03-15	19.6	-0.76%	72.18	0
AAV	Long	2006-03-15	19.6	2006-04-05	19.09	-2.60%	-249.17	-2
AAI	Short	2006-04-05	17.93	2006-05-10	14.12	-21.2...	1 981.89	21
AAI	Long	2006-05-10	14.12	2006-05-12	13.28	-5.95%	-672.76	-5

(35 rows) Profit = 5322.05 (53.22%), CAR = 25.08%, MaxSysDD = -2567.19 (-18.90%), CAR/MDD = 1.3

Automatic analysis window enables you to check your quotations against defined buy/sell rules. AmiBroker can produce report telling you if buy/sell signals occurred on given symbol in the specified period of time. It can also simulate trading, giving you an idea about performance of your system.

In the upper part of window you can see the path to the formula used along with **Pick** and **Edit** buttons.

Pick button opens up a file dialog that allows you to choose the formula you want to use for the analysis.

Edit button opens up the AFL Formula Editor that allows you to edit currently selected formula.

If you want to create new formula just open Formula Editor directly from **Tools->Formula Editor** menu, type the formula and press **Analysis** button in the Formula Editor toolbar.

In the formula editor you need to write the code that specifies either scan/exploration you want to run or a trading system you want to back test. You can find the description of this language in AFL reference guide.

In order to make things work you should write two assignment statements (one for buy rule, second for the sell rule), for example:

```
buy = cross( macd(), 0 );
sell = cross( 0, macd() );
```

Below these fields there are several controls for setting:

1. To which symbol(s) analysis should be applied.
   You can select here all symbols, only currently selected symbol (selected in Select Window) or

custom filter setting
2. Time range of analysis
   analysis can be applied to all available quotations or only to the defined number of most recent quotations (or days) or to a date range (from/to)

In the lower part of the analysis window you can see 4 buttons:

1. **Scan**
   this starts the signal scan mode - AmiBroker will search through defined range of symbols and quotations for buy/sell signals defined by your trading rules.If one of the buy/sell conditions is fulfilled, AmiBroker will display a line describing when and on which symbol the signal has occurred. Next AmiBroker proceeds to the end of the range so multiple signals on single symbol may be generated.
2. **Explore**
   this starts an exploration mode when AmiBroker scans through database to find symbols that match user-defined filter. The user can define output columns that show any kind of information required. For more information please check out "Tutorial: How to create your own exploration"
3. **Back Test**
   this starts the back-testing mode - AmiBroker will search through defined range of symbols and quotation for BUY signal defined by your buy rule. If the buy rule is fulfilled, AmiBroker will "buy" currently analyzed shares. Next it will search for SELL signal. Then, if sell rule is fulfilled, AmiBroker will "sell" previously bought symbols. At the same time it will display the information about this trading in the listview. After performing simulation the summary will be displayed. Read more in "Tutorial: How to backtest your trading system"...
   The back testing parameters could be changed using Settings window.
4. Settings - allows you to change back tester settings
5. **Optimize** - allows you to optimize your trading system. Read more in the "Tutorial: How to optimize your trading system"...
6. **Check** - this option allows you to check if your formula references future quotes. AmiBroker analyses your formula and detects if it uses quotes past current bar. Please note that formulas referencing future can give unrealistic backtesting results that can not be reproduced in real trading, therefore you should avoid systems that reference future.
7. **Report**
   this displays Report window that allows you to watch, print and save test results
8. **Equity**
   - available only after backtesting - displays Equity curve for currently selected symbol in a new chart pane. Check out "AFL: Equity chart and function".
9. **Export** - allows you to export the results to CSV (comma separated values) file
10. **Close**
    this closes the analysis window

Moreover you two options "Load" and "Save" for loading and saving your trading rules from/to files.

Enlarging results view in Automatic analysis window

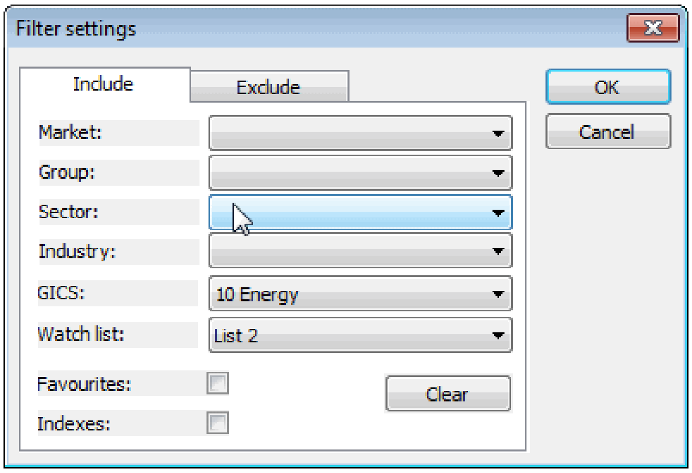
There is a small arrow button next to the "Result list" horizontal divider line. This button is provided to enlarge/shrink the result list. When you are editing your formula it is good to have edit view larger, but to see the backtesting results it is convenient to enlarge the result list. In that case just click on that button and the result list will be enlarged (and the edit field will get shrinked). To do the reverse just click the button again.

Filter settings window

This window is available from "Filter/Define..." button in quick-review and analysis windows.

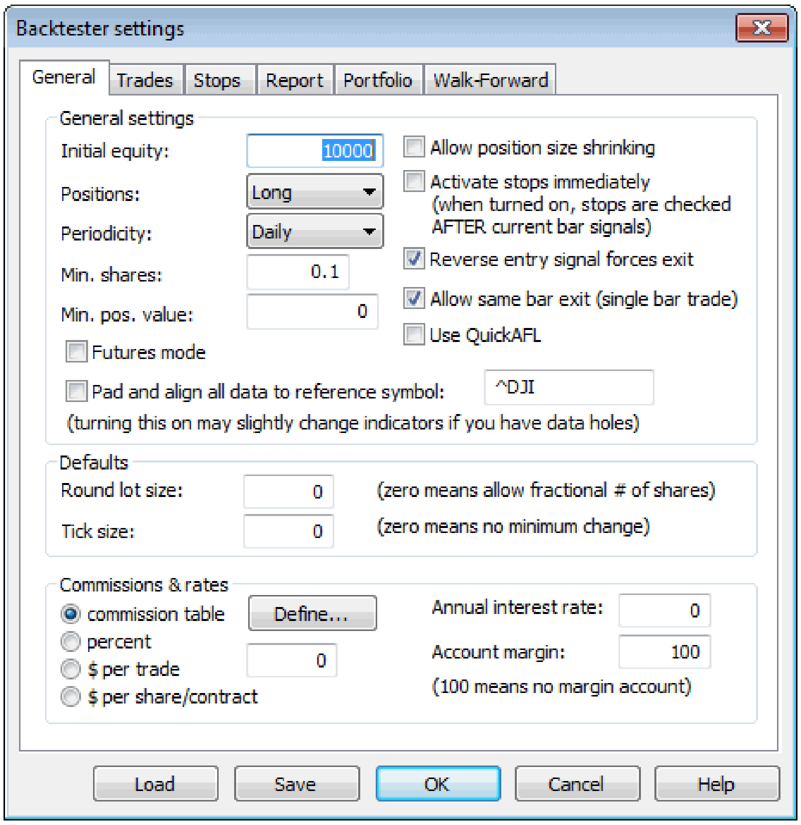
Filtering option gives you ability to narrow your search to symbols belonging to the specified market, group, sector and industry. You can also mark to include only favourites or indexes. You can use include and/or exclude type filter so you can also selectively exclude some kind of symbols .

If you use more than one category (for example you select Market and Sector) the filter will pass only those symbols that match first AND second category (this logical conjunction, not alternative)



Market:	
Group:	
Sector:	
Industry:	
GICS:	10 Energy
Watch list:	List 2
Favourites:	☐
Indexes:	☐

System test settings window



Backtester settings

General | Trades | Stops | Report | Portfolio | Walk-Forward

General settings

Initial equity: 10000

Positions: Long

Periodicity: Daily

Min. shares: 0.1

Min. pos. value: 0

Futures mode

Allow position size shrinking

Activate stops immediately (when turned on, stops are checked AFTER current bar signals)

[✓] Reverse entry signal forces exit

[✓] Allow same bar exit (single bar trade)

Use QuickAFL

Pad and align all data to reference symbol: ^DJI

(turning this on may slightly change indicators if you have data holes)

Defaults

Round lot size: 0 (zero means allow fractional # of shares)

Tick size: 0 (zero means no minimum change)

Commissions & rates

[✓ commission table] Define...

percent

$ per trade 0

$ per share/contract

Annual interest rate: 0

Account margin: 100

(100 means no margin account)

Load | Save | OK | Cancel | Help

Here you can define the following parameters of back-testing:

General tab

Initial equity - defines the size of your account. In Portfolio backtest - it represents entire portfolio size. In "Individual" backtest it is per-symbol initial equity.

Positions considered (long, short, both long and short)

Futures mode

This check box in the settings page is the key to backtesting futures. It instructs backtester to use margin deposit and point value in calculations.

Min. shares

The minimum number of shares that are allowed to buy/short. Backtester will not enter trades below that limit. Should be 1 for stocks. Fractional values are good for mutual funds.

Min. pos value

The minimum position value (in base currency) of the trade that is allowed to be entered. Backtester will not enter trades below that limit. Zero means no limit.

Pad and align to reference symbol

When this is turned on, all symbols' quotes are padded and aligned to reference symbol. Note: by default this setting is OFF. Use responsibly. It may slow down backtest/exploration/scan and introduce some slight changes to indicator values when your data has holes and holes are filled with previous bar data. The feature is intended to be used when your system uses general market timing (generates global signals based on data and/or indicators calculated using Foreign from 'reference' symbol) or when you are creating composites out of unaligned data. Note: if reference symbol does not exist, data won't be padded.

Account margin

This setting defines percentage margin requirement for entire account. The default value of Account margin is 100. This means that you have to provide 100% funds to enter the trade, and this is the way how backtester worked in previous versions. But now you can simulate a margin account. When you buy on margin you are simply borrowing money from your broker to buy stock. With current regulations you can put up 50% of the purchase price of the stock you wish to buy and borrow the other half from your broker. To simulate this just enter 50 in the Account margin field (see pic. 1) . If your intial equity is set to 10000 your buying power will be then 20000 and you will be able to enter bigger positions. Please note that this settings sets the margin for entire account and it is NOT related to futures trading at all. In other words you can trade stocks on margin account.

Commissions

- **commission table** - backtester will use commission table as defined in Commission Schedule table window (press **Define...** button to show it).
- **percent** - commission is expressed as a percent of trade value
- **$ per trade** - commission is fixed amount of dollars (or your currency) per trade
- **$ per share/contract -** commission is expressed in dollars (or your currency) per share/contract purchased/sold

Annual interest rate

This setting allows you to define annual interest earned when you are out of the market or your position is less than available equity.

Periodicity

This setting controls bar interval used for backtesting/scan/exploration/optimization. To backtest intraday data you should switch to proper interval there and then run the backtest.

Allow position size shrinking

If you mark this box AmiBroker will shrink down positions if available equity is less than requested position size (via PositionSize variable). If this box is unmarked positions will not be entered in such case.

Activate stops immediatelly

When you trade on open and want to have built-in stops activated on the same bar - just mark this box.

If you trade on close and want built-in stops to be activated from the next bar - unmark this box.

You may ask why do not simply check the buyprice or shortprice array if it is equal to open price. Unfortunatelly this won't work. Why? Simply because there are doji days when open price equals close and then backtester will never know if trade was entered at market open or close.

Round lot size

Various instruments are traded with various "trading units" or "blocks". For example you can purchase fractional number of units of mutual fund, but you can not purchase fractional number of shares. Sometimes you have to buy in 10s or 100s lots. AmiBroker now allows you to specify the block size on global and per-symbol level.

You can define per-symbol round lot size in the Symbol->Information page. The value of zero means that the symbol has no special round lot size and will use "Default round lot size" (global setting) from the Automatic Analysis settings page. If default size is set also to zero it means that fractional number of shares/contracts are allowed.

You can also control round lot size directly from your AFL formula using RoundLotSize reserved variable, for example:

```
RoundLotSize = 10;
```

Tick size

This setting controls the minimum price move of given symbol. You can define it on global and per-symbol level. As with round lot size, you can define per-symbol tick size in the Symbol->Information page. The value of zero instructs AmiBroker to use "default tick size" defined in the Settings page of Automatic Analysis window. If default tick size is also set to zero it means that there is no minimum price move.

You can set and retrieve the tick size also from AFL formula using TickSize reserved variable, for example:

```
TickSize = 0.01;
```

Note that the tick size setting affects ONLY trades exited by built-in stops and/or ApplyStop(). The backtester assumes that price data follow tick size requirements and it does not change price arrays supplied by the user.

So specifying tick size makes sense only if you are using built-in stops so exit points are generated at "allowed" price levels instead of calculated ones. For example in Japan - you can not have fractional parts of yen so you should define global ticksize to 1, so built-in stops exit trades at integer levels.

Reverse entry signal forces exit

When it is ON (the default setting) - backtester works as in previous versions and closes already open positon if new entry signal in reverse direction is encountered. If this switch is OFF - even if reverse signal occurs backtester maintains currently open trade and does not close positon until regular exit (sell or cover) signal is generated.
In other words when this switch is OFF backtester ignores Short signals during long trades and ignores Buy signals during short trades.

Allow same bar exit (single bar trade)

When it is ON - entry and exit at the very same bar is allowed, when it is OFF then exit may occur only on bars following the entry bar. You may turn "Allow same bar exit" option ON only if you are entering trades on OPEN. If you are entering trades on any other time than bar's open, this option should be turned off to avoid looking into the future.

Use QuickAFL

QuickAFL(tm) is a feature that allows faster AFL calculation under certain conditions. Initially (since 2003) it was available for indicators only, as of version 5.14+ it is available in Automatic Analysis too.

Initially the idea was to allow faster chart redraws through calculating AFL formula only for that part which is visible on the chart. In a similar manner, automatic analysis window can use subset of available quotations to calculate AFL, if selected “range” parameter is less than “All quotations".

Detailed explanation on how QuickAFL works and how to control it, is provided in this Knowledge Base article: http://www.amibroker.com/kb/2008/07/03/quickafl/

Note that this option works in the backtester/optimizer, explorations and scans.

Trades tab

- **prices** buy/sell/short/cover price fields - allows the user to define at which price to buy/sell/short sell/buy to cover during system test
- **delays** buy/sell/short/cover delay - allows to define custom delay between signal and trade

Stop tab

- max. loss stop
- profit target stop
- trailing stop
- N-bar stop

See APPLYSTOP function for more details on different stop settings

Report tab

Result list shows

This decides which format of result list is used by new backtester. Possible choices:

- Trade list (the default) - each trade is listed in a separate row. Trades are ordered by exit date by default
- Detailed log - each data bar is listed separately. The log shows scores, positions and other very detailed information useful for debugging your trading system/position sizing/scoring strategies
- Summary - one row per backtest is generated. The row contains backtest summary/statistics (like the report)

Risk free rates

Defines risk free rates for Sharpe and UPI stats

Distribution charts spacing

Defines the spacing of profit, MAE and MFE distribution charts. The spacing is the % amount of profit/MAE/MFE per single bar in a chart.

Generate detailed reports for individual backtests

This causes that in Individual backtest mode full report is generated and stored for every security under test. Note that this will slow down the test and take up quite a bit of hard disk space

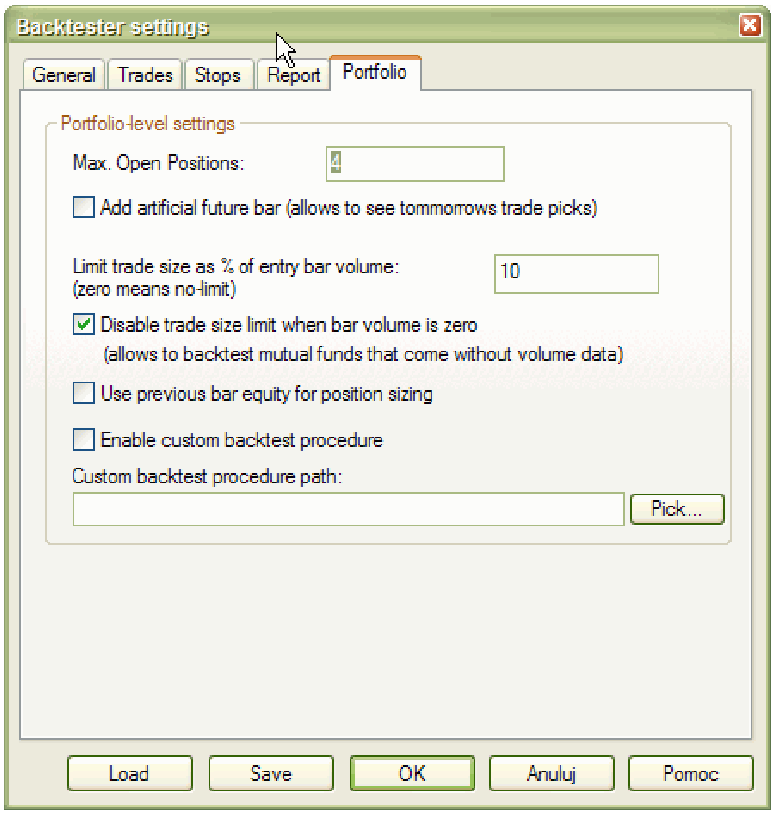
Include trade list in the report

When turned ON (by default) the backtest report includes also trade list. Note that trade lists may be huge and consume quite a bit of disk space

Warn before time-consuming optimizations

When turned ON (by default), AmiBroker will display confirmation dialog box when your optimization has more than 300 steps.

Portfolio tab



Portfolio-level settings

Max. Open Positions: 4

☐ Add artificial future bar (allows to see tommorrows trade picks)

Limit trade size as % of entry bar volume: (zero means no-limit) 10

☑ Disable trade size limit when bar volume is zero (allows to backtest mutual funds that come without volume data)

☐ Use previous bar equity for position sizing

☐ Enable custom backtest procedure

Custom backtest procedure path:

Max. Open Positions

Max. Open Positions - the maximum number of simultaneously open positions. .Settable also using SetOption("MaxOpenPositions", number) function.

Add artificial future bar

When checked AmiBroker adds tommorrow's bar and this enables you to see tommorrow's (or next bar) trade recommendations when your system uses one bar delay. Artificial future bar is has incremented date and volume set to zero and all price fields (OHLC) set to CLOSE price of last data bar.

Limit trade size as % of entry bar volume

This prevents from entering the trades greater than given percentage of entry bar's volume. For example if backtesting daily data and today's volume for thinly traded stock is 177,000 shares, setting this to 10% will limit the maximum trade size to 17,700 shares (10% of total daily volume). This prevents from 'affecting the market' by huge orders.

IMPORTANT NOTE:
Some instruments like MUTUAL FUNDS come without VOLUME data. To backtest such instruments please set this field to ZERO (0) or check "Disable trade size limit weh bar volume is zero" box. This effectively turns OFF this feature. Otherwise you won't be able to enter any trade at all.

Disable trade size limit when bar volume is zero

When it is turned ON and the entry bar volume is zero the backtesterwill not apply the "limit trade size as % of entry bar volume"- this is to allow backtesting mutual funds that come with zero volume data When it is OFF and entry bar volume is zero then backtester will not allow to enter the trade on such bar.

Use previous bar equity for position sizing

Affects how percent of current equity position sizing is performed.
Unchecked (default value) means: use current (intraday) equity to perform position sizing, checked means: use previous bar closing equity to perform position sizing.

Enable custom backtest procedure

When checked AmiBroker applies the custom backtest formula specified in the field below to every backtest that you run. This is useful if you want to permantently add your custom metrics to all backtests without need to copy paste the same code.

Custom backtest procedure path

The full path to custom backtest formula (see above).

Old tab

Drawdown figures based on...

Drawdown figures in the backtest report measure equity dip experienced during the trade(s). To calculate the dip you can use the worst case scenario: low price for long trades and high price for short trades or single price (open or close) for both long and short trades. "Drawdown figures based on..." setting (pic. 2) allows you to choose the price(s) used to calculate drowndowns. Using worst case scenario you will get a few percent bigger drawdowns than using close or open price. On the other hand Equity() function always uses shortprice/coverprice array so you may choose open or close field here to match drawdowns as observed in equity line.

Formula

- mark this box to include AFL formula in the backtest report

Settings

- mark this box to include settings in the backtest report

Incl. out-of-market pos

- mark this box to include out-of-market positions in the backtest report

Overall summary

- mark this box to include sum of individual symbol backtest results

Symbol summary

- mark this box to include per-symbol summaries

Trade list

- choose format of trade list included in the report

System test report window

NEW BACKTESTER REPORT

Exposure % - 'Market exposure of the trading system calculated on bar by bar basis. Sum of bar exposures divided by number of bars. Single bar exposure is the value of open positions divided by portfolio equity.

Net Risk Adjusted Return % - Net profit % divided by Exposure %

Annual Return % - Compounded Annual Return % (CAR) - this is

Risk Adjusted Return % - Annual return % divided by Exposure %

Avg. Profit/Loss - (Profit of winners + Loss of losers)/(number of trades)

Avg. Profit/Loss % - '(% Profit of winners + % Loss of losers)/(number of trades)

Avg. Bars Held - sum of bars in trades / number of trades

Max. trade drawdown - The largest peak to valley decline experienced in any single trade

Max. trade % drawdown - The largest peak to valley percentage decline experienced in any single trade

Max. system drawdown - The largest peak to valley decline experienced in portfolio equity

Max. system % drawdown - The largest peak to valley percentage decline experienced in portfolio equity

Recovery Factor - Net profit divided by Max. system drawdown

CAR/MaxDD - Compound Annual % Return divided by Max. system % drawdown

RAR/MaxDD - Risk Adjusted Return divided by Max. system % drawdown

Profit Factor - Profit of winners divided by loss of losers

Payoff Ratio - Ratio average win / average loss

Standard Error - Standard error measures chopiness of equity line. The lower the better.

Risk-Reward Ratio - Measure of the relation between the risk inherent in a trading the system compared to its potential gain. Higher is better. Calculated as slope of equity line (expected annual return) divided by its standard error.

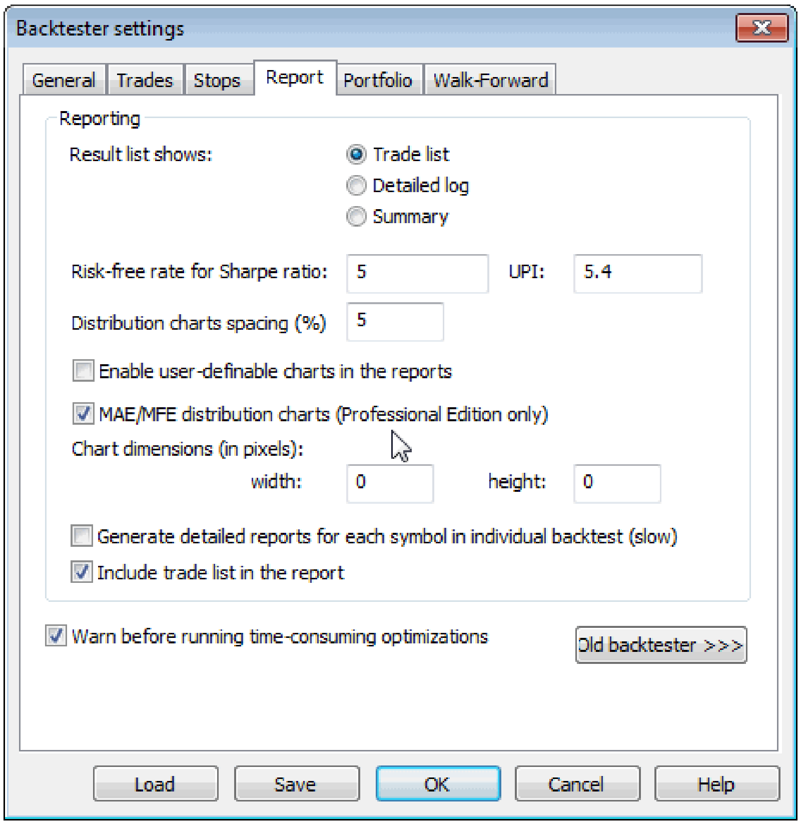
Ulcer Index - Square root of sum of squared drawdowns divided by number of bars

Ulcer Performance Index - (Annual profit - Tresury notes profit)/Ulcer Index'>Ulcer Performance Index. Currently tresury notes profit is hardcoded at 5.4. In future version there will be user-setting for this.

Sharpe Ratio of trades - Measure of risk adjusted return of investment. Above 1.0 is good, more than 2.0 is very good. More information http://www.stanford.edu/~wfsharpe/art/sr/sr.htm . Calculation: first average percentage return and standard deviation of returns is calculated. Then these two figures are annualized by multipling them by ratio (NumberOfBarsPerYear)/(AvgNumberOfBarsPerTrade). Then the risk free rate of return is subtracted (currently hard-coded 5) from annualized average return and then divided by annualized standard deviation of returns.

K-Ratio - Detects inconsistency in returns. Should be 1.0 or more. The higher K ratio is the more consistent return you may expect from the system. Linear regression slope of equity line multiplied by square root of sum of squared deviations of bar number divided by standard error of equity line multiplied by square root of number of bars. More information: Stocks & Commodities V14:3 (115-118): Measuring System Performance by Lars N. Kestner

OLD BACKTESTER REPORT



Reporting

Result list shows: [✓ Trade list] Detailed log Summary

Risk-free rate for Sharpe ratio: 5 UPI: 5.4

Distribution charts spacing (%) 5

☐ Enable user-definable charts in the reports

☑ MAE/MFE distribution charts (Professional Edition only)

Chart dimensions (in pixels): width: 0 height: 0

☐ Generate detailed reports for each symbol in individual backtest (slow)

☑ Include trade list in the report

☑ Warn before running time-consuming optimizations

This window (accessible from **Report** button in Automatic analysis window) provides very useful information about the performance of a trading system under the test. The information included here can be customized using system test settings dialog.

Explanation of values:

Total net profit: This is total profit/loss realized by the test. Includes the closed-out value of the open position (if there is any).
Return on account: This is total profit/loss as a percentage of initial investment.
Total commissions paid: The amount of commissions paid during trades.
Open position gain/loss: The closed-out value of open position that existed at the end of the test.
Buy-and-hold profit: The total profit/loss realized by buy-and-hold strategy (including commission).
Buy-and-hold % return: The total buy-and-hold strategy return as a percentage of initial investment.
Bars in test: The number of bars tested (Overall summary shows sum of number of bars in all symbols).
Days in test: The number of days between first bar date and last bar date (overall summary shows arithmetic

average of number of days accross the population of symbols under test)
System to buy-and-hold index: An index showing how much better/worse is the system compared to buy-and-hold strategy. A value of 0% means that system gives the same profit as buy-and-hold strategy. A value of 200% means that system gives 200% more profit than buy-and-hold strategy. A value of -50% means that system gives a half of the gains of buy-and-hold strategy.
Annual system % return: Calculated compound annual percentage return of the system (*see the note)
Annual B&H % return: Calculated compound annual percentage return of the buy and hold strategy (*see the note)
System drawdown: The largest equity dip experienced by the system (relative to the initial investment).
B&H drawdown: The largest equity dip experienced by the buy and hold strategy (relative to the initial investment).

Max. system drawdown: The largest point distance between equity peak value and the following trough value experienced by the system
Max. system % drawdown: The largest percentage distance between equity peak value and the following trough value experienced by the system
Max. B&H drawdown: The largest point distance between equity peak value and the following trough value experienced by the buy and hold strategy
Max. B&H % drawdown: The largest percentage distance between equity peak value and the following trough value experienced by the buy and hold strategy

Trade drawdown: The largest equity dip experienced by any single trade (relative to the trade's entry price).
Max. trade drawdown: The largest point distance between equity peak value and the following trough value experienced by any single trade
Max. trade % drawdown: The largest percentage distance between equity peak value and the following trough value experienced by any single trade

Total number of trades: The number of trades (winners + losers)
Percent profitable: The number of winning trades compared to total number of trades shown as a percentage
Profit of winners/Loss of losers: Total amount of money gained in winners/lost in losers.
Total # of bars in winners/losers: The number of bars spent during winning/losing trades
Largest winning/losing trade: The amount of biggest winner/loser
# of bars in largest winner/loser: The number of bars in the biggest winning/losing trade
Average winning/losing trade: The average of winning/losing trades (sum of winners/losers divided by a number of winning/losing trades)
Average # of bars in winners/losers: The average of number of bars in winning/losing trades (total number of bars in winners/losers divided by a number of winning/losing trades)
Max consec. winners/losers: The largest number of consecutive winning/losing trades.

Bars out of the market: The number of bars for which the system was completely out of the market (was neither long nor short). If you open and close the position during single day, even if you have no open position on market open and no position on close this day is NOT considered as out of the market.

Interest earned: The total interest earned between trades. Note that AmiBroker simulates O/N (overnight) deposits. This means that if you closed the position on Monday and opened the next one on Tuesday you earn interest for single O/N deposit.

Exposure: Shows how much you are exposed to the market. It is a ratio of bars in the market divided by total number of bars under test. (The number of bars in the market is given by total number of bars minus bars out of the market)

Risk adjusted ann. return: Shows annual return of the system (*see note) adjusted (divided) by market exposure. If your system gained 10% over one year with the exposure of 50% the adjusted return would be 20% (10%/0.5)

Ratio avg win/avg loss: The absolute value of the ratio of average winning trade to average losing trade
Profit factor: The absolute value of the ratio of the profit of winners to loss of losers
Avg. trade (win & loss): The average trade profit calculated as sum of winners and losers divided by the number of trades.

<u>*Note: Calculation method used for annual percentage returns:</u>

Most of the software (including two the most popular so-called professional packages) use very simple annualization method based on the following formula:

simple_annualized_percentage_return = percentage_return * (365 / days_in_test);

unfortunatelly this method is **wrong** and very misleading since it would tell you that annual return is 22% when your system earned 44% during two years. This value is too optimistic. In fact annual return in this case is only 20%: if your initial investment was 10000 you earn 20% during the first year so you then get 12000 and 20% the second year that gives you 14400 = (12000 * 120 %). So after two years you earned 44% but annually it is only 20%.

AmiBroker is one of the few programs that calculates annual returns correctly and will give you correct value of 20% as shown in the example above. The formula that AmiBroker uses for annual return calculation is as follows:

correctly_annualized_perc_return = 100% * ((final_value/initial_value) ^ (365 / days_in_test) - 1)

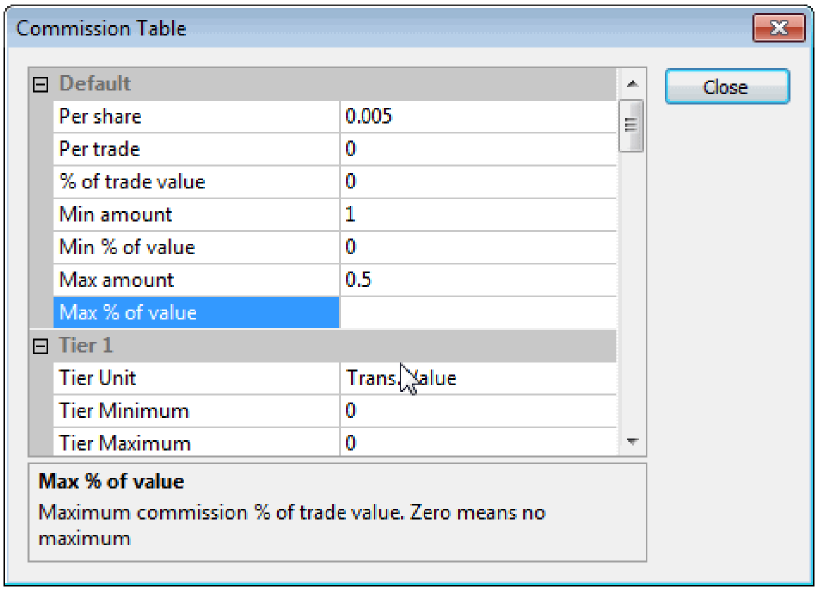
where x^y means rising x to the power of y.

Known differencies between statistics produced by 'old' and <u>'new' (portfolio) backtester</u>

	Old backtester	New (portfolio) backtester
System and trade drawdown calculations based on	Open/Close/H-L range (worst case) selectable in settings	Close price only (regardless of settings) - subject to change
Max. % trade drawdown	Calculated based on total equity	Calculated based on ACTUAL trade value at entry point.
Stats available	for all trades only	separately for long, short and all trades
PositionSizing	Based on individual symbol equity	Based on portfolio equity. PositionSize = -25; will enter 25% of current porfolio equity
Trade statistics	Include only closed trades, open trade is reported separately	Include all trades (closed and those still open at the end of analysis period). Any open trades are closed out at 'close' price always.
Exposure		

	calculated regardless of position size (no matter on what is position size if trade is taken for particular bar it assumes 100% exposure at that bar)	calculations include now (in 4.43.0) the total amount of open positions compared to total portfolio equity. Exposure is calculated on bar by bar basis so if only 50% funds are in open trade, then exposure for this bar is 0.5. Then individual bar exposures are summed up and divided by number of bars to produce exposure figure. This way true market exposure is calculated.
Multiple security testing	N independent accounts (multiple single equity)	Portfolio equity common to all symbols under test

Commission window



Default	
Per share	0.005
Per trade	0
% of trade value	0
Min amount	1
Min % of value	0
Max amount	0.5
Max % of value	
Tier 1	
Tier Unit	Trans. Value
Tier Minimum	0
Tier Maximum	0

Max % of value
Maximum commission % of trade value. Zero means no maximum

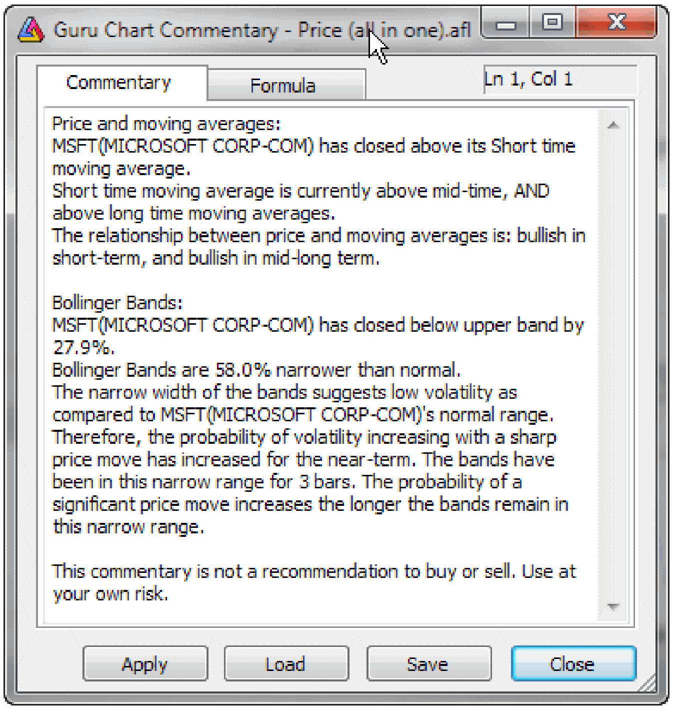
Commission table is available in the Account manager and in **Automatic Analysis -> Settings** window, "**General**" tab, **"Commission and rates: Define..."**

In this window you can enter commission taken by buy/sell transactions.

There are 5 tiers of commission schedule table plus "default" tier that is used when others are not defined or transaction does not match any tier defined. Tiers can be defined based on transaction value or number of shares/contracts traded. Each tier has user definable minimum and maximum. If min/max is not defined or set to zero - the tier is not active.

Each tier allows to define commission on per-share, per-trade, % of trade volume basis and allows to define minimum and maximum commission values based on dollar or percent values.

Commentary window



Price and moving averages:
MSFT(MICROSOFT CORP-COM) has closed above its Short time moving average.
Short time moving average is currently above mid-time, AND above long time moving averages.
The relationship between price and moving averages is: bullish in short-term, and bullish in mid-long term.

Bollinger Bands:
MSFT(MICROSOFT CORP-COM) has closed below upper band by 27.9%.
Bollinger Bands are 58.0% narrower than normal.
The narrow width of the bands suggests low volatility as compared to MSFT(MICROSOFT CORP-COM)'s normal range.
Therefore, the probability of volatility increasing with a sharp price move has increased for the near-term. The bands have been in this narrow range for 3 bars. The probability of a significant price move increases the longer the bands remain in this narrow range.

This commentary is not a recommendation to buy or sell. Use at your own risk.

Commentary window enables you to view textual descriptions of actual technical situation on given market.

Commentaries are generated using formulas written in AmiBroker's own formula language. You can find the description of this language in AmiBroker Formula Language Reference Guide.

Moreover Commentary feature gives you also graphical representation of buy & sell signals by placing the marks (arrows) on the price chart.

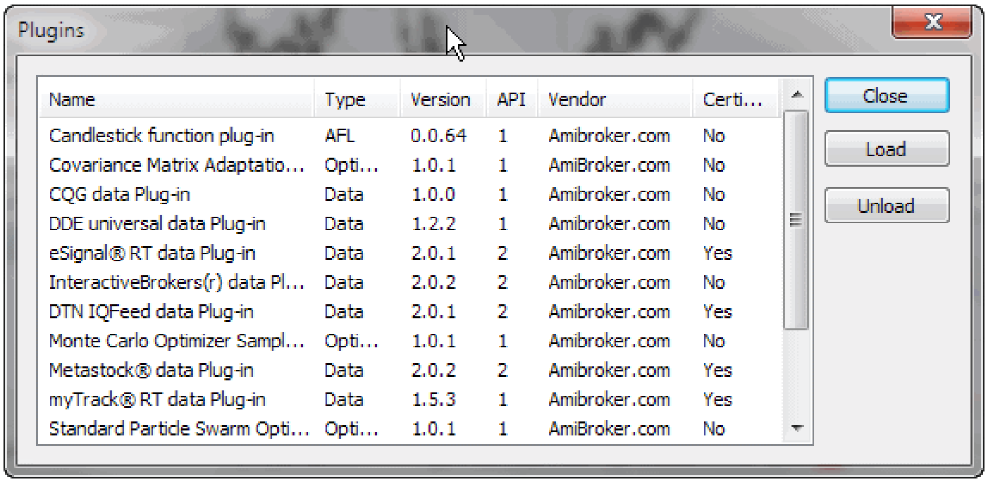
Newbies should read "Tutorial: How to write your own commentary" for step-by-step instructions and working with AFL editor.

"Refresh" button causes AmiBroker to reinterpret the commentary using currently selected symbol/date.
"Load" and "Save" buttons allow to load/save commentary formulas.
"Close" button closes the commentary window.

Now the Guru chart commentary window is automatically updated and sychronized with the date selected on the chart using "Pick" selector tool. This way you can easily read any indicator value on any selected date right off the chart commentary window.

Plugins window

Plugins window lists all loaded plug in DLLs. It is useful for inspecting which plugins are active.



Name	Type	Version	API	Vendor	Certi...
Candlestick function plug-in	AFL	0.0.64	1	Amibroker.com	No
Covariance Matrix Adaptatio...	Opti...	1.0.1	1	AmiBroker.com	No
CQG data Plug-in	Data	1.0.0	1	Amibroker.com	No
DDE universal data Plug-in	Data	1.2.2	1	Amibroker.com	No
eSignal® RT data Plug-in	Data	2.0.1	2	Amibroker.com	Yes
InteractiveBrokers(r) data Pl...	Data	2.0.2	2	Amibroker.com	No
DTN IQFeed data Plug-in	Data	2.0.1	2	Amibroker.com	Yes
Monte Carlo Optimizer Sampl...	Opti...	1.0.1	1	Amibroker.com	No
Metastock® data Plug-in	Data	2.0.2	2	Amibroker.com	Yes
myTrack® RT data Plug-in	Data	1.5.3	1	Amibroker.com	Yes
Standard Particle Swarm Opti...	Opti...	1.0.1	1	AmiBroker.com	No

In addition to just showing the list of plugins you can unload all DLLs by pressing "Unload" button and load them back by pressing "Load" button. Please note that a DLL **must** be placed in the "Plugins" subfolder of AmiBroker main directory to be seen.

At start AmiBroker scans the "Plugins" folder and loads the DLLs that follows the specifications of AmiBroker plugin. If a DLL is loaded it is "locked" for writing so it can not be overwritten or modified.

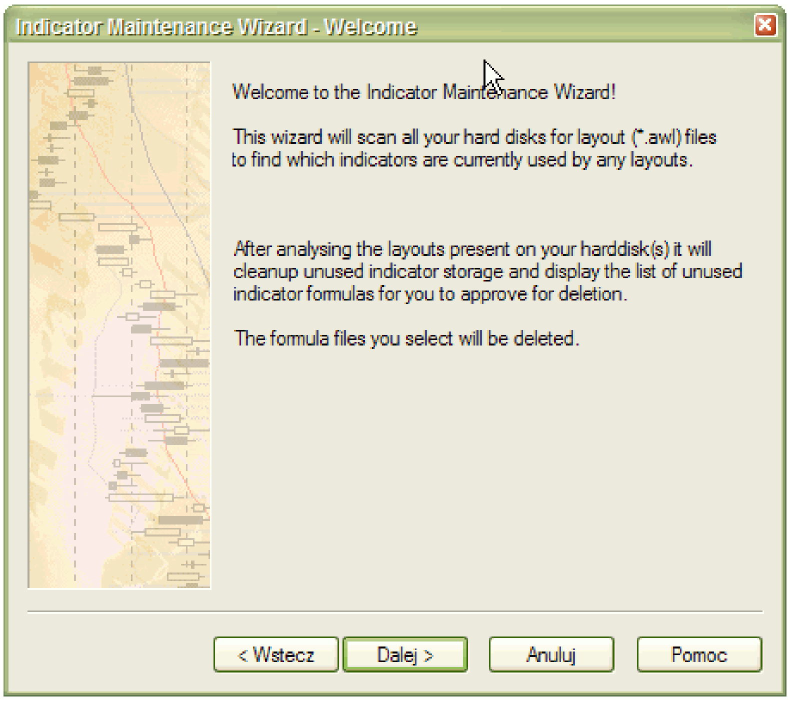
During the development process it is necessary to overwrite/modify the DLL code - because when you apply the changes to the source code these changes must be recompiled and stored into DLL file. To allow the developer to overwrite the DLL used by AmiBroker the "Unload" function is available in this window. Unloading releases the DLL so it can be overwritten without the need to restart AmiBroker. Then, after modifying the DLL code, you can load the DLL back using "Load" function.

IMPORTANT NOTE: AmiBroker makes no representations on features and performance of non-certified third-party plug-ins. Specifically certain plug-ins can cause instabilities or even crashes. Entire use of non-certified third-party plugins is at your own risk.

Indicator Maintenance Wizard

Indicator maintenance checks for any indicators that were deleted from any layouts on your hard disk and frees table from entries
allocated for indicators that were deleted. This procedure is rarely but still needed because if you delete indicator from one
layout there is no guarantee that there is no other layout file buried somewhere on your hard disk that still references given indicator.

So Indicator Maintenance scans all hard disks and all partitions looking for layout files and analysing them to built the
" actually used" table of indicators.



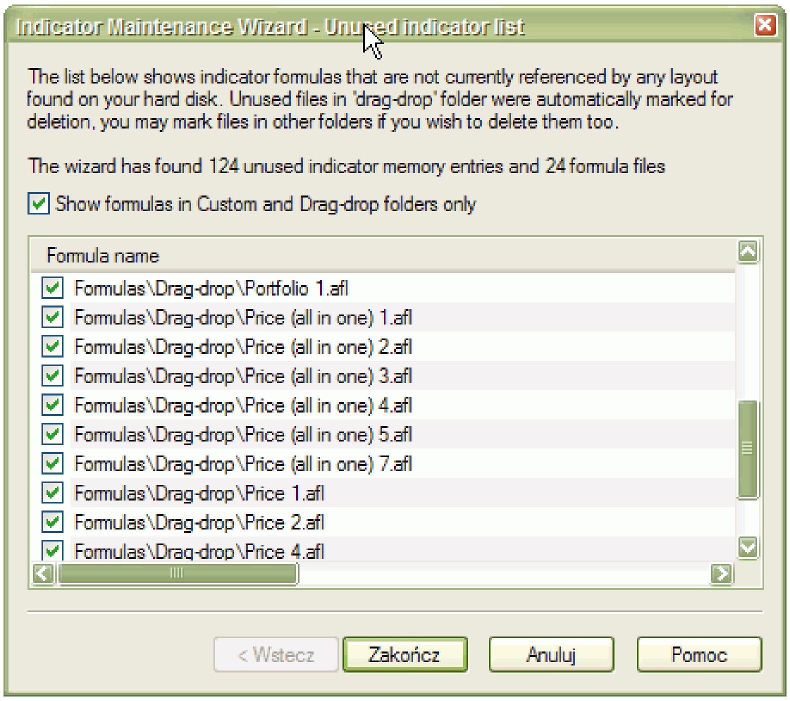
Welcome to the Indicator Maintenance Wizard!

This wizard will scan all your hard disks for layout (*.awl) files to find which indicators are currently used by any layouts.

After analysing the layouts present on your harddisk(s) it will cleanup unused indicator storage and display the list of unused indicator formulas for you to approve for deletion.

The formula files you select will be deleted.

The ones which are not referenced by any layout can be deleted from internal table.
Depending on your choice you may leave default behaviour (cleaning up only internal table) or deleting actual formula files
that are not referenced. This is up to you. If you don't use particular formula for say AA Scan/Backtest/Optimization you can
delete it. If you use it or need it for some archive purposes - leave it unchecked.



The list below shows indicator formulas that are not currently referenced by any layout found on your hard disk. Unused files in 'drag-drop' folder were automatically marked for deletion, you may mark files in other folders if you wish to delete them too.

The wizard has found 124 unused indicator memory entries and 24 formula files

[✓] Show formulas in Custom and Drag-drop folders only

Formula name
✓ Formulas\Drag-drop\Portfolio 1.afl
✓ Formulas\Drag-drop\Price (all in one) 1.afl
✓ Formulas\Drag-drop\Price (all in one) 2.afl
✓ Formulas\Drag-drop\Price (all in one) 3.afl
✓ Formulas\Drag-drop\Price (all in one) 4.afl
✓ Formulas\Drag-drop\Price (all in one) 5.afl
✓ Formulas\Drag-drop\Price (all in one) 7.afl
✓ Formulas\Drag-drop\Price 1.afl
✓ Formulas\Drag-drop\Price 2.afl
✓ Formulas\Drag-drop\Price 4.afl

If you are not sure what options to choose, just press "Next" all the time and you will safely complete the procedure without changing any settings.

Log window

The Log window (available from **Window->Log** menu) allows to view:

- edit-time errors displayed during formula check
- run-time errors that occur when formula is running (not edited)
- _trace command output within AmiBroker (without using 3rd party debug view)

To perform tasks such clearing the output, copying, changing settings use right - mouse click over the log window list.
Double click on the error line brings up the editor, so you can fix the error easily.

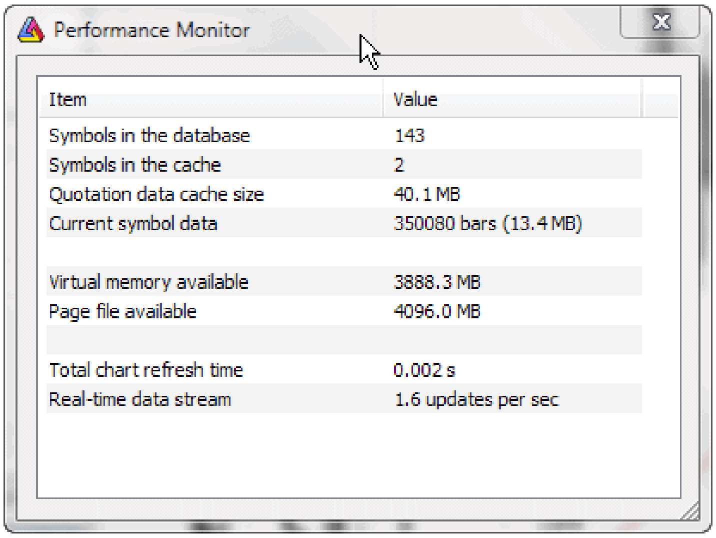
While "edit-time" error list is cleared automatically each time you check the syntax in the editor, the run-time error list is NOT cleared, so all errors remain listed, even if they are fixed already, unless you manually clear the list.

Note that _TRACE output is by default directed to outside debugger (like DebugView), in order to enable internal display you need to switch appropriate option in the **Tools->Preferences->AFL** You can choose to display internally / externally or in both places.
Internal _trace has much lower performance penalty (order of magnitude) than external, but that is achieved by the fact that
internal log window is refreshed only when application is not busy. It is appropriate for some uses, but you may prefer more immediate
refresh offered by DebugView.

Note that internal log window accepts special string "!CLEAR!" that causes deleting contents of the log window, as presented in the example below:

```
_TRACE("!CLEAR!"); // this clears the internal log window.
_TRACE("First line after clear");
```

Performance Monitor window



Item	Value
Symbols in the database	143
Symbols in the cache	2
Quotation data cache size	40.1 MB
Current symbol data	350080 bars (13.4 MB)
Virtual memory available	3888.3 MB
Page file available	4096.0 MB
Total chart refresh time	0.002 s
Real-time data stream	1.6 updates per sec

The performance monitor is available from **Tools->Performance Monitor** menu and it shows some memory and usage statistics:

- number of symbols in the database
- number of symbols cached in RAM
- quotation data memory usage
- current symbol memory usage
- total chart refresh time
- real-time data stream update frequency

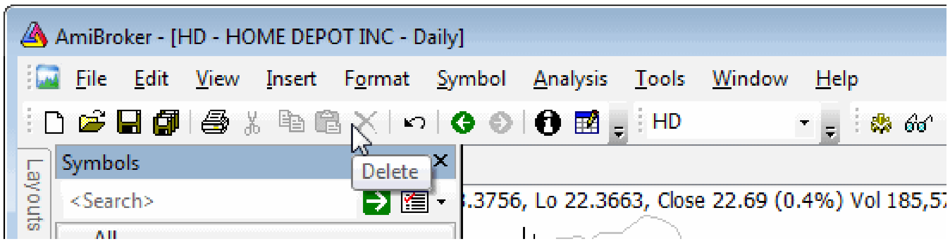
The contents of the window is updated automatically every 3 seconds

This tool is intended to be used now for two purposes:
a) tweaking cache settings for best RAM usage (for example optimizations will run faster if all quotation data can be kept in RAM)
b) monitoring real-time performance

More uses will probably come in the future.

Menus

This chapter describes AmiBroker menus.


AmiBroker - [HD - HOME DEPOT INC - Daily]
File Edit View Insert Format Symbol Analysis Tools Window Help
HD
Delete
Symbols
<Search>
Layouts
.3756, Lo 22.3663, Close 22.69 (0.4%) Vol 185,5

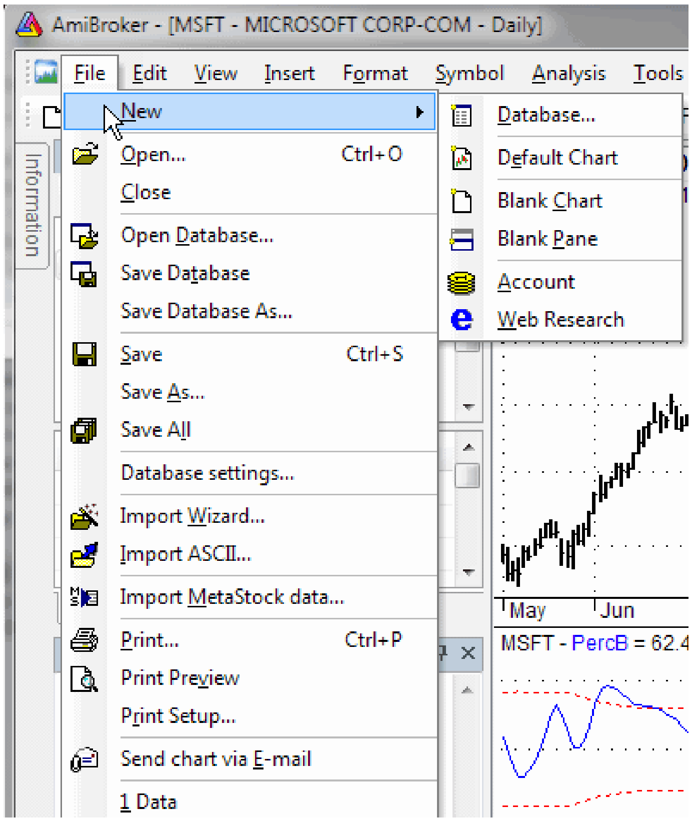
There are following main pull down menus:

- File
- Edit
- View
- Insert
- Format
- Symbol
- Analysis
- Tools
- Window
- Help

And the following CONTEXT menus:

- AFL Editor context menu (available when you click with RIGHT mouse button in the AFL editor)
- Automatic Analysis window context menu (available when you click with RIGHT mouse button over Automatic Analysis RESULT LIST)
- Alert Output context menu (available when you click with RIGHT mouse button in the Alert Output window)
- Chart pane context menu (available when you click with RIGHT mouse button in the chart pane)
- Layouts context menu (available when you click with RIGHT mouse button in the Workspace -> Layouts tree)
- Formula context menu (available when you click with RIGHT mouse button in the Workspace -> Charts tree)
- Layers context menu (available when you click with RIGHT mouse button in the Workspace -> Layers list)
- RealTime Quote context menu (available when you click with RIGHT mouse button in the Real time quote list)

File menu



New

- **Database**
  Creates a new AmiBroker database and launches Database settings window.
- **Default Chart**
  Creates new chart window using default template. It's possible to select the symbols and time frame independently in each of the windows opened.
- **Linked Chart**
  Creates linked chart window based on current template and active chart. Linked windows use the same symbol selection, so if you change the selected symbol for one of them, the other one will synchronize automatically. Linked windows can have DIFFERENT viewing time frame selected. Simply activate the window and select desired interval from View menu for one window, then switch to the other one and select different interval for it. This option allows you to select different time frame or indicators' set in each window and easily move through the database.
- **Blank Chart**
  Creates new (blank) chart window. This is useful if you want to create completely new setup of charts that do not share the same chart IDs. It is important if you want to have indicators that have independent parameters from the other windows that you have created.
- **Blank Pane**
  Creates new (blank) chart pane
- **Account**
  Creates New Account (Account Manager)
- **Web Research**
  Creates New Web Research window

Open
Opens document (account, database or HTML file - you can pick document type from "Files of type" combo in the File selector window)

Close
Closes current (active) document window (chart, account, web research)

Open Database
Allows you to open an existing AmiBroker database. Please select the database folder and press OK.

Save Database
Saves the currently used database

Save Database As...
Saves database into new location

Save
Saves current document (account, html file)

Save As...
Saves current document (account, html file) under new name

Save All
Saves all documents currently open

Database Settings
Opens Database settings dialog that allows you to change your database parameters or intraday settings.

Import Wizard
Launches ASCII Import Wizard window, that allows you to easily import ASCII (text) files into your database

Import ASCII
Allows you to import ASCII files with use of predefined import formats. To learn more how to use ASCII importer, please read ASCII Importer reference chapter.

Import MetaStock data
Launches Metastock importer window. *IMPORTANT NOTE: Metastock importer should be used ONLY if you want to import MS data to native, local AmiBroker database once. If you want AmiBroker to just read Metastock database* ***DIRECTLY*** *without need to import new data over and over please set up your database WITH METASTOCK PLUGIN as described in the Tutorial.*

Print
Allows you to print currently displayed charts.

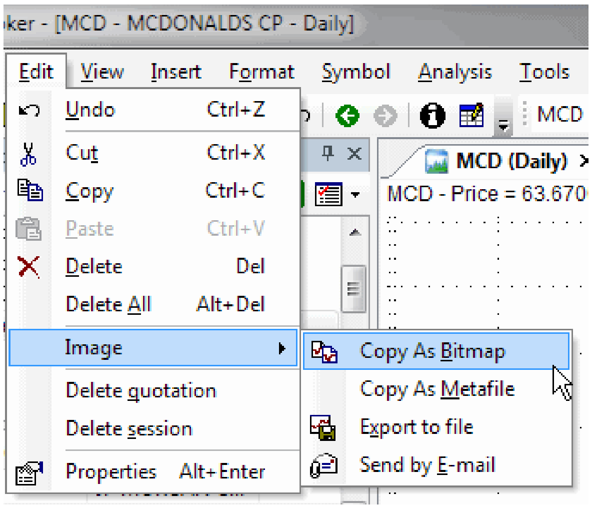
Print Preview
Prints currently displayed charts with the preview (you can check the appearance of the document before it's printed).

Print Setup
Opens printout setup dialog.

Send Chart via E-mail
AmiBroker creates .png image (with the currently displayed chart) and uses your default mailing program (e.g. Outlook Express) to send the file as an attachement.

Exit
Closes AmiBroker program.

Edit menu



Undo
Allows to undo the last operation performed on chart studies (trendlines etc.). This option will be unavailable if no study has been drawn or moved.

Cut, Copy, Paste, Delete
These options can be used to cut, copy, paste or delete studies from the chart. Cut, copy and delete will be greyed out if no object on the chart is selected. To paste the object, it's necessary to use 'copy' or 'cut' option first.
To learn more about drawing tools in AmiBroker, please read Drawing tools reference chapter.

Delete All
Deletes all the objects from the currently opened chart window.

Image

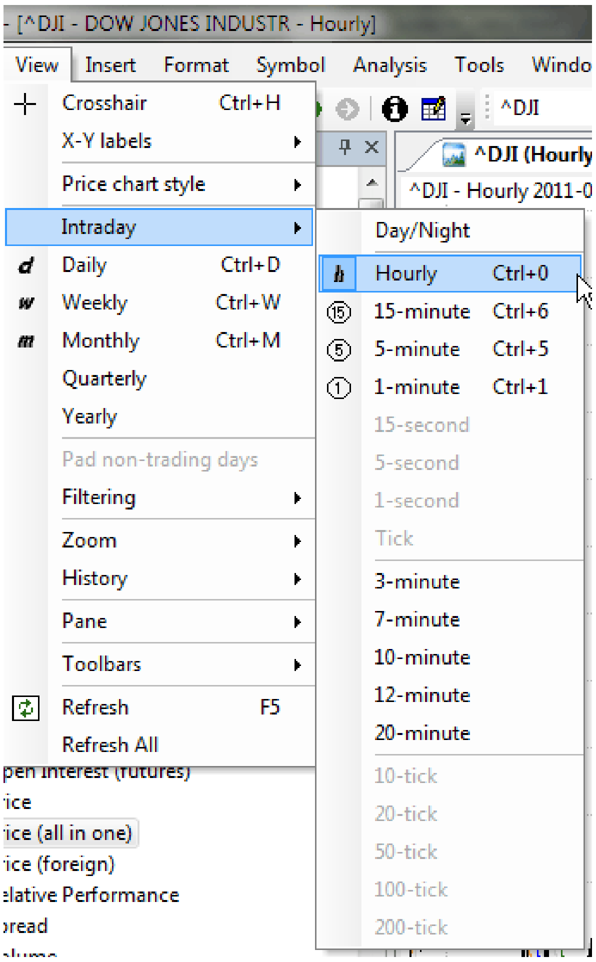
- **Copy As Bitmap** - copies the curently opened chart to the system clipboard as a .BMP image. You can paste the clipboard contents e.g. into 'Paint' application.
- **Copy As Metafile** - copies the curently opened chart to the system clipboard as a metafile
- **Export to file** - saves the currently displayed chart as .PNG file
- **Send by E-mail** - AmiBroker creates .png image (with the currently displayed chart) and uses your default mailing program (e.g. Outlook Express) to send the file as an attachement.

Delete quotation
Deletes currently selected bar.

Delete Session
Deletes currently selected bar from ALL the symbols in the database.

Properties
Opens a study properties dialog. More information can be found in Drawing tools reference chapter.

View menu



Crosshair
Turns on/off crosshair.

X-Y labels
Controls the display of X-Y value labels

- **Off**
- **With crosshair only** - display X-Y value labels when crosshair is activated
- **Always on** - always display X-Y value labels

Price Chart Style
Changes the style of the default Price chart

- **Auto** - uses settings defined in Tools ->Preferences
- **Line** - line chart
- **CandleSticks** - candlestick chart
- **Bars** - traditional bar chart

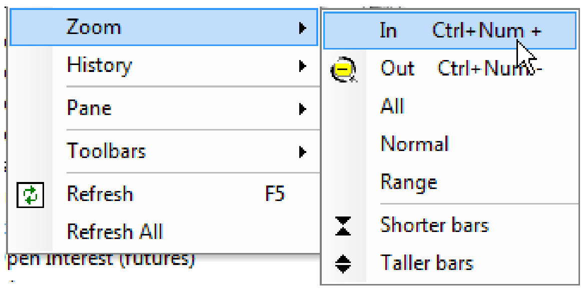
Intraday
Allows you to chose one of intraday time intervals and decide whether to display day or night sessions. Day and night sessions' hours can be set in: Database settings window (*File -> Database Settings -> Intraday settings*) or separately for group in Categories window (*Symbol -> Categories*).

- **Day / Night** - shows two bars (day and night) per day
- **Show 24 hours trading** - no filtering is applied and all the data in the database is included in the chart.
- **Show day session only** - displays day sessions only.
- **Show night session only** - displays night sessions only.
- **Show day and night sessions** - displays day and night sessions.

Daily, Weekly, Monhly, Quarterly, Yearly
Allows to change the display time interval.

Pad non-trading days
Enable padding of Saturdays, Sundays and other non-trading days with previous close price

Filtering
Allows to choose between no filtering (24 hours trading display), regular trading only, extended trading only.

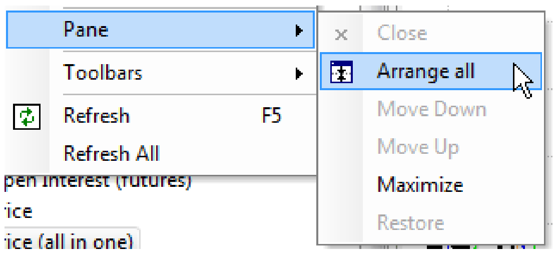


Zoom
Controls the zoom of the chart

- **In** - reduces number of bars displayed
- **Out** - increases number of bars displayed
- **All** - displays all the available bars for the current symbol
- **Normal** - displays default number of bars (defined in Tools -> Preferences -> Charting)
- **Range** - displays the bars from the selected range
- **Shorter bars** - reduces the vertical size of the bars
- **Longer bars** - increases the vertical size of the bars

History
Allows to move Back/Forward in 'browser-like' way.

- **Previous** move to previous symbol (keyboard shortcut: Ctrl+Alt+LEFT)
- **Next** move to next symbol (keyboard shortcut: Ctrl+Alt+RIGHT)



Pane

- **Close** - closes curently selected chart pane
- **Arrange All** - arranges all the displayed charts
- **Move Down** - moves curently selected chart pane one position down
- **Move Up** - moves curently selected chart pane one position up
- **Maximize** - maximizes curently selected chart pane
- **Restore** - restores the charts layout after using Maximize

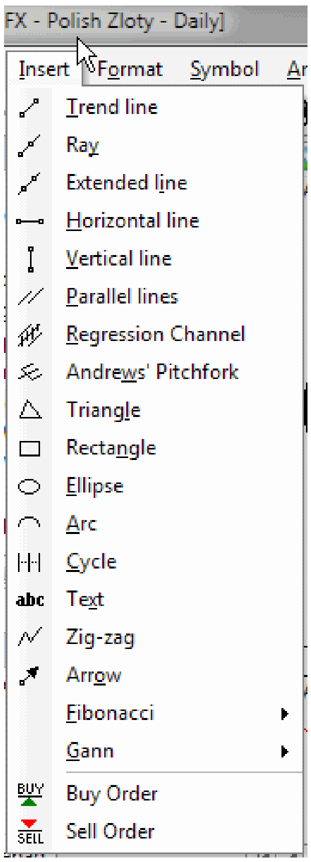
Toolbars
Allows you to display/hide the toolbars.

Refresh
Refreshes the chart window.

Refresh All
Refreshes the chart window and re-reads the contents of all the categories in symbols tree in Workspace window.

Note to users of previous versions: the items that control Symbol, Layouts, Layers, Charts, Information and other windows have been moved to Window menu.

Insert menu



Trend line
Draws a trend line.

To draw a trend line the chart - start drawing by pointing the mouse and pressing left mouse button where you want to start the drawing. Then move the mouse and study tracking line will appear. Release left mouse button when you want to finish drawing. Alternatively you can click once in the place where you want the trendline to begin, move the mouse and click once again to finish drawing. You can also cancel study drawing by pressing ESC (escape) key.

Ray
Draws a ray. Ray is a right-extended trend line.

Extended line
Draws an extended line. Extended line is a trend line that is extended automatically from both left- and right-sides.

Horizontal line
Draws a horizontal line. Horizontal line is self expanding so it is only necessary to click on the chosen

price-level.

Vertical line
Draws a vertical line. Vertical line is self expanding so it is only necessary to click on the chosen bar.

Parallel lines
Draws parallel trendlines.

This tool allows to draw a series of parallel trend line segments. First you draw a trend line as usual, then a second line parallel to the first is automatically created and you can move them around with the mouse. Once you click on the chart it is placed in given position. Then another parallel line appears that can be placed somewhere else. And again, and again. To stop this please either press ESC key or choose "Select" tool.

Regression channel
Draws Raff, standard deviation, standard error channels. To read the detailed information regarding this tool please read Drawing tools reference chapter.

Andrews' pitchfork
Draws an Andrews' pitchfork. Read Drawing tools reference chapter for more detailed information.

Triangle
Draws a Triangle. Left-click in the first point, move to the second point then click once, then move to the third point and click once again.

Rectangle
Draws a rectangle. Left-click in the first point, move to the position where you want to place the oposioposite corner and click once again.

Ellipse
Draws an Ellipse. Ellipse is connected to the date/price coordinates (as trend lines) rather than to the screen pixels so it can change the visual shape when displayed at various zoom factors or screen sizes. To see the properties of ellipse you should double-click on the clock-like 3, 6, 9 or 12 hour positions.

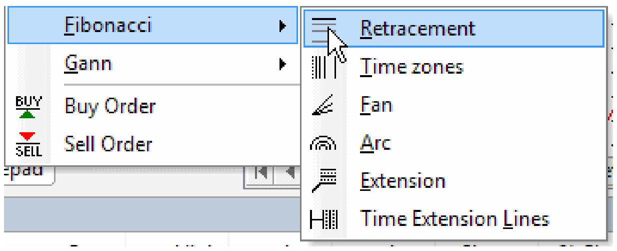
Arc
Draws an Arc. Arc, the same as Ellipse is connected to the date/price coordinates (as trend lines) rather than to the screen pixels so it can change the visual shape when displayed at various zoom factors or screen sizes. To see the properties of ellipse you should double-click on the clock-like 3, 6, 9 or 12 hour positions.

Cycle
Draws time cycles. To use time cycles tool, click on the cycles drawing tool button in the toolbar then click at the starting point of the cycle and drag to the end of the cycle. These two control points control the interval between the cycle lines. When you release the mouse button you will get a series of parallel lines with equal interval in between them.

Text
Allows to place a custom text on the chart. Left-click on the chart to start typing. To finish - click once again on the chart, outside the text box. You can also cancel typing by pressing ESC (escape) key.

Zig-zag
Draws a series of connected trend lines. To finish the series double-click or press ESC (escape) key.

Arrow
Draws a line that ends with an arrow. Drawing technique is exactly the same as drawing a trend line.



Fibonacci
Group of Fibonacci drawing tools. Read Drawing tools reference chapter for more detailed information.

- Fibonacci Retracement study
- Fibonacci Time zones study
- Fibonacci Fan
- Fibonacci Arc
- Fibonacci Extensions
- Fibonacci Time Extension lines

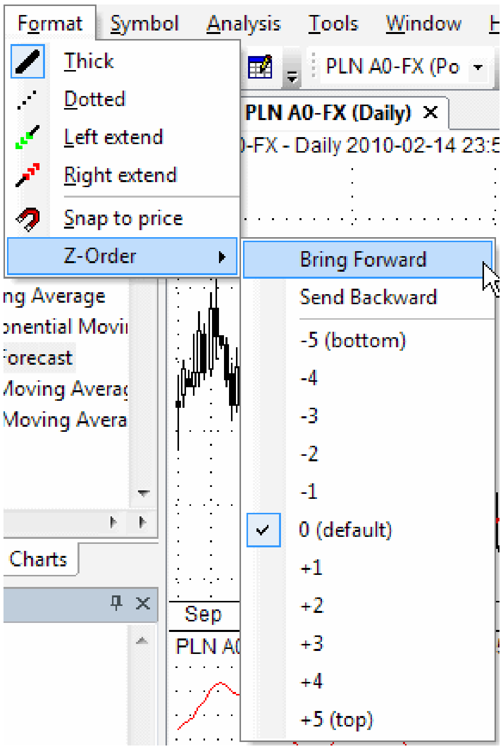
Gann

Group of Gann drawing tools.

- Gann Fan
- Gann Square

Read Drawing tools reference chapter for more detailed information.

Format menu



These options allow you to apply color or style to the objects. Note that you can also select color and style of the object before drawing new object: simply deselect previous object (if any), change style selections and draw new object.

Thick
Changes drawn object formatting to thick style.

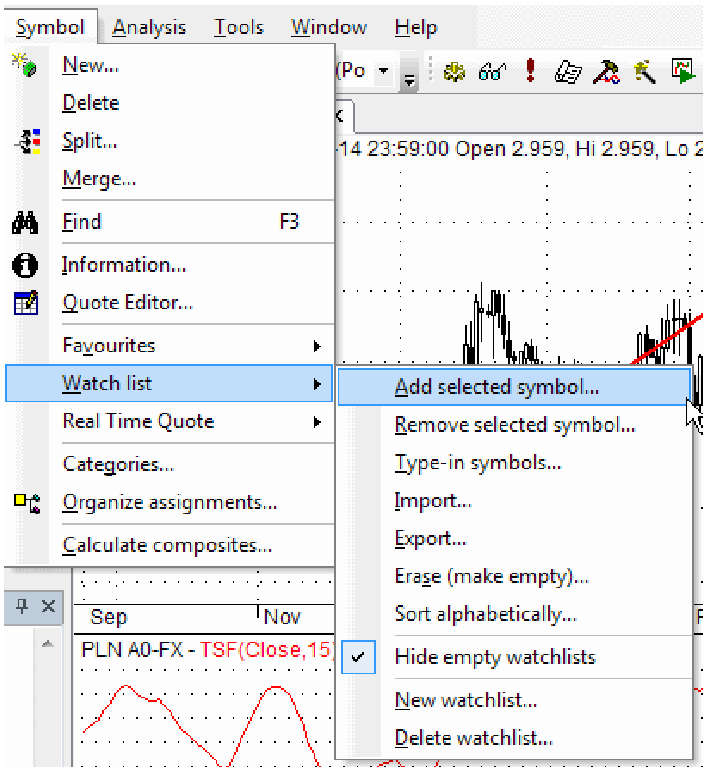
Dotted
Changes study formatting to dotted style.

Left Extend
Extends the trendline to the left.

Right Extend
Extends the trendline to the right.

Snap to price
Turns on the magnet that snaps the drawn studies to the prices. Snap to price % threshold can be set in Preferences window. Snap to price % threshold defines how far price 'magnet' works, it will snap to price when the mouse is nearer than % threshold from H/L/C price

Symbol menu



New
Allows you to add new symbols into the database. After selecting this function you will be prompted for new ticker symbol. Please try not to exceed 26 chars. For proper import functioning you should enter the symbol with CAPITALS.

Delete
Removes currently selected symbol from the database. After choosing this function you will be asked for confirmation of symbol removing. Note well that this operation can not be undone.

Split
Allows you to perform stock split. AmiBroker provides easy way of handling stock splits. Program will try to guess split date and ratio by analyzing quotations. If there is just a single quotation after split this should work, if not you will be asked for split date and ratio. You can specify a split using following expression: x->y which means that x shares before split become y after it. For example 2->3 means that 2 shares become 3 after the split. It is also possible to perform reverse-split, for example 2->1, which means that 2 shares are joined together into 1 share.

Merge
This function allows you to merge two tickers, when the ticker for the symbol is changed and in your database - one symbol holds historical quotes and the second one holds newest quotes (after name change). I You should just select the new ticker (after name change) and use *Symbol->Merge*. Then from the combo you should choose original ticker ("merge with") and optionally check the following fields:

- overwrite duplicate quotes - checking this option will overwrite the quotes already existing in "new" ticker with those present in "old" ticker (this should really not be the case, but may happen).
- delete "merge with" afterwards - checking this option will delete the "old" ticker after merging

- assign alias name - checking this option will copy the "old" ticker to the alias field of the "new" ticker

Find
Opens Symbol finder window that allows you to quickly search the database for a symbol by typing the first letters of its full name or ticker.

Information
Opens the Information window for the symbol, which allows you to change the symbol properties.

Finances
Finances window allows you to enter some fundamental data for the symbol (sales income, earnings before taxes (EBT), earnings after taxes (EAT)). AmiBroker will compute P/E (Price to Earnings ratio) and EPS (Earnings Per Share) indicators out of the data given.

Quote Editor
Opens Quote Editor window that allows you to edit, delete and add quotations into your database.

Watchlist
These options allow you to manage your watchlists. Working with watch lists chapter explains in more detail the way you can use the below options.

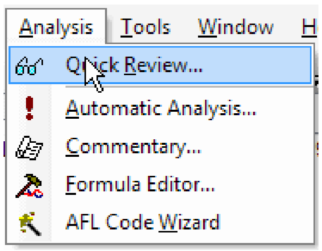
- **Add Selected Symbol** - adds the currently selected symbol to the specified watchlist(s).
- **Remove Selected Symbol** - removes the currently selected symbol from the specified watchlist(s).
- **Type-in Symbols** - allows you to type-in the symbols to the watchlist(s).
- **Import** - allows to import the watchlist from the .TLS file
- **Export** - exports the symbols belonging to the watchlist to the .TLS (symbol list) file
- **Erase (make empty)** - removes all the symbols from the specified watchlist.
- **Sort alphabetically** - sorts tickers alphabetically in the specified watchlist
- **Hide empty watchlists** - hides watch lists with no symbols in the symbol tree
- **New watchlist** - creates new watch list
- **Delete watchlist** - deletes selected watch list (it does not delete symbols from teh database)

Categories
Categories window allows you to define names of markets, groups, sectors and industries. For each market you can also define base indexes for calculating relative strength, composite data, beta or web profile URL.

Organize assignments
Assignment organizer window allows you to easily change the category assignments for the symbols or to delete multiple symbols from the database.

Calculate Composities
Opens Composite calculation window that allows for automatic calculation of number and volume of advancing/declining/unchanged issues or volume numbers for indices.

Analysis menu



Quick Review
Opens Quick review window that provides overall market information like: daily symbol quotes, daily/weekly/monthly/quarterly/yearly returns comparison table or Price/Earnings and Price/Book value comparison.

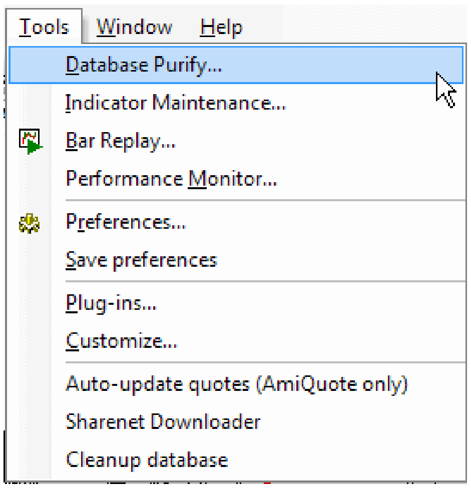
Automatic Analysis
Opens Automatic Analysis window that enables you to check your quotations against defined buy/sell rules or explore your database. AmiBroker can produce report telling you if buy/sell signals occurred on given symbol in the specified period of time, simulate trading, giving you an idea about performance of your system or optimize the trading system you use to improve it's performance.

Commentary
Displays Commentary window which allows you to view textual descriptions of actual technical situation on given market.

Formula Editor
Opens the Formula Editor window that enables you to write your own formulas.

AFL Code Wizard
Opens the AFL Code Wizard - the add-on program that creates trading system AFL code from plain English sentences. See introduction video to AFL Code Wizard.

Tools menu



Database Purify
Database purify tool allows to detect missing/extra quotes, possible splits or invalid OHLC relationship.

Indicator Maintenance
Opens Indicator Maintenance wizard, that helps to clean up unused indicator space

Bar Replay
Opens Bar Replay tool, which allows to replay historical data.

Preferences
Opens Preferences window which allows you to configure the program.

Save Preferences
Saves all the preferences changes (the information is store in *broker.prefs* file).

Plugins
Opens Plugins window. It contains the lists of all loaded plug-in DLLs and can be used for inspecting which plugins are active. It's also possible to unload the plugins.

Customize
Customize tools dialog allows you to define custom tools that can be invoked from Tools menu.

Auto-update quotes
Auto-update quotes option updates historical quotes from the last date present in AmiBroker upto today with use of AmiQuote Downloader. The detailed description on how to use AmiQuote do obtain free quotations can be found in Automatic update of EOD quotes tutorial chapter.

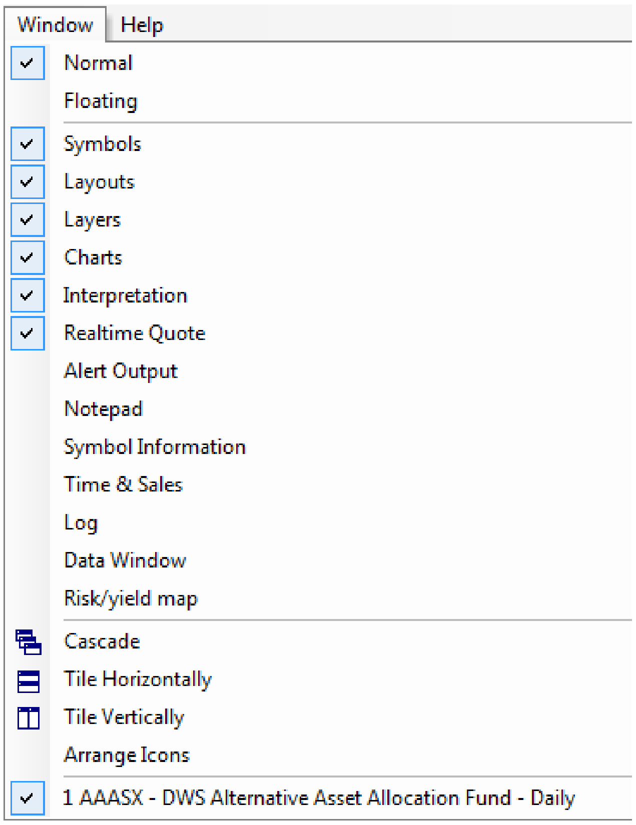
Sharenet Downloader
Launches the script which downloads the quotations from Sharenet (South Africa only).

Export to CSV file
Runs a script that exports the database to the CSV file. Note that you can use Automatic Analysis window to export the quotes way faster than with use of this script.

Cleanup database

Launches the script that allows you to find non-traded stocks in the database. Script automatically scans the database and checks the latest quotation date. If it is old enough, the script will display warning message and lets you decide whether the stock should be deleted or not. Additionally script can generate a list of "old" stocks and save it to the text file. The detailed information is available in: 05-2000 issue of the newsletter.

Window menu



*IMPORTANT NOTE to old version users: **Window -> New** and **Window -> New Linked** options have been moved to File->New->Default Chart and File->New->Linked Chart menus.*

Symbols tab - symbols tree with categories (See: Understanding categories).

Layouts tab - list of available global and local layouts (See: Working with chart sheets and window layouts).

Layers tab - list of chart layers (See: Working with layers).

Charts tab - the window showing the list of chart formulas (See: Working with drag-drop charting interface).

Interpretation
Displays/hides the Interpretation window.

Realtime Quote
Displays/hides the Realtime Quote window. The RT quote window provides real-time streaming quotes and some basic fundamental data. To learn more read: How to use AmiBroker in Real Time mode chapter.

Alert Output
Shows/hides Alert Output window. The window displays texts generated by formula based alert. The detailed information on how to use alerts is available in: Using formula-based alerts part of the Users' Guide.

Notepad
Displays/hides Notepad window, that allows to store free-text notes about particular security. Just type any text and it will be automatically saved / read back as you browse through symbols. Notes are global and are saved in "Notes" subfolder as ordinary text files.

Symbol Information

Shows symbol information window with fundamental data.

Time & Sales

Shows Time and Sales real time window

Log

Shows the log window that displays AFL error messages, run-time errors and _TRACE output

Data window

Shows the data window that displays values of chart indicators

Risk/Yeld map
Displays Risk/Yeld map of all the symbols in the database. Risk/yield map calculates average weekly return (the yield) and standard deviation of the weekly returns (the risk) over at least 12 weeks. It requires at least 60 bars worth of data for every stock. To zoom in - mark the area with the mouse. To zoom-out simply click on the map.

Cascade
Cascades opened chart windows.

Tile Horizontaly
Tiles the opened chart windows horizontally.

Tile Vertically
Tiles the opened chart windows vartically.

Normal
Switches the chart window to "normal" (non-floating) state. More info here.

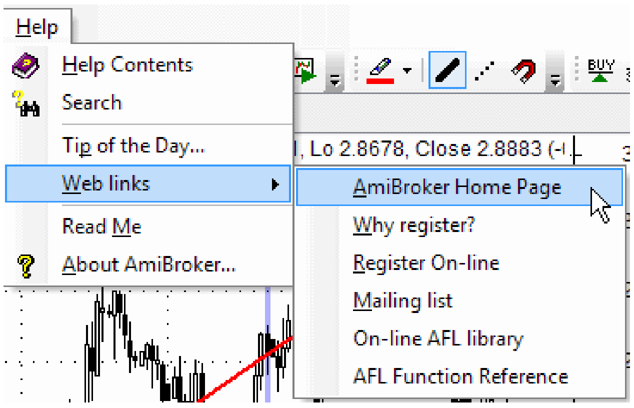
Floating
Switches the chart window to floating state. More info here.

Arange Icons
Allows you to arrange the minimized windows. Arrange icons works only if:

- You created more than two windows (via Window->New or Window->New Linked)
- You have minimized them
- You moved the minimized boxes

Arrange icons option will align the windows nicely at the bottom of the AmiBroker window.

Help menu



Help Contents
Displays the contents of the AmiBroker Users' Guide.

Search
Allows you to search the Users' Guide.

Tip of the day
Shows *Tip of the day* dialog where many useful usage tips are displayed.

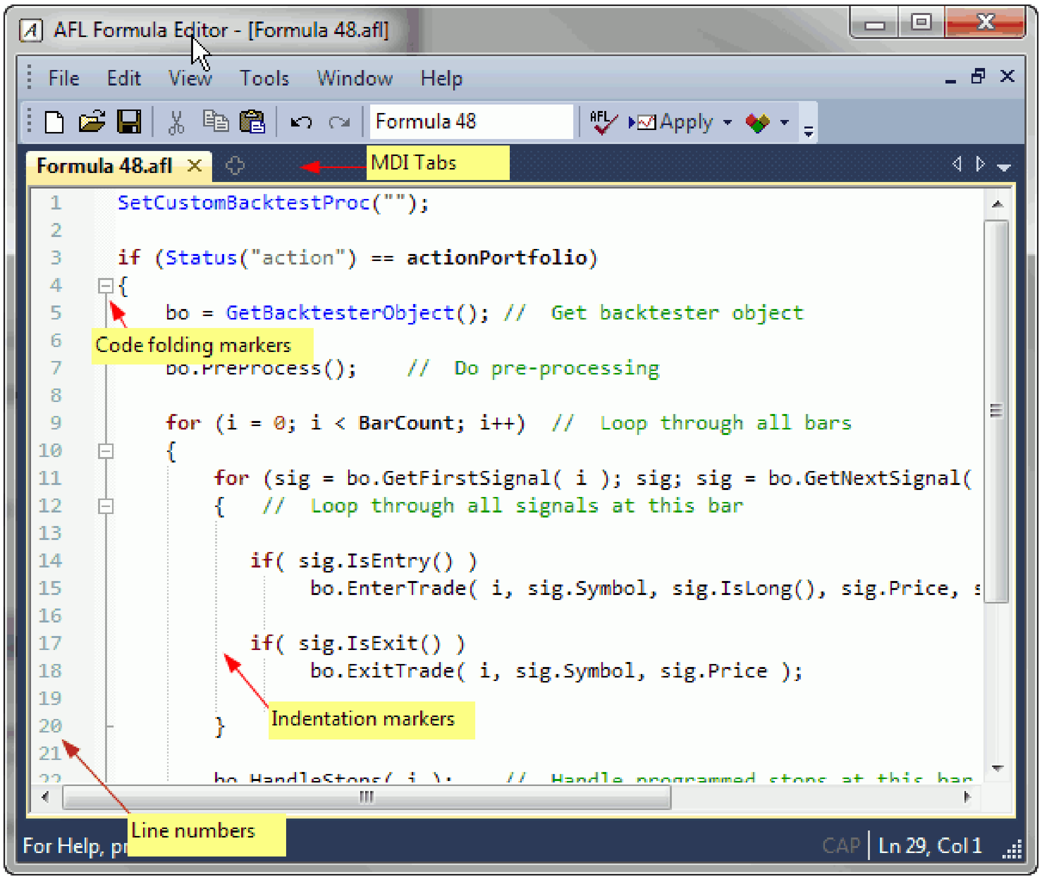
AmiBroker on the web

- link to AmiBroker Home page
- list of benefits for registered users
- secure On-line order form
- AmiBroker Mailing List
- On-line formula library
- On-line AFL function reference

Readme
Displays the contents of Readme file. Please note that all the recent changes in beta releases are reported in Readme.

About AmiBroker
Shows the 'About' window, which contains the information about program version and user details.

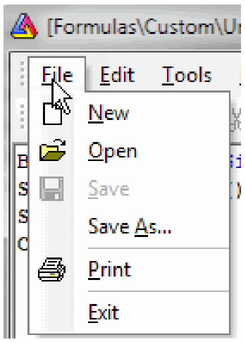
AFL Editor menu



```
SetCustomBacktestProc("");

if (Status("action") == actionPortfolio)
{
    bo = GetBacktesterObject(); //  Get backtester object

    bo.PreProcess();     //  Do pre-processing

    for (i = 0; i < BarCount; i++)  //  Loop through all bars
    {
        for (sig = bo.GetFirstSignal( i ); sig; sig = bo.GetNextSignal(
        {   //  Loop through all signals at this bar

            if( sig.IsEntry() )
                bo.EnterTrade( i, sig.Symbol, sig.IsLong(), sig.Price, s

            if( sig.IsExit() )
                bo.ExitTrade( i, sig.Symbol, sig.Price );

        }

        bo.HandleStops( i );     //  Handle programmed stops at this bar
```

AFL editor features separate menu consisting of the following choices:

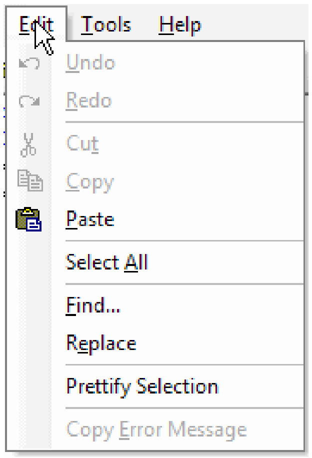
1. File



where

- New - clears the formula editor window
- Open - opens the formula file
- Save - saves the formula under current name
- Save As.. - saves the formula under new name
- Print - prints the formula
- Exit - closes the editor

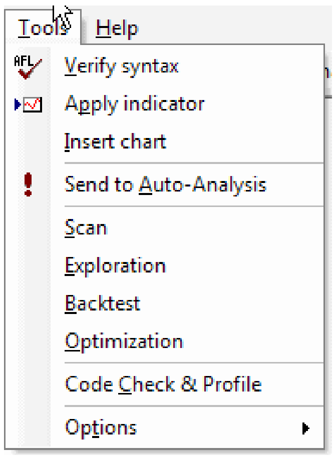
2. Edit



where

- Undo - un-does recent action (multiple-level)
- Redo - re-does recent action (multiple-level)
- Cut - cuts the selection and copies to the clipboard
- Copy - copies the selection to the clipboard
- Paste - pastes current clipboard content in the current cursor position
- Select All - selects entire text in the editor
- Find... - provides access to text search tool
- Copy Error Message - copies current error message displayed in the bottom of the editor window to the clipboard (option is active only when there are any errors displayed after syntax check)

3. Tools

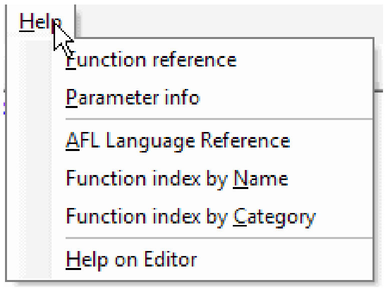


where

- Verify syntax - checks current formula for errors
- Apply indicator - saves the formula and applies current formula as a chart/indicator ONCE
- Insert chart - saves the formula and applies current formula as a chart MANY TIMES (inserts multiple times)
- Send to Auto-Analysis - saves the formula and selects it as current formula in Automatic Analysis window

- Scan - saves the formula and performs Scan in Automatic Analysis window
- Exploration - saves the formula and performs Exploration in Automatic Analysis window
- Backtest - saves the formula and performs Backtest in Automatic Analysis window
- Optimization - saves the formula and performs Optimization in Automatic Analysis window
- Check - saves the formula and performs Check (if given formula references future) in Automatic Analysis window
- Options: Auto-save formula before running analysis - when checked, any click on Scan/Explore/Backtest/Optimize button in Automatic Analysis window triggers automatic save of current formula.

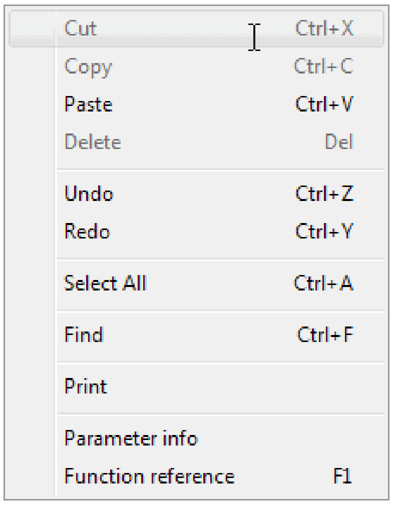
4. Help



where

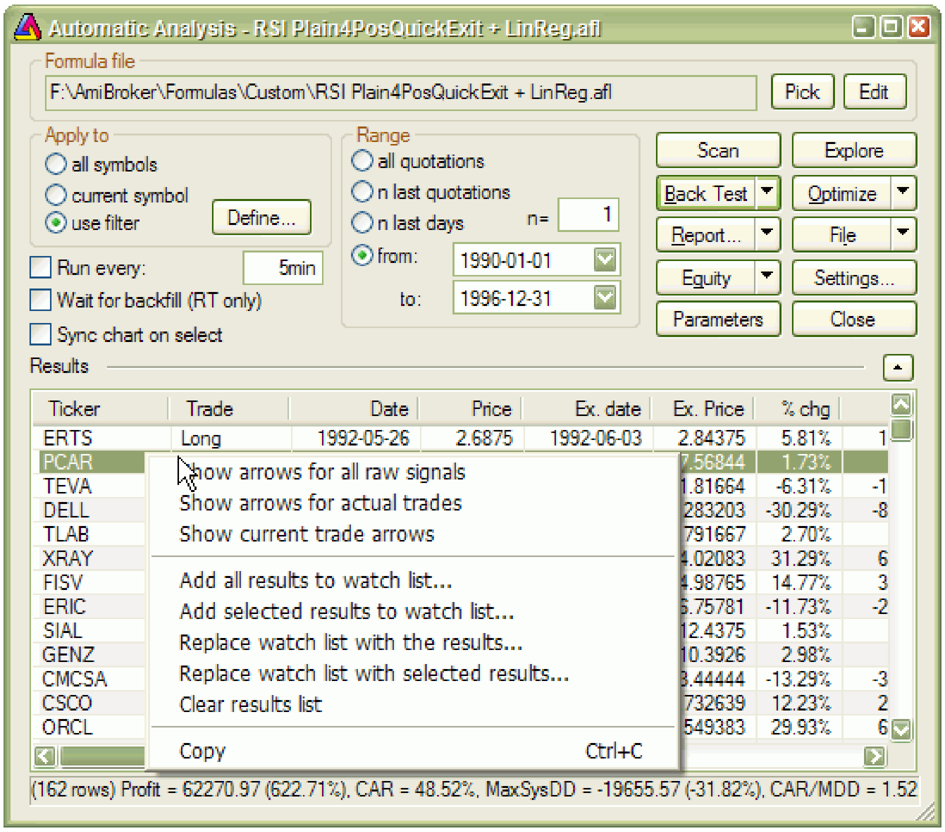
- Function reference - displays reference page for currently highligted AFL function, more on this feature here.
- Parameter info - displays parameter tooltip for currently highlighted AFL function, more on this feature here.
- AFL Language reference - displays language reference page.
- Function index by Name - displays alphabetical list of AFL functions.
- Function index by Category - displays categorized list of AFL functions.
- Help on Editor - displays this help page.

as well as context menu (available via RIGHT click over the formula):



which essentially duplicates choices available from regular menu.

Automatic Analysis result list context menu


Automatic Analysis - RSI Plain4PosQuickExit + LinReg.afl
Formula file
Pick
Edit
Apply to
all symbols
current symbol
use filter
Define...
Range
all quotations
n last quotations
n last days
Scan
Explore
Back Test
Optimize
Report...
File
Equity
Settings...
Parameters
Close
Results
Show arrows for all raw signals
Show arrows for actual trades
Show current trade arrows
Add all results to watch list...
Add selected results to watch list...
Replace watch list with the results...
Replace watch list with selected results...
Clear results list
Copy Ctrl+C

Formula file: F:\AmiBroker\Formulas\Custom\RSI Plain4PosQuickExit + LinReg.afl

Apply to: [✓ use filter]

Range: [✓ from:] 1990-01-01 to: 1996-12-31, n= 1

Run every: 5min

Wait for backfill (RT only)

Sync chart on select

Ticker	Trade	Date	Price	Ex. date	Ex. Price	% chg	
ERTS	Long	1992-05-26	2.6875	1992-06-03	2.84375	5.81%	1
PCAR					7.56844	1.73%	
TEVA					1.81664	-6.31%	-1
DELL					283203	-30.29%	-8
TLAB					791667	2.70%	
XRAY					4.02083	31.29%	6
FISV					4.98765	14.77%	3
ERIC					5.75781	-11.73%	-2
SIAL					12.4375	1.53%	
GENZ					10.3926	2.98%	
CMCSA					3.44444	-13.29%	-3
CSCO					732639	12.23%	2
ORCL					549383	29.93%	6

(162 rows) Profit = 62270.97 (622.71%), CAR = 48.52%, MaxSysDD = -19655.57 (-31.82%), CAR/MDD = 1.52

This menu shows up when you click with RIGHT mouse button over Automatic Analysis result list.

Available choices:

- Show arrows for all raw signals - show buy/sell/short/cover arrows for all raw (unfiltered) signals. If your formula is for example

  buy = C > MA(C, 10);

  you will get a buy (solid green) arrow for all bars where close was above 10-bar moving average
- Show arrows for actual trades - show arrows only on trade entry/exit bars. This shows arrows for ALL TRADES. If your formula is for example

  buy = C > MA(C, 10);

  you will get a buy (solid green) arrow only for the very first bar when close crossed above moving average and trade was initiated, and you won't get any subsequent buy arrows until a matching sell (trade exit) occurs.

  Note that trade arrows represent all possible trades taken. Given trade may not be taken by backtester if there are insufficient funds to enter it.
- Show current trade arrows - show entry/exit arrows for selected trade only. This displays the arrows for currently selected trade (from the result list). It represents trade actually taken.

- Add all results to watch list - adds all symbols from the result list to the watch list of your choice. More on this here
- Add selected results to watch list - adds symbols from selected rows to the watch list of your choice. More on this here

- Replace watch list with all results - empties the watch list and then adds all symbols from the result list to the watch list of your choice. More on this here
- Replace watch list with selected results - empties the watch list and then adds symbols from selected rows to the watch list of your choice. More on this here
- Clear result list - removes all rows from the result list

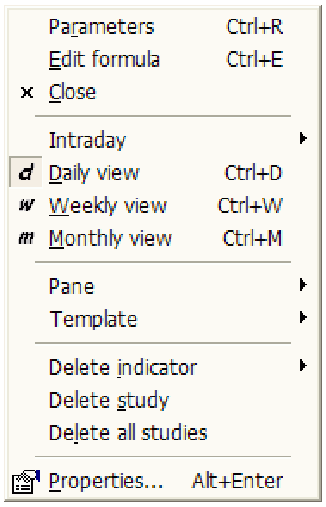
- Copy - copies result list to the Windows clipboard, so you can paste it to some other application, like Excel for example

IMPORTANT NOTES:

1. Buy arrow is solid green, Sell arrow is solid red, Short arrow is hollow red, Cover arrow is hollow green

2. Arrows are shown only on the charts that have "Show arrows" property turned ON.

Chart context menu

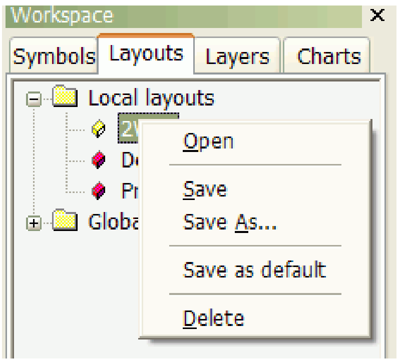


This context menu shows up when you click with RIGHT mouse button over chart pane.

Available options:

- **Parameters...** - brings up Parameters dialog allowing you to modify parameters of indicators, as well as colors, styles, scaling and axes settings
- **Edit Formula...** - brings up Formula Editor allowing you to view/modify the AFL code of indicator
- **Close** - closes chart pane
- **Intraday ...** - allows you to switch viewing time frame to one of available intraday intervals
- **Daily view** - switches viewing interval to daily
- **Weekly view** - switches viewing interval to weekly
- **Monthly view** - switches viewing interval to monthly
- **Pane**
    - **Close** - closes chart pane
    - **Arrange all** - arranges panes to equal height
    - **Move up** - moves selected chart pane up (switches pane vertical order)
    - **Move down** - moves selected chart pane down (switches pane vertical order)
    - **Maximize** - maximizes selected pane so it fills entire screen
    - **Restore** - restores selected pane to previous size
- **Template**
    - **Load...** - loads single window chart template from the selected file (more on templates and layouts here)
    - **Save...** - saves single window chart template to the selected file
    - **Load default** - loads default single window template
    - **Save as default** - saves current single window setup as default template
- **Delete indicator** - deletes one of drag-and-drop indicator sections found in the code
- **Delete study** - deletes selected manually drawn study (like trend line, Fibonacci, Gann...) - more on this here
- **Delete All studies** - deletes all manually drawn studies (like trend line, Fibonacci, Gann...)
- **Properties** - displays properties (coordinates, colors, etc) of manually drawn study (like trend line, Fibonacci, Gann...) more on this here and here.

Layouts context menu



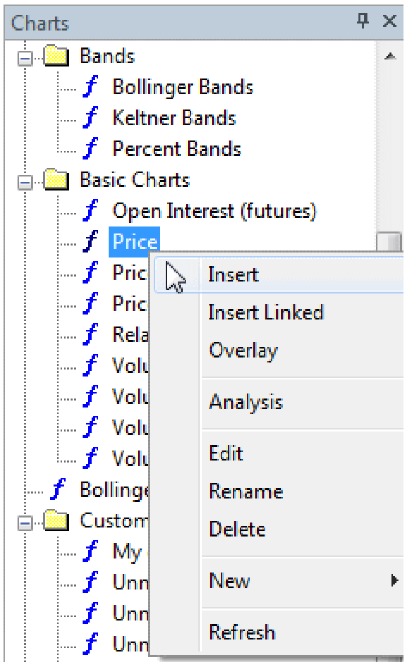
Layouts context menu shows up when you click with RIGHT mouse button over layout in the **Workspace** window, **Layouts** tab.

Available choices:

- Open - loads selected layout
- Save - saves current window layout under current name
- Save As... - save current window layout under new name
- Save as default - save current window layout as default (startup) layout for given database
- Delete - delete selected layout

To learn more about Layouts please check Tutorial: Chart sheets and layouts

Formula (chart) context menu



Formula (chart) context menu shows up when you click with RIGHT mouse button over formula listed in the **Charts** tab of **Workspace** pane (see picture on the left)

Available choices:

- **Insert** - inserts selected indicator into new chart pane.

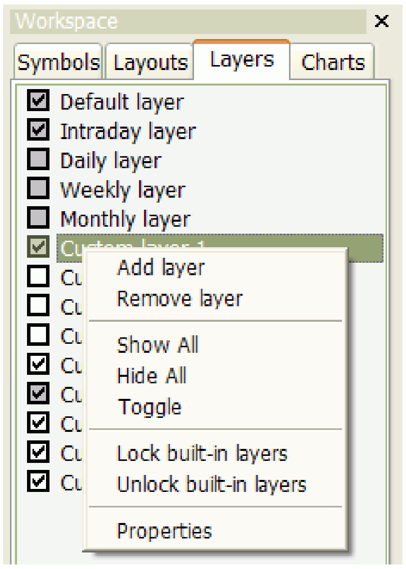
  Insert command internally creates a copy of the original formula file and places such copy into hidden drag-drop folder so original formula will not be affected by subsequent editing or overlaying other indicators onto it.

  Double clicking on formula name is equivalent with choosing Insert command from the menu.
- **Insert Linked** - inserts selected indicator into new chart pane directly (i.e. linked to original).

  Insert Linked command does not create any copy of the formula. Instead it creates new chart pane that directly links to original formula. This way subsequent editing and/or overlaying other indicators will modify the original
- **Overlay** - overlay selected indicator onto selected chart pane

  Overlay command internally appends additional code to the formula used by the chart pane. If given chart pane was created usign Insert Linked, it will modify original (linked) formula.
- **Analysis** - show up Automatic Analysis window and pick selected formula
- **Edit** - open Formula Editor window to edit selected formula
- **Rename** - rename currently selected formula file
- **Delete** - delete currently selected formula file
- **New**
  - ♦ **Formula** - creates new formula file in currently selected folder
  - ♦ **Folder** - creates new subfolder under currently selected folder
- **Refresh** - re-reads Formula directory and re-display formula tree

Layers context menu



Layers context menu shows up when you click with RIGHT mouse button over layer list in the **Layers** tab of **Workspace** pane.

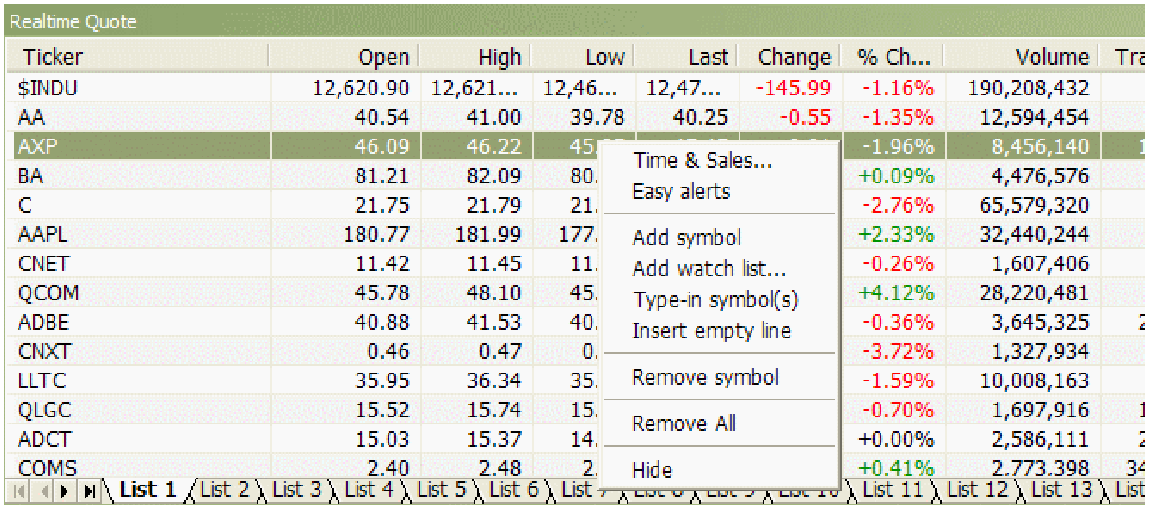
Available options:

- **Add layer** - adds new layer
- **Remove layer** - removes selected layer.

  Please note that you can not remove first 5 (built-in) layers
- **Show All** - shows all not locked layers
- **Hide All** - hides all not locked layers
- **Toggle** - toggles visibility of not locked layers
- **Lock built-in layers -** allows you to lock 5 first (built-in) layers. When layer is locked its visibility changes automatically when interval changes and you can not show/hide it manually.
- **Unlock built-in layers -** allows you to unlock 5 first (built-in) layers. Once layer is unlocked its visibility does not change automatically when interval changes and you can show/hide it manually.
- **Properties -**
  this launches properties box that allows you to rename layer and decide if given layer should or should not be locked to interval displayed. If you mark "Lock visibility to interval" box the layer will show/hide automatically depending on what interval is
  currently displayed. You can define visibility for **each** layer using "Interval" combo and "Show/hide automatically" buttons. Note that there is a *separate* visibility setting for EACH interval. The layer properties box ALWAYS shows "monthly" interval at start but this is just a startup condition you just switch to particular interval
  and modify visibility. To setup locked layer completely you have to set visibility for **every layer listed** in the "Interval" combo-box. Simply select the interval and choose if layer should be shown or hidden for this interval, select next interval and again choose show or hide, select next and so on...until you define visibility for all intervals.

More information about what layers are and how to use them is in the Tutorial: Using Layers section of the guide.

Real-time quote context menu



Ticker	Open	High	Low	Last	Change	% Ch...	Volume	Tra
$INDU	12,620.90	12,621...	12,46...	12,47...	-145.99	-1.16%	190,208,432	
AA	40.54	41.00	39.78	40.25	-0.55	-1.35%	12,594,454	
AXP	46.09	46.22	45.			-1.96%	8,456,140	1
BA	81.21	82.09	80.			+0.09%	4,476,576	
C	21.75	21.79	21.			-2.76%	65,579,320	
AAPL	180.77	181.99	177.			+2.33%	32,440,244	
CNET	11.42	11.45	11.			-0.26%	1,607,406	
QCOM	45.78	48.10	45.			+4.12%	28,220,481	
ADBE	40.88	41.53	40.			-0.36%	3,645,325	2
CNXT	0.46	0.47	0.			-3.72%	1,327,934	
LLTC	35.95	36.34	35.			-1.59%	10,008,163	
QLGC	15.52	15.74	15.			-0.70%	1,697,916	1
ADCT	15.03	15.37	14.			+0.00%	2,586,111	2
COMS	2.40	2.48	2.			+0.41%	2,773,398	34

Time & Sales
Opens Time & Sales window that provides information about every bid, ask and trade streaming from the market.

Easy Alerts
Opens Easy Alerts window that provides way to define realtime alerts executed when bid/ask/last and other fields hit user-defined levels

Add Symbol
Adds current symbol to Real-Time Quote list

Add watch list...
Adds entire watch list to real-time quote window

Type-in symbols
Allows to type the symbols directly as comma-separated list

Insert empty line
Adds empty (separator) line - useful for grouping symbols

Remove Symbol
Removes highlighted line (symbol) from the Real-Time Quote list.

Remove All
Removes all symbols from real-time quote list

Hide
Hides Real-Time Quote list

Keyboard shortcuts

AmiBroker allows complete customization of the user interface, including keyboard shortcuts. To define your own shortcuts use **Tools->Customize** menu, **Keyboard** tab. Read more about it in the Tutorial: User Interface Customization.

Pre-defined keyboard shortcut list follows below, please note that if you used keyboard customization features the list here may not be valid because some of the entries may have been changed to your own.

Keyboard shortcut	**Command**
CTRL+0	VIEW_HOURLY
CTRL+1	VIEW_1MINUTE
CTRL+5	VIEW_5MINUTE
CTRL+6	VIEW_15MINUTE
CTRL+C	EDIT_COPY
CTRL+D	VIEW_DAILY
CTRL+E	CHART_EDITFORMULA
CTRL+H	VIEW_CROSSHAIR
CTRL+I	CHART_MORE_INDICATORS
CTRL+M	VIEW_MONTHLY
CTRL+N	FILE_NEW
CTRL+O	FILE_OPEN
CTRL+P	FILE_PRINT
CTRL+R	CHART_PARAMETERS
CTRL+S	FILE_SAVE
CTRL+V	EDIT_PASTE
CTRL+ADD (CTRL+'+')	VIEW_ZOOM_IN
ALT+BACK	EDIT_UNDO
DELETE	EDIT_CLEAR
ALT+DELETE	EDIT_CLEAR_ALL
SHIFT+DELETE	EDIT_CUT
END	CHART_SCROLL_END
F1	HELP
SHIFT+F1	CONTEXT_HELP
F12	CHART_RANGE_BEGIN
CTRL+F12	CHART_RANGE_HIDE
SHIFT+F12	CHART_RANGE_END

F3	STOCK_FIND
F4	QUICK_FIND
F5	VIEW_REFRESH_CHARTS
F6	NEXT_PANE
SHIFT+F6	PREV_PANE
HOME	CHART_SCROLL_BEGIN
CTRL+INSERT	EDIT_COPY
SHIFT+INSERT	EDIT_PASTE
ALT+LEFT	SYMBOL_PREV
CTRL+ALT+LEFT	VIEW_GO_PREV
SHIFT+ALT+LEFT	SYMBOL_PREV_TREE
PAGE_DOWN	CHART_SCROLL_PAGE_RIGHT
CTRL+PAGE_DOWN	VIEW_SHEET_NEXT
PAGE_UP	CHART_SCROLL_PAGE_LEFT
CTRL+PAGE_UP	VIEW_SHEET_PREV
ALT+RETURN	CHART_STUDY_PROPERTIES
ALT+RIGHT	SYMBOL_NEXT
CTRL+ALT+RIGHT	VIEW_GO_NEXT
SHIFT+ALT+RIGHT	SYMBOL_NEXT_TREE
CTRL+SUBTRACT (CTRL+'-')	VIEW_ZOOM_OUT
CTRL+W	VIEW_WEEKLY
CTRL+X	EDIT_CUT
CTRL+Z	EDIT_UNDO

Import ASCII

AmiBroker has easy-to-use and flexible quotation import feature. This document describes advanced concepts of AmiBroker ASCII importer. Novice users should start with ASCII Import Wizard.

How does it work?

Quotation data may come from various sources so the format of the ASCII (i.e. text based) file may be much different from one source to another. To handle all those differences AmiBroker uses format definition commands that define the way the text information is interpreted by the ASCII importer. The format definition commands are keywords that begin with a dollar sign '$'. These commands may be embedded in the data file itself or, may be stored in the separate format definition file for multiple use. Storing format definition commands in separate file avoids the need to include the commands in every data file. The default format definition file name is "default.format". This file, all other ".format" files and "import.types" file (described later) should be stored in **\Formats** subdirectory of AmiBroker's current working directory. The defaults are overridden by any commands included (embedded) in the data file itself.

So, when you use the "Import from ASCII" menu, AmiBroker first looks for the format definition stored in "default.format" file and then parses the file you have chosen. If there is no "default.format" file then it uses internal defaults (described below).

You can modify "default.format" file to suit your needs. Moreover using OLE Automation (Win32 version) or ARexx (Amiga) interface you can specify the name of the format definition file which will be used instead of "default.format" file.

Format definition commands

The command keywords begin with a dollar sign '$''. Every line starting with command is interpreted in special way. Here is the list of commands recognized by AmiBroker's built-in importer. Bold letters mark keywords.

Command	**$ADDRESS**	Define company address
Arguments	<string>	address of company
Alias		
Examples	**$ADDRESS "One Microsoft Way"**	

Command	**$AUTOADD**	Switch new ticker add mode
Arguments	<number>	0 - do not add , 1 - add a new stock when non-existing ticker detected (default = 0)
Alias		
Examples	**$AUTOADD 1**	

Command	**$ALLOWNEG**	Allow negative numbers in prices
Arguments	<number>	

		0 - do not allow negative values (default), 1 - allow negative values in prices. This additionally switches off any checking for OHLC relationship so you can import any data into OHLC fields.
		
		when $ALLOWNEG is NOT specified in the ASCII importer definition AmiBroker performs the following range checking and fixup on open, low and high prices
		if(open == 0) open = close;
		if(high < max(open, close)) high = max(open, close);
		if(low == 0) low = min(open, close)

Alias		
Examples	**$ALLOWNEG 1**	

Command	**$ALLOW99SECONDS**	Convert invalid second stamp
Arguments	<onoff>	This flag works ONLY in conjunction with $TICKMODE 1 (see below for details) $ALLOW99SECONDS set to 1 will convert all records with invalid seconds (i.e greater than 59)to 59s. So record stamped 16:29:70 will be treated as 16:29:59
Alias		
Examples	**$ALLOW99SECONDS 1**	

Command	**$APPENDNAME**	append string to the ticker name (useful when you need to join several fields together to make unique stock symbol)
Arguments	<string>	string to append to the ticker symbol
Alias	**$APPENDTICKER**	
Examples		

Command	**$BREAKONERR**	Define on-error behaviour
Arguments	<number>	0 - to continue, 1 - to break import on error (default=0)
Alias		

Examples	**$BREAKONERR 1**

Command	**$CONT**	Define continuous quotations flag
Arguments	<number>	<0 or 1> - continuous quotations flag, this affects $AUTOADD 1 mode - if this is set, newly added stocks are switched to continuous quotation mode (this means enabling candlestick charts for example)
Alias		
Examples	**$CONT 1**	

Command	**$CURRENCY**	Define symbol's currency
Arguments	<string>	Defines currency of symbol
Alias		
Examples	**$CURRENCY EUR** or **$FORMAT NAME, CURRENCY** **$OVERWRITE 1** **$AUTOADD 1**	

Command	**$DATE_DMY**	Define date
Arguments	<number>	The date in Canadian format (DD-MM-YY). If there is no argument given the date is taken from the file name (without an extension)
Alias	**$DATE_CDN**	
Examples	**$DATE_DMY 12-05-99** **$DATE_CDN 12-05-1999**	

Command	**$DATE_MDY**	Define date
Arguments	<number>	The date in US format (MM-DD-YY). If there is no argument given the date is taken from the file name (without an extension)
Alias	**$DATE_USA**	
Examples	**$DATE_MDY 05/12/99** **$DATE_USA 05/12/99**	

Command	**$DATE_YMD**	Define date
Arguments	<number>	The date in International format (YY-MM-DD). If there is no argument given the date is taken from the file name (without an extension)
Alias	**$DATE_INT**	
Examples	**$DATE_INT 99-05-12** **$DATE_CDN 1999.05.12**	

Command	**$DEBUG**	Switch logging (debug) mode
Arguments	<number>	0 - no error logging, 1 - log errors to "import.log" file (default=0)
Alias		
Examples	**$DEBUG 1**	

Command	**$FORMAT**	Define line format (sequence and types of fields)
Arguments	**DATE_MDY**	date in US format: MM-DD-YY (alias: **DATE_USA**)
	DATE_DMY	date in Canadian format: DD-MM-YY (alias: **DATE_CDN**)
	DATE_YMD	date in International format: YY-MM-DD (alias: **DATE_INT**)
	TIME	time in HH:MM:SS or HH:MM or HHMM or HHMMSS format
	NAME	ticker name (alias: **TICKER**)
	ALIAS	symbol alias ($AUTOADD and $OVERWRITE modes only)
	FULLNAME	symbol full name ($AUTOADD and $OVERWRITE modes

	only)
OPEN	open price
HIGH	high price
LOW	low price
CLOSE	close price
ADJCLOSE	split-adjusted close This is provided to read adj. close column from Yahoo. Works **only** in conjunction with CLOSE field. When both CLOSE and ADJCLOSE are present in the ASCII format definition then importer calculates split factor by dividing ADJCLOSE/CLOSE. It then multiples OPEN, HIGH, LOW and CLOSE fields by this factor and divides VOLUME field by this factor. This effectively converts unadjusted prices to split adjusted prices. Split ratio gets locked once ADJCLOSE drops below 0.05.
OPENINT	open interest
VOLUME	volume
VOL1000	volume in thousands shares
VOLMIL	volume in millions shares
VOLFACTOR	volume factor (number of shares in a block) default =1
TURNOVER	turnover
AUX1	AUX1 field (auxilliary data)
AUX2	AUX2 field (auxilliary data)
SKIP	skip (ignore) field
MARKET	specify a field that contains market ID (affects $AUTOADD and

	$OVERWRITE modes only)
GROUP	specify a field that contains group ID (affects $AUTOADD and $OVERWRITE modes only)
WATCHLIST	specify a field that contains watch list number (0-31) (affects $AUTOADD and $OVERWRITE modes only)
INFO	specify a field with additional information (WSE specific: nk, ns, rk, rs, ok, os, zd, bd)
REDUCTION	specify a field with reduction rate in percents (WSE specific)
ICB	(new in 5.60) specify ICB code For example if your file looks as follows: (format is symbol, full name, ICB code) AAN,AARON'S INC,5375 Then to import it usign AmiBroker's import wizard use the following $FORMAT Ticker,FullName,ICB $OVERWRITE 1 $SEPARATOR , $CONT 1 $GROUP 255 $AUTOADD 1 $NOQUOTES 1
GICS	specify GICS code For example if your file looks as follows: (format is symbol, full name, gics sub industry code) AAN,AARON'S

		INC,25504060 Then to import it usign AmiBroker's import wizard use the following $FORMAT Ticker,FullName,GICS $OVERWRITE 1 $SEPARATOR , $CONT 1 $GROUP 255 $AUTOADD 1 $NOQUOTES 1
	INDUSTRY	specify a field that contains industry ID (affects $AUTOADD and $OVERWRITE modes only)
	INDUSTRYNAME	(new in 5.60) specifies a field that contains Industry Name. AmiBroker will check if given industry name already exists and if not, it will create a new Industry and assign imported stock to the industry specified. Also if SECTORNAME is specified, it will assign newly added industry to specified sector. (affects $AUTOADD and $OVERWRITE modes only)
	SECTORNAME	(new in 5.60) specifies a field that contains Sector Name. AmiBroker will check if given sector name already exists and if not, it will create a new Sector. Also if INDUSTRYNAME is specified, it will assign newly added industry to specified sector. (affects $AUTOADD and $OVERWRITE modes

		only)
	APPENDTICKER	specify a field that contains string that should be appended to the ticker name (useful when you need to join several fields together to make unique symbol symbol)
	MARGIN	future contract margin deposit (positive value = dollars, negative value - percent of full value)
	POINTVALUE	future contract point value
	ROUNDLOTSIZE	round lot size (trading unit size)
	TICKSIZE	tick size
	ADVISSUES	number of advancing issues
	ADVVOLUME	volume of advancing issues
	DECISSUES	number of declining issues
	DECVOLUME	volume of declining issues
	UNCISSUES	number of unchanged issues
	UNCVOLUME	volume of unchanged issues
	ADDRESS	street address of company
	CURRENCY	specifies currency of symbol
	WEBID	specifies web ID
	DIV_PAY_DATE **EX_DIV_DATE** **LAST_SPLIT_DATE** **LAST_SPLIT_RATIO** **EPS** **EPS_EST_CUR_YEAR** **EPS_EST_NEXT_YEAR** **EPS_EST_NEXT_QTR** **FORWARD_EPS** **PEG_RATIO** **BOOK_VALUE** (requires	fundamental data fields. For more info read Using Fundamental Data

	SHARES_OUT to be specified as well) **BOOK_VALUE_PER_SHARE** **EBITDA** **PRICE_TO_SALES** (requires **CLOSE** to be specified as well) **PRICE_TO_EARNINGS** (requires **CLOSE** to be specified as well) **PRICE_TO_BV** (requires **CLOSE** to be specified as well) **FORWARD_PE** (requires **CLOSE** to be specified as well) **REVENUE** **SHARES_SHORT** **DIVIDEND** **ONE_YEAR_TARGET** **MARKET_CAP** (requires **CLOSE** to be specified as well - it is used to calculate shares outstanding) **SHARES_FLOAT** **SHARES_OUT** **PROFIT_MARGIN** **OPERATING_MARGIN** **RETURN_ON_ASSETS** **RETURN_ON_EQUITY** **QTRLY_REVENUE_GROWTH** **GROSS_PROFIT** **QTRLY_EARNINGS_GROWTH** **INSIDER_HOLD_PERCENT** **INSTIT_HOLD_PERCENT** **SHARES_SHORT_PREV** **FORWARD_DIV** **OPERATING_CASH_FLOW** **FREE_CASH_FLOW** **BETA**	
Alias		
Examples	**$FORMAT TICKER DATE_MDY OPEN HIGH LOW CLOSE VOLUME** **$FORMAT TICKER, DATE_INT, CLOSE, VOLUME** **$FORMAT SKIP, TICKER, SKIP, SKIP, DATE_INT, OPEN, HIGH, LOW, CLOSE, TURNOVER**	

Command	**$FULLNAME**	Define full symbol name
Arguments	<string>	full symbol name
Alias		
Examples	**$FULLNAME Apple Computer Inc.**	

Command	**$GICS**	Define GICS code (Global Industry Category System)
Arguments	<number>	this affects $AUTOADD 1 and $OVERWRITE 1 modes - if this is specified symbols are assigned to given GICS category
Alias		
Examples	Now you can import GICS symbol-code assignments using ASCII importer. $FORMAT command now supports GICS code and there is $GICS command for single-symbol files. For example if your file looks as follows: (format is symbol, full name, gics sub industry code) AAN,AARON'S INC,25504060 Then to import it usign AmiBroker's import wizard use the following $FORMAT Ticker,FullName,GICS $OVERWRITE 1 $SEPARATOR , $CONT 1 $GROUP 255 $AUTOADD 1 $NOQUOTES 1	

Command	**$GROUP**	Define group ID
Arguments	<number>	this affects $AUTOADD 1 mode - if this is specified, newly added symbols are assigned to group with given number.
Alias		
Examples		

Command	**$HYBRID**	Switch hybrid mode on/off
Arguments	<number>	0 (off) or 1 (on). When this flag is set, you can combine quotations from multiple files - for example one file can contain only open prices and volume and the other file can contain high/low/close data. Useful especially for Warsaw Stock Exchange for combining the data from fixing and later continuous quotations.

Alias		
Examples		

Command	**$INDUSTRY**	Define industry ID
Arguments	<number>	this affects $AUTOADD 1 mode - if this is specified, newly added symbols are assigned to industry with given number.
Alias		
Examples		

Command	**$MARKET**	Define market ID
Arguments	<number>	this affects $AUTOADD 1 mode - if this is specified, newly added symbols are assigned to market with given number.
Alias		
Examples		

Command	**$NAME**	Define ticker name
Arguments	<ticker>	ticker name (symbol) (default = file name without path and extension)
Alias	**$TICKER**	
Examples	**$NAME AAPL** **$TICKER MSFT**	

Command	**$NOQUOTES**	Switch quotation data mode
Arguments	<number>	0 - (default) accept only quotation data (AmiBroker checks for non-zero prices and valid dates) 1 - switch off quotation data checking - this allows importing non-quotation data - for example only ticker and full names
Alias	**$TICKER**	
Examples	**$NAME AAPL** **$TICKER MSFT**	

Command	**$OVERWRITE**	Switch overwrite mode on/off
Arguments	<number>	0 - off, 1 - on. When overwrite mode is on then information provided by GROUP, MARKET, INDUSTRY, FULLNAME fields

		is overwritten for existing symbols (not only for newly added)
Alias		
Examples	**$OVERWRITE 1**	

Command	**$PRICEFACTOR**	Define price factor
Arguments	<number>	the factor by which price data are multiplied (default = 1)
Alias		
Examples	**$PRICEFACTOR 100**	

Command	**$RAWCLOSE2OI**	Put Raw Close price to OI field
Arguments	<number>	0 - off, 1- on. (off by default) - causes that OpenInterest field gets assigned CLOSE (raw close) field value multiplied by 100
Alias		
Examples	**$RAWCLOSE2OI 1**	

Command	**$RECALCSPLITS**	Recalculate splits
Arguments	<number>	0 - off, 1- on. (off by default) causes that splits are recalculated by AmiBroker by the algorithm that tries to construct correct adjusted price, based on inaccurate information provided by Yahoo. Note that Yahoo provides only 2 decimal digits in adj. close field therefore the more adj. close approaches zero due to adjustements the error grows. The option $RECALCSPLITS 1 is intended to address this problem (at least partially). It works as follows: 1. for each bar ratio ADJCLOSE/CLOSE is calculated 2. if the ratio changes in two consecutive bars by more than 10% it means that split happened that bar. True split ratio is guessed by matching true fraction in the format of X/Y, where X and Y =

		1..9, to the change in ratios. 3. Then true split ratio is used to adjust all past bars until new split is detected. Works only in conjunction with ADJCLOSE
Alias		
Examples	**$RECALCSPLITS 1**	

Command	**$RECALCVOL**	Switch automatic index volume recalculation
Arguments	<number>	0 - off, 1 - on (base index only), 2 - on (all indexes). When this is on AmiBroker calculates volumes for indexes based on assignments to markets and base indexes defined in Categories window
Alias		
Examples	**$RECALCVOL 2**	

Command	**$RECALCAD**	Switch automatic advance/decline composite recalculation
Arguments	<number>	0 - off, 1 - on. When this is on AmiBroker calculates numbers and volumes of issues advancing, declining and unchanged based on assignments to markets and base indexes defined in Categories window.
Alias		
Examples	**$RECALCVOL 2**	

Command	**$ROUNDADJ**	Round split adjusted prices to given number of decimaldigits
Arguments	<decimaldigits>	*decimaldigits* - causes split-adjusted prices (see above) to be rounded to 'decimaldigits' precision. By default no rounding is done Works only in conjunction with ADJCLOSE
Alias		
Examples	**$ROUNDADJ 2**	

Command	**$SEPARATOR**	Define field separator character
Arguments	<separator char>	the character used to separate data fields (default = space)
Alias		
Examples	**$SEPARATOR ,** **$SEPARATOR ;**	

Command	**$SKIPLINES**	Define how many lines to skip (ignore)
Arguments	<number>	number of lines to skip (default = 0)
Alias		
Examples	**$SKIPLINES 1**	

Command	**$STRICT**	Switches on/off strict checking if Open, High, Low prices are greater than zero
Arguments	<onoff>	(default = 0)
Alias		
Examples	**$STRICT 1**	

Command	**$TICKMODE**	Switches on/off tick mode $TICKMODE is a special mode of importer that allows to import quotes that haved It makes two assumptions: a) input data should come in the ascending time order (i.e. OLDER records first, L b) input data should consist of entire tick history because importer will DELETE ar Once again: Turning on $TICKMODE 1 will DELETE ANY QUOTES that already exist in the database and then will impor You have been warned. For example data files like this: MOL,0,20050606,162959,16400.0000,16400.0000,16400.0000,16400.0000,2MO Can be imported using the following definition file: $FORMAT Ticker, Skip, Date_YMD, Time, Open, High, Low, Close, Volume

		$SKIPLINES 1 $SEPARATOR , $CONT 1 $GROUP 255 $AUTOADD 1 $DEBUG 1 $TICKMODE 1 Sometimes it happens that input files have invalid timestamps (seconds > 59). For example: MOL,0,20050606,162970,16400.0000,16400.0000,16400.0000,16400.0000,2 Please take a closer look at first line shown in this example it has time:16:29:70 (y So I had to add a special flag to the importer that works around such data errors. It is called $ALLOW99SECONDS 1 and will convert all records with invalid secon So record stamped 16:29:70 will be treated as 16:29:59 Now for tick mode to work with such incorrect records you would need to add two $TICKMODE 1 $ALLOW99SECONDS 1
Arguments	<onoff>	(default = 0)
Alias		
Examples	**$TICKMODE 1**	

Command	**$TIMESHIFT**	Define intraday time shift used during import
Arguments	<number>	number of hours to shift date/time stamps (can be fractional)
Alias		
Examples	**$TIMESHIFT 2** ; will shift 2 hours forward **$TIMESHIFT -11.5** ; will shift 11 and half hour backward	

Command	**$VOLFACTOR**	Define volume factor
Arguments	<number>	

		the factor by which volume data is multiplied (default = 1)
Alias		
Examples	**$VOLFACTOR 10**	

Command	**$WATCHLIST**	Define watch list number
Arguments	<number>	this affects $AUTOADD 1 and $OVERWRITE 1 modes - if this is specified, newly added symbols are added to the watch list with given number.
Alias		
Examples		

Command	**$CLEANSECTORS**	Clean (wipe) existing sector/industry structure
Arguments	<number>	if this is turned on (1), existing sector/ industry structure will be deleted and initialized with Sector 0, 1, 2, 3...63/ Industry 0...255 This command should only be used in conjunction with **SECTORNAME**, **INDUSTRYNAME** $FORMAT fields to allow setting up fresh industry structure
Alias		
Examples	See example below (importing sector/industry structure)	

Command	**$SORTSECTORS**	Sort sector/industry structure
Arguments	<number>	if this is turned on (1), sector/ industry structure will be sorted alphabetically after importing. This command should only be used in conjunction with **SECTORNAME**, **INDUSTRYNAME** $FORMAT fields to allow setting up fresh industry structure
Alias		
Examples	See example below (importing sector/industry structure)	

Command	**$USEONLYLOCALDB**	Switches "Use only local database" option for the symbol

Arguments	<number>	If data is fed by database plugin, using the ASCII importer to add any symbol causes these newly added symbol to have "Use only local database" flag turned on. A new command: $USEONLYLOCALDB 0 allows to turn this off (so newly added symbols have "use only local database" turned off) This flag does NOT affect existing symbols.
Alias		
Examples		

Command	**$WEBID**	Define web ID
Arguments	<string>	web ID
Alias		
Examples	**$WEBID aapl**	

Notes:

- for **DATE_xxx** you can use -, / or \ as day/month/year separators. You can even omit separators at all if only you give a date in a 6 digit (YYMMDD, MMDDYY, DDMMYY) or 8 digit format (YYYYMMDD, MMDDYYYY, DDMMYYYY).
- AmiBroker recognizes decimal as well as true fractions in price data. True fractions must follow the whole value after at least single space. For example you can specify: 5.33 or 5 1/3

AmiBroker is not limited to any kind of fraction, if you wish you can write even: 5 333/999

Comments

You can include comments in both format definition file and the data file(s). Each line starting with ***** (asterisk) or **;** (semicolon) or **#** (hash) is treated as a comment and ignored by the ASCII importer.

Usage examples

What may look complicated from command list will become quite clear after some examples. So I will give you four examples of how to write format definition files. First example will show the definition for CSV (comma separated values) quotes available from Yahoo's finances site. Second example will show definition for

Metastock ASCII file format. Third example shows definition for Omega SuperCharts ASCII file format. And fourth example will show the definition for s-files used by DM BOS (Polish brokerage company).

Yahoo CSV

The data from Yahoo's site looks as follows:

```
Date,Open,High,Low,Close,Volume
1-Feb- 0,104,105,100,100.25,2839600
31-Jan- 0,101,103.875,94.50,103.75,6265000
28-Jan- 0,108.1875,110.875,100.625,101.625,3779900
```

The first line gives us a hint about the meaning of the comma separated fields. First field will hold the date. The remaining fields will hold open, high, low, close prices and volume. Importer should skip the first line and parse all the remaining lines that hold just comma-separated data. Appropriate format definition file would look like this:

```
$FORMAT Date_DMY,Open,High,Low,Close,Volume
$SKIPLINES 1
$SEPARATOR ,
$DEBUG 1
$AUTOADD 1
$BREAKONERR 1
```

$DEBUG switches on error logging to "import.log" file and $BREAKONERR will cause importer to stop after the first error found. $AUTOADD ensures that new ticker will be added to the database if it is missing. Well... you may ask: how does it know the ticker name? The answer is simple: if there is no field which defines the ticker name, the importer takes the file name (without path and extension) as a ticker. So if you are importing file "C:\My data\AAPL.CSV" AmiBroker will use "AAPL" as a ticker name.

Metastock ASCII

The data in Metastock ASCII format looks as follows:

```
<ticker>,<per>,<date>,<high>,<low>,<close>,<vol>
AAP,D,1/17/2000,5483.33,5332.01,5362.3,0
AKS,D,1/17/2000,9868.45,9638.03,9687.62,0
FET,D,1/17/2000,3741.3,3540.2,3570.81,0
```

First field will hold the ticker name, second - time period ("D" means daily data), third - quotation date. The rest will hold high, low, close prices and volume. The importer should then skip the first line and parse all the remaining lines that hold just comma-separated data. Appropriate format definition file would look like this:

```
$FORMAT Ticker,Skip,Date_MDY,High,Low,Close,Volume
$SKIPLINES 1
$SEPARATOR ,
$DEBUG 1
$AUTOADD 1
$BREAKONERR 1
```

Skip in $FORMAT defines a field which should be ignored by the importer.

Omega SuperCharts ASCII

The data in Omega SC ASCII format looks as follows:

```
ticker,date,open,high,low,close,vol
AAP,20000117,5333.01,5483.33,5332.01,5362.3,3433450
```

This format is similar to previous ones, however the date is in YYYYMMDD format without separators between year, month and day part. AmiBroker, however, can handle such dates with ease. Appropriate format definition file would look like this:

```
$FORMAT Name,Date_Int,Open,High,Low,Close,Volume
$SEPARATOR ,
$DEBUG 1
$SKIPLINES 1
$AUTOADD 1
$BREAKONERR 1
```

Skip in $FORMAT defines a field which should be ignored by the importer.

DMBOS S-files

The data in this format looks as follows:

```
0,29-02-00,12:05,MIDWIG,1069.1,,,+1.2,336002000,
0,29-02-00,12:05,NIF,48.6,,,+0.8,1763000,
0,29-02-00,12:05,WIG20,2300.3,,,+1.1,336002000,
0,29-02-00,12:05,WIG,21536.8,,,+0.2,336002000,
0,29-02-00,12:05,WIRR,2732.8,,,+1.6,16373000,
1,29-02-00,12:05,AGORA,144.00,,,+4.7,15802000,
1,29-02-00,12:05,AGROS,40.00,nk,72,+5.0,840000,
1,29-02-00,12:05,AMERBANK,28.00,,,+3.7,22000,
1,29-02-00,12:05,AMICA,41.50,nk,99,+2.2,564000,
```

This format is a little bit more complicated. For us useful fields are: 2nd - date, 4th - ticker, 5th - close price, 9th - the turnover value (close * volume). The remaining fields holds other information that is not useful for us. Appropriate format definition file would look like this:

```
$FORMAT Skip,Date_DMY,Skip,Name,Close,Skip,Skip,Skip,Turnover
$SEPARATOR ,
$DEBUG 1
```

Importing Sector/Industry structure

Let's assume we have a text file with Stock tickers, Full names, Sector name and industry name listed line by line, as follows:

"DDD","3D Systems Corporation","Technology","Computer Software: Prepackaged Software"
"MMM","3M Company","Health Care","Medical/Dental Instruments"
"SVN","7 Days Group Holdings Limited","Consumer Services","Hotels/Resorts"
"AHC","A.H. Belo Corporation","Consumer Services","Newspapers/Magazines"
"AIR","AAR Corp.","Capital Goods","Aerospace"

```
"AAN","Aaron's, Inc.","Technology","Diversified Commercial Services"
"ABB","ABB Ltd","Consumer Durables","Electrical Products"
```

To import such file we use the following format definition:

```
$FORMAT Ticker, FullName,SectorName,IndustryName
$SEPARATOR ,
$AUTOADD 1
$NOQUOTES 1
$OVERWRITE 1
$CLEANSECTORS 1
$SORTSECTORS 1
```

$NOQUOTES 1 tells the importer that we will be importing non-quotation data. $AUTOADD 1/$OVERWRITE 1 is required to automatically add new symbols and overwrite existing symbol information. $CLEANSECTORS 1 wipes existing stock/industry structure prior to importing and $SORTSECTORS 1 - sorts sectors/industries after importing so they will be listed in alphabetical order in the Symbol window. $FORMAT command just specifies the order and types of field to import

AmiBroker will read such ASCII file one-by one, then it will check whenever given sector name/industry name already exists, if not - it will create new sector/industry. Then it will assign given symbol to specified sector/industry.

The result will be a database with new sector/industry structure being set up and symbols assigned to proper sectors and industries.

Described functionality is used to implement Tools->Update US symbol list and categories tool.

Default behaviour

When importing ASCII files, AmiBroker attempts to open "default.format" file (in the AmiBroker's directory) to obtain the format definition. If such file is missing the following default format is applied:

```
$FORMAT DATE_USA, OPEN, HIGH, LOW, CLOSE, VOLUME
$SEPARATOR
```

This means that by default ASCII importer will use space character as a separator and will parse the following fields: date, open, high, low, close, volume. The file name (without path and extension) will be used as a ticker name. All other import parameters ($DEBUG,$AUTOADD, etc.) are set to zero.

User-definable file types and formats

Now AmiBroker can use not only default.format definition file but also other user-specified files. File types, filters and format definition files are specified in **import.types** file (example is included in the update package). Now user can prepare/modify **import.types** file with the description of supported ASCII formats and filters to use. The format of **import.types** file is:

```
      <Descriptive name>|<File filter>|<definition file name>
```

Note vertical line characters between these three fields. Example import.types file looks as follows:

```
      Default ASCII (*.*)|*.*|default.format
      Yahoo's CSV (*.csv)|*.csv|yahoo.format
      Metastock ASCII (*.mst)|*.mst|metastock.format
      Omega SC ASCII (*.txt)|*.txt|omega.format
      S-Files (s*.*)|s*.*|sfile.format
      C-Files (c*.*)|c*.*|cfile.format
      Sharenet DAT (*.dat)|*.dat|dat.format
```

If such file exists you will see your types in the "Files of type" combo-box and when you select one - appropriate filter will be used and after selecting some files and clicking OK - importer will use specified ".format" file.

In that way you can define as many text-based data formats as you like and AmiBroker will be able to "understand" them all.

Ticker aliases

Now each ticker can have an alias assigned, so the AmiBroker's built-in importers can recognize that security by both ticker symbol and alias names. This is useful when you are using two data sources that are using slightly different symbol naming convention or if you want to give the symbols more intuitive name while retaining the ability to use importers without problems.

GICS categorisation

GICS is global industry classification standard, see
http://en.wikipedia.org/wiki/Global_Industry_Classification_Standard
for more details on GICS system.

GICS codes are from 2 to 8 digits. Such as 10 for energy sector or 351010 for "Health Care Equipment & supplies" industry.
The codes are fixed even if new classifications are added at some point in the future. It is important to understand that these codes work in hierarchical way.

NOTE: current databases DO NOT have GICS codes assigned to symbols.
As far as I know PremiumData http://www.premiumdata.net/ is planning to release AmiBroker-compatible database with GICS support.

AmiBroker now reads GICS.txt file from its installation folder. It contains GICS categories listed one by one in order of GICS code in the following format
GICS;Name;Description<CRLF>

GICS is numeric code from 2 digits upto 8 digits
Name is GICS category name
Description is GICS category description
These fields must be separated by semicolon
< CRLF> means carriage return/line feed characters (means "new line" - just press ENTER/RETURN key if you are editing with text editor)

There must be only one category per line in GICS.txt file

The default GICS.txt file is supplied already.

ICB categorisation

ICB stands for Industry Classification Benchmark
(http://en.wikipedia.org/wiki/Industry_Classification_Benchmark).

AmiBroker allows also ICB 4-level classification system, but demo database does not have symbols classified according to that standard. You can find ICB classification codes in ICB.txt file inside AmiBroker folder.

NOTE: current databases DO NOT have ICB codes assigned to symbols.
ICB classification for NYSE stocks can be imported from http://www.nyse.com/indexes/nyaindex.csv

AmiBroker now reads ICB.txt file from its installation folder. It contains ICB categories listed one by one in order of ICB code in the following format
ICB;Name<CRLF>

ICB is numeric 4 digit code.
Name is ICB category name
These fields must be separated by semicolon
< CRLF> means carriage return/line feed characters (means "new line" - just press ENTER/RETURN key if you are editing with text editor)

There must be only one category per line in ICB.txt file

The default ICB.txt file is supplied already.

High resolution timestamps (milli- and micro-second)

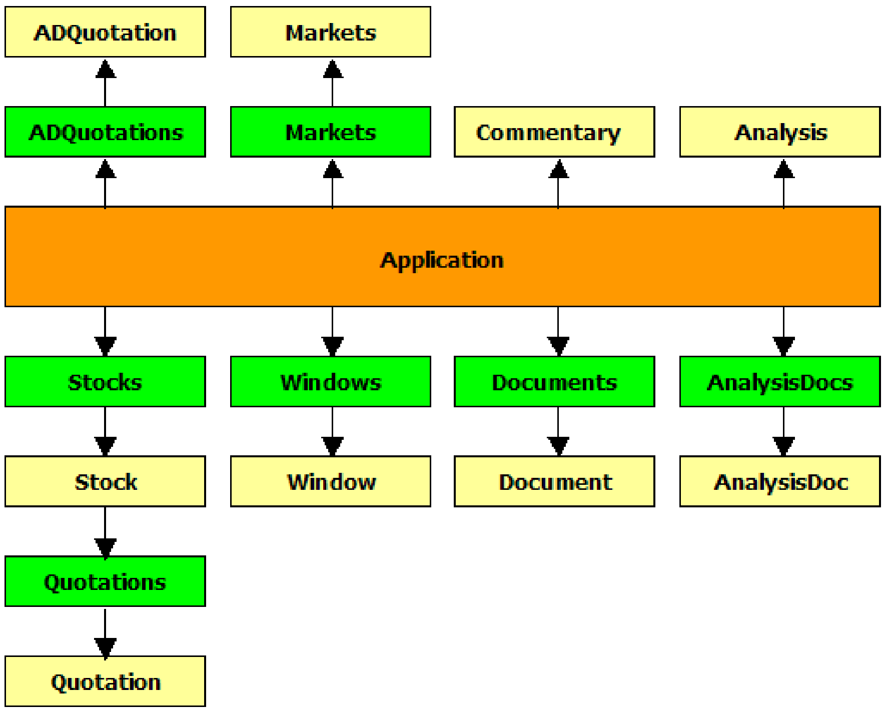
Starting from version 6.14 ASCII importer supports microsecond resolution timestamps (HH:MM:SS.mmmuuu) where mmm - milliseconds 000..999, uuu - microseconds 000..999

AmiBroker's OLE Automation Object Model

Important note about OLE automation:

OLE automation interface is provided to control AmiBroker from the OUTSIDE process (such as windows scripting host). While it is possible to access Broker.Application and underlying objects from AFL formulas you should be very careful NOT to touch any user interface objects (Documents, Document, Windows, Window , Analysis object) from AFL formula because doing so, you will be likely "Sawing Off the Branch You're Sitting On". Especially things like switching chart tabs from currently running chart formula are totally forbidden. Changing user interface objects via OLE from AFL that is currently running within those user interface parts is recipe for disaster. You have been warned.

AmiBroker object model hierarchy. V5.50



Index of objects

- ADQuotation
- ADQuotations
- Analysis[1]
- AnalysisDoc[2]
- AnalysisDocs[2]
- Application
- Window
- Windows
- Commentary
- Document
- Documents
- Market
- Markets
- Quotation
- Quotations
- Stock
- Stocks

[1] - **Analysis** object is obsolete as of 5.50. It is left here for backward compatibility and accesses Old Automatic Analysis window only
[2] - **AnalysisDoc** object and **AnalysisDocs** collection are new objects introduced in v5.50 and allow to control New Analysis window

ADQuotation

Properties:

- **Date** As Date
- **AdvIssues** As Long
- **AdvVolume** As Single
- **DecIssues** As Long
- **DecVolume** As Single
- **UncIssues** As Long
- **UncVolume** As Single

Description:

ADQuotation class keeps one bar of advance/decline information

ADQuotations

Methods:

- Function **Add**(ByVal **Date** As Variant) As Object
- Function **Remove**(ByVal **Date** As Variant) As Boolean

Properties:

- **Item**(ByVal **Date** As Variant) As Object [r/o] [default]
- **Count** As Long

Description:

ADQuotations is a collection of ADQuotation objects

Analysis

This object is obsolete. It is provided only to maintain compatibility with old code. Analysis object always accesses OLD Automatic Analysis.

Properties:

- Property **Filter**(ByVal **nType** As Integer, ByVal **pszCategory** As String) As Long [r/w]

Methods:

- Sub **Backtest**([ByVal **Type** As Variant])
- Sub **ClearFilters**()
- Sub **Edit**([ByVal **bForceReload** As Variant])
- Sub **Explore**()
- Function **Export**(ByVal **pszFileName** As String) As Boolean
- Function **LoadFormula**(ByVal **FileName** As String) As Boolean
- Function **LoadSettings**(ByVal **pszFileName** As String) As Boolean
- Sub **MoveWindow**(ByVal **Left** As Long, ByVal **Top** As Long, ByVal **Width** As Long, ByVal **Height** As Long)

- Sub **Optimize**([ByVal **Type** As Variant])
- Function **Report**(ByVal **pszFileName** As String) As Boolean
- Function **SaveFormula**(ByVal **pszFileName** As String) As Boolean
- Function **SaveSettings**(ByVal **pszFileName** As String) As Boolean
- Sub **Scan**()
- Sub **ShowWindow**(ByVal **nShowCmd** As Long)
- Sub **SortByColumn**(ByVal **iColumn** As Long, ByVal **bAscending** As Integer, ByVal **bMultiMode** As Integer)

Properties:

- **RangeMode** As Long
- **RangeN** As Long
- **RangeFromDate** As Date
- **RangeToDate** As Date
- **ApplyTo** As Long

Description:

Analysis object provides programmatic control of automatic analysis window

Notes:

Analysis.Backtest(Type = 2); - runs backtest
Type parameter can be one of the following values:
0 : portfolio backtest/optimize
1 : individual backtest/optimize
2 : old backtest/optimize

IT IS IMPORTANT TO NOTE THAT FOR BACKWARD COMPATIBILITY REASONS THE DEFAULT BACKTESTER MODE
IS "OLD" BACKTEST. THEREFORE YOU MUST SPECIFY TYPE = 0 IF YOU WANT TO GET PORTFOLIO BACKTEST.

Analysis.Optimize(Type = 2); - runs optimization
Type parameter can be one of the following values:
0 : portfolio backtest/optimize
1 : individual backtest/optimize
2 : old backtest/optimize
3 : walk-forward test (AmiBroker version 5.11.0 or higher)

Analysis.Report(FileName: String) - saves report to the file or displays it if FileName = ""

Analysis.ApplyTo - defines apply to mode: 0 - all stocks, 1 - current stock, 2 - use filter
Analysis.RangeMode - defines range mode: 0 - all quotes, 1 - n last quotes, 2 - n last days, 3 - from-to date
Analysis.RangeN - defines N (number of bars/days to backtest)
Analysis.RangeFromDate - defines "From" date
Analysis.RangeToDate - defines "To" date
Analysis.Filter(nType: short, Category : String) - sets/retrieves filter setting
nType argument defines type of filter 0 - include, 1 - exclude
Category argument defines filter category:
"index", "favorite", "market", "group", "sector", "index", "watchlist"

AnalysisDoc

AnalysisDoc is a new object introduced in version 5.50. It allows to access New Analysis project documents (apx extension) and perform multithreaded scans/explorations/backtests/optimizations in New Analysis window in asynchronous way. Asynchronous means that Run() method only starts the process and returns immediatelly. To wait for completion you must check IsBusy flag periodically (such as every second) in your own code.

Properties:

- ♦ Property **IsBusy** As Boolean [r]

Methods:

- ♦ Sub **Close**()
- ♦ Function **Export**(ByVal **pszFileName** As String, [ByVal **WhatToExport** As Variant]) As Long
- ♦ Function **Run**(ByVal **Action** As Long) As Long
- ♦ Sub **Abort**() (new in 6.20)

Description:

AnalysisDoc object provides programmatic control of New Analysis document/window.

IsBusy property allows to check whenever Analysis window is busy doing analysis. You must check this flag periodically if you want to wait for completion. Take care NOT to call this too often as it will decrease performance. For best results check it every one second. Also you need to check this flag if you are not sure whenever Analysis window is busy before trying to call Export() or Run(), otherwise these calls would fail if analysis is in progress.

Close( ) method closes Analysis document/window. If there is any operation in progress it will be terminated. To prevent premature termination, check IsBusy property.

Export(pszFileName, whatToExport) method allows to export analysis result list to either .HTML or .CSV file. Returns 1 on success (successfull export) or 0 on failure (for example if analysis window is busy). WhatToExport decides what data should be exported:
whatToExport = 0 - exports result list (the default behavior when this parameter is not provided), whatToExport = 1 - exports walkforward tab.

Run(Action) method allows to run asynchronously scan/explorations/backtest/optimizations.
Action parameter can be one of the following values:
0 : Scan
1 : Exploration
2 : Portfolio Backtest
3 : Individual Backtest
4 : Portfolio Optimization
5 : Individual Optimization (supported starting from v5.69)
6 : Walk Forward Test

It is important to understand that Run method just starts the process and returns immediatelly. It does NOT wait for completion.
To wait for completion you need to query IsBusy flag periodically (such as every one second).

Run() returns 1 on success (successfully starting process) or 0 on failure (for example if analysis window is busy)

The procedure to run automated backtest involves opening previously saved Analysis project (it includes all settings that are necessary to perform any action), call Run() and wait for completion.

Since currently you can have multiple analysis projects running, there is an AnalysisDocs collection that represents all open Analysis documents and allow you to open previously saved files (that contain formula, settings and everything needed to run).

New AnalysisDoc object does not allow you to read/write settings for the purpose - you are not supposed to manipulate UI while new Analysis window is running. Correct way of using New Analysis window is to open existing project file and run. If you want to modify the settings, you should write/modify existing project file. The analysis project file (.apx extension) is human-readable self-explanatory XML-format file that can be written/edited/modified from any language / any text editor.

The following JScript example
a) opens analysis project from C:\Analysis1.apx file
b) starts backtest (asynchronously)
c) waits for completion
d) exports results
e) closes analysis document

```
AB = new ActiveXObject( "Broker.Application" ); // creates AmiBroker
object

try
{
    NewA = AB.AnalysisDocs.Open( "C:\\analysis1.apx" ); // opens
previously saved analysis project file
     // NewA represents the instance of New Analysis document/window

    if ( NewA )
    {
         NewA.Run( 2 ); // start backtest asynchronously

         while ( NewA.IsBusy ) WScript.Sleep( 500 ); // check IsBusy
every 0.5 second

         NewA.Export( "test.html" ); // export result list to HTML
file

         WScript.echo( "Completed" );

         NewA.Close(); // close new Analysis
     }
}
catch ( err )
{
     WScript.echo( "Exception: " + err.message ); // display error
that may occur
}
```

Abort( ) method will abort any running Analysis scan/exploration/backtest

AnalysisDocs

AnalysisDocs is a new object introduced in version 5.50. It is a collection of AnalysisDoc objects. Allows to Add new Analysis, Open existing analysis project, and iterate thru analysis objects.

Methods:

- Function **Add**() As Object
- Sub **Close**()
- Function **Open**(ByVal **FileName** As String) As Object

Properties:

- **Item**(ByVal **Index** As Long) As Object [r/o] [default]
- **Count** As Long
- **Application** As Object
- **Parent** As Object

Description:

AnalysisDocs is a collection of AnalysisDoc objects.

Add method creates new Analysis document/window. The method returns **AnalysisDoc** object.

Close method closes all open Analysis documents/windows. If any analysis project is running it will be terminated immediatelly

Open method allows to open existing Analysis project file (.apx). The method returns **AnalysisDoc** object.

Item property allows to access Index-th element of collection. The property returns **AnalysisDoc** object.

Count property gives number of open analysis documents.

Both **Application** and **Parent** properties point to Broker.Application object

For example usage, see **AnalysisDoc** object description.

Application

Methods:

- Function **Import**(ByVal **Type** As Integer, ByVal **FileName** As String, [ByVal **DefFileName** As Variant]) As Long
- Function **LoadDatabase**(ByVal **Path** As String) As Boolean
- Function **LoadLayout**(ByVal **pszFileName** As String) As Boolean
- Sub **LoadWatchlists**()
- Function **Log**(ByVal **Action** As Integer) As Long
- Sub **Quit**()
- Sub **RefreshAll**()
- Sub **SaveDatabase**()
- Function **SaveLayout**(ByVal **pszFileName** As String) As Boolean

Properties:

- **ActiveDocument** As Object
- **Stocks** As Object
- **Version** As String
- **Documents** As Object
- **Markets** As Object
- **DatabasePath** As String
- **Analysis** As Object
- **Commentary** As Object
- **ActiveWindow** As Object
- **Visible** As Integer

Description:

Application object is main OLE automation object for AmiBroker. You have to create it prior to accesing any other objects. To create Application object use the following code:

JScript:

AB = new ActiveXObject("Broker.Application");

VB/VBScript:

AB = CreateObject("Broker.Application")

AFL:

AB = CreateObject("Broker.Application");

Window

Methods:

- Sub **Activate**()
- Sub **Close**()
- Function **ExportImage**(ByVal **FileName** As String, [ByVal **Width** As Variant], [ByVal **Height** As Variant], [ByVal **Depth** As Variant]) As Boolean
- Function **LoadTemplate**(ByVal **lpszFileName** As String) As Boolean
- Function **SaveTemplate**(ByVal **lpszFileName** As String) As Boolean

- ♦ Function **ZoomToRange**(ByVal **From** As Variant, ByVal **To** As Variant) As Boolean

Properties:

- ♦ **SelectedTab** As Long
- ♦ **Document** As Object

Description:

Window object provides programmatic control over charting window.

Windows

Methods:

- ♦ Function **Add**() As Object

Properties:

- ♦ **Item(ByVal Index As Long) As Object** **[r/o] [default]**
- ♦ **Count** As Long

Description:

Windows is a collection of Window objects.

Commentary

Methods:

- ♦ Sub **Apply**()
- ♦ Sub **Close**()
- ♦ Function **LoadFormula**(ByVal **pszFileName** As String) As Boolean
- ♦ Function **Save**(ByVal **pszFileName** As String) As Boolean
- ♦ Function **SaveFormula**(ByVal **pszFileName** As String) As Boolean

Description:

Commentary object gives programmatic control over guru commentary window.

Document

Methods:

- ♦ Sub **Activate**()
- ♦ Sub **Close**()
- ♦ Sub **ShowMessage**(ByVal **Text** As String)

Properties:

- ♦ **Application** As Object
- ♦ **Parent** As Object
- ♦ **Name** As String
- ♦ **ActiveWindow** As Object
- ♦ **Windows** As Object
- ♦ **Interval** As Integer

Description:

Document object represents active document (of 'chart' type). In document-view architecture each document can have multiple windows (views) connected. Name property defines currently selected symbol for the document.

Name is a ticker symbol, Interval is a chart interval in seconds.

Documents

Methods:

- Function **Add**() As Object
- Sub **Close**()
- Function **Open**(ByVal **Ticker** As String) As Object

Properties:

- **Item**(ByVal **Index** As Long) As Object [r/o] [default]
- **Count** As Long
- **Application** As Object
- **Parent** As Object

Description:

Documents is a collection of document objects.

Market

Properties:

- **Name** As String
- **ADQuotations** As Object

Description:

Market represents market category and its related data (i.e. per-market advance/decline information)

Markets

Properties:

- **Item(ByVal Index As Integer) As Object [r/o] [default]**
- **Count** As Integer

Description:

Markets is a collection of Market objects

Quotation

Properties:

- **Date** As Date
- **Close** As Single
- **Open** As Single
- **High** As Single
- **Low** As Single
- **Volume** As Single
- **OpenInt** As Single

Description:

Quotation class represents one bar of price data

Quotations

Methods:

- Function **Add**(ByVal **Date** As Date) As Object
- Function **Adjust**(ByVal **FieldList** As String, ByVal **Multiplier** As Float, ByVal **Offset** as Float, ByVal **DateTime** as Date, ByVal **Before** as Boolean) As Long - *new in version 6.40.2*

  *the function performs price adjustment on quotations, **FieldList** defines which fields will be adjusted, such as "OHLC", value of each field mentioned in field list gets multiplied by multiplier and added offset value: new_value = old_value * **Multiplier** + **Offset***
  *quotes affected will be **Before** (=True) or after specified **DateTime***

- Function **Remove**(ByVal **Item** As Variant) As Boolean
- Function **Retrieve**(ByVal **Count** As Long, ByRef **Date** As Variant, ByRef **Open** As Variant, ByRef **High** As Variant, ByRef **Low** As Variant, ByRef **Close** As Variant, ByRef **Volume** As Variant, ByRef **OpenInt** As Variant) As Long

Properties:

- **Item(ByVal Item As Variant) As Object** **[r/o] [default]**
- **Count** As Long

Description:

Quotations is a collection of Quotation objects. It represents all quotations available for given symbol. Quotations collection is available as a property of Stock object.

Stock

Properties:

- **Ticker** As String
- **Quotations** As Object
- **FullName** As String
- **Index** As Boolean
- **Favourite** As Boolean
- **Continuous** As Boolean

- **MarketID** As Long
- **GroupID** As Long
- **Beta** As Single
- **SharesOut** As Single
- **BookValuePerShare** As Single
- **SharesFloat** As Single
- **Address** As String
- **WebID** As String
- **Alias** As String
- **IsDirty** As Boolean
- **IndustryID** As Long
- **WatchListBits** As Long
- **DataSource** As Long
- **DataLocalMode** As Long
- **PointValue** As Single
- **MarginDeposit** As Single
- **RoundLotSize** As Single
- **TickSize** As Single
- **WatchListBits2** As Long
- **Currency** As String
- **LastSplitFactor** As String
- **LastSplitDate** As Date
- **DividendPerShare** As Single
- **DividendPayDate** As Date
- **ExDividendDate** As Date
- **PEGRatio** As Single
- **ProfitMargin** As Single
- **OperatingMargin** As Single
- **OneYearTargetPrice** As Single
- **ReturnOnAssets** As Single
- **ReturnOnEquity** As Single
- **QtrlyRevenueGrowth** As Single
- **GrossProfitPerShare** As Single
- **SalesPerShare** As Single
- **EBITDAPerShare** As Single
- **QtrlyEarningsGrowth** As Single
- **InsiderHoldPercent** As Single
- **InstitutionHoldPercent** As Single
- **SharesShort** As Single
- **SharesShortPrevMonth** As Single
- **ForwardDividendPerShare** As Single
- **ForwardEPS** As Single
- **EPS** As Single
- **EPSEstCurrentYear** As Single
- **EPSEstNextYear** As Single
- **EPSEstNextQuarter** As Single
- **OperatingCashFlow** As Single
- **LeveredFreeCashFlow** As Single

Description:

Stock class represents single symbol data. For historical reasons the name of the object is Stock, but it can hold any kind of instrument (including futures, forex, etc).

Stocks

Methods:

- Function **Add**(ByVal **Ticker** As String) As Object
- Function **GetTickerList**(ByVal **nType** As Long) As String
- Function **Remove**(ByVal **Item** As Variant) As Boolean

Properties:

- **Item**(ByVal **Item** As Variant) As Object [r/o] [default]
- **Count** As Long

Description:

Stocks is a collection of Stock objects. It is available as a property of Application object.

Notes:

Stock.WatchListBits (long) - each bit 0..31 represents assignment to one of 32 watch lists to add a stock to nth watch list write (JScript example):
Stock.WatchListBits |= 1 << nth;

Stock.WatchListBits2 (long) - each bit 0..31 represents assignment to one of watch lists numbered from 32..63 to add a stock to nth watch list write (JScript example):
Stock.WatchListBits2 |= 1 << (nth - 32);

Stock.DataSource (0 - default, 1 - local only)
Stock.DataLocalMode (0 - default, 1 - store locally, 2 - don't store locally)

Practical Examples:

Example 1: Running simple backtest

```
AB = new ActiveXObject( "Broker.Application" ); // creates AmiBroker object

try
{
    NewA = AB.AnalysisDocs.Open( "C:\\analysis1.apx" ); // opens previously saved
analysis project file
     // NewA represents the instance of New Analysis document/window

    if ( NewA )
    {
         NewA.Run( 2 ); // start backtest asynchronously

         while ( NewA.IsBusy ) WScript.Sleep( 500 ); // check IsBusy every 0.5
second
```

```
         NewA.Export( "test.html" ); // export result list to HTML file

         WScript.echo( "Completed" );

         NewA.Close(); // close new Analysis
     }
}
catch ( err )
{
     WScript.echo( "Exception: " + err.message ); // display error that may occur
}
```

Example 2: Execute commentary

```
AB = new ActiveXObject("Broker.Application");
AB.Commentary.LoadFormula("C:\\Program Files\\AmiBroker\\AFL\\MACD_c.afl");
AB.Commentary.Apply();
AB.Commentary.Save("Test.txt");
AB.Commentary.SaveFormula("MACDTest.afl");
//AB.Commentary.Close();
```

AmiQuote's OLE Automation Object Model

Index of objects

- Document

AmiQuote is SDI (single document) application therefore there is only one class - Document - creatable using the following code:

JScript:

AB = new ActiveXObject("AmiQuote.Document");

VB/VBScript:

AB = CreateObject("AmiQuote.Document")

AFL:

AB = CreateObject("AmiQuote.Document");

Document

Methods:

- Function **AddSymbols**(ByVal **pszSymbols** As String) As Boolean
- Function **Download**() As Boolean
- Function **GetSymbolsFromAmiBroker**() As Boolean
- Function **Import**() As Boolean
- Sub **MoveWindow**(ByVal **x** As Long, ByVal **y** As Long, ByVal **width** As Long, ByVal **height** As Long)
- Function **Open**(ByVal **pszFileName** As String) As Boolean
- Function **RemoveAllSymbols**() As Boolean
- Function **RemoveSymbols**(ByVal **pszSymbols** As String) As Boolean
- Function **Save**() As Boolean
- Function **SaveAs**(ByVal **pszFileName** As String) As Boolean

Properties:

- **DownloadInProgress** As Boolean
- **ImportInProgress** As Boolean
- **Source** As Long
- **From** As Date
- **To** As Date
- **AutoImport** As Boolean
- **AllSessions** As Boolean
- **Interval** As Long
- **RunEvery** As Long
- **DestinationFolder** As String

Technical analysis guide

Introduction

Technical analysis is the examination of past price movements to forecast future price movements. Technical analysts are sometimes referred to as chartists because they rely almost exclusively on charts for their analysis.

Technical analysis is applicable to stocks, indices, commodities, futures, currencies or any tradable instrument where the price is influenced by the forces of supply and demand. Price refers to any combination of the open, high, low or close for a given security over a specific timeframe. The time frame can be based on intraday, daily, weekly or monthly price data and last a few hours or many years. In addition, some technical analysts include volume or open interest figures with their study of price action.

AmiBroker provides a comprehensive set of technical analysis tools that will be presented in this chapter.

Basic tools

AmiBroker has following basic technical analysis tools:

- Price charts
- Trend lines
- Moving averages
- Fibonacci retracement
- Fibonacci time zones
- Regression channels
- Bollinger bands

Price charts

AmiBroker can display the prices using:

- **line chart**
  this mode is used when current symbol uses price fixing and only close price is available
- **traditional bar chart**
  this mode is used when continuous trading is enabled, but open price is not available (or equals to close price)
- **Japanese Candlesticks**
  this mode is used when continuous trading is enabled with open/close/high/low data

A line chart is the simplest type of chart. One price (close) is plotted for each time period. A single line connects each of these price points. The main strength of this chart type is simplicity.

Bar charts are one of the most popular types of charts used in technical analysis. For each trading day a vertical line is plotted. The top of the vertical line indicates the highest price a security traded at during the day, and the bottom represents the lowest price. The closing price is displayed by the mark on the right side of the bar and opening prices are shown on the left side of the bar.

Developed by the Japanese in the 1600's, candlestick charts are merely bar charts that extenuate the relationship between open, high, low and closing prices. Each candlestick represents one period of data (day-week) and consists of an upper shadow, lower shadow and the body. The upper shadow is the highest price that the stock traded at for the period while the lower shadow represents the lowest price. The candlestick body is black when the close is less than the open or white when the close is greater than the open. The top of the body is the opening price if the candle is black and the candle is referred to as a long black candle. If the candle is white, the top of the body is the closing price and the candle is referred to as a long white candle.

Steven Nison's articles that explain Candlestick charting appeared in the December, 1989 and April, 1990 issues of Futures Magazine. The definitive book on the subject is Japanese Candlestick Charting Techniques also by Steve Nison.

There are many different candlestick formations. Some are considered to be minor formations while others are major. Candlestick charts dramatically illustrate supply/demand concepts defined by classical technical analysis theories.

Major Candlestick Chart Formations:

Gravestone Doji: A doji (open and close are the same) and the high is significantly higher than the open, high and closing prices. This formation typically occurs at the bottom of a trend and signals a bullish reversal.

Dragon-fly Doji: A doji (open and close are the same) and the low is significantly lower than the open, high and closing prices. This formation typically occurs at the top of a trend and signals a bearish reversal.

Abandoned Baby Doji: A doji, which occurs at the bottom of a chart formation with gaps on both sides of the doji.

Harami Cross: This formation signals a market top. It consists of a harami, which is a long black line candlestick which precedes and engulfs a doji with no body.

Engulfing Pattern: A two-candle bullish formation consisting of a small long black line engulfed by the second candle, a long white line.

Evening Star: A bearish pattern usually occurring at a top. The formation consists of three candles. The first is a long white line followed by a star and then a long black line. The star can be either black or white.

Dark Cloud Cover: A two candle formation whereby the first candle is a long white line and the second candle is a long black line whose body is below the center of the first candle. This is a bearish formation.

Trend lines

Technical analysis is built on the assumption that prices trend. Trendlines are an important tool in technical analysis for both trend identification and confirmation. A trendline is a straight line that connects two or more price points and then extends into the future to act as a line of support or resistance. Many of the principles applicable to support and resistance levels can be applied to trendlines as well.

Up Trendline

An up trendline has a positive slope and is formed by connecting two of more low points. The second low must be higher than the first for the line to have a positive slope. Up trendlines act as support and indicate that net-demand (demand less supply) is increasing even as the price rises. A rising price combined with increasing demand is very bullish and shows a strong determination on the part of the buyers. As long as prices remain above the trendline, the uptrend is considered solid and intact. A break below the up trendline indicates that net-demand has weakened and a change in trend could be imminent.

Down Trendline

A down trendline has a negative slope and is formed by connecting two or more high points. The second high must be lower than the first for the line to have a negative slope. Down trendlines act as resistance and indicate that net-supply (supply less demand) is increasing even as the price declines. A declining price combined with increasing supply is very bearish and shows the strong resolve of the sellers. As long as prices remain below the down trendline, the downtrend is considered solid and intact. A break above the down trendline indicates that net-supply is decreasing and a change of trend could be imminent.

Scale Settings

High points and low points appear to line up better for trendlines when prices are displayed using a semi-log scale. This is especially true when long-term trendlines are being drawn or there has been a large change in price. AmiBroker allows to set the scale as arithmetic or logarithmic (semi-log). An arithmetic scale displays incremental values (5,10,15,20,25,30) evenly as they move up the y-axis. A $10 movement in price will look the same from $10 to $20 or from $100 to $110. A semi-log scale displays incremental values in percentage terms as they move up the y-axis. A move from $10 to $20 is a 100% gain and would appear to be a much larger than a move from $100 to $110, which is only a 10% gain.

Please remember however that straight line in the log chart is no longer straight in the linear scale, so trend lines drawn in one scale may look strange in the other scale.

Validation

It takes two or more points to draw a trendline. The more points used to draw the trendline, the more validity attached to the support or resistance level represented by the trendline. It can sometimes be difficult to find more than 2 points from which to construct a trendline. Even though trendlines are an important aspect of technical analysis, it is not always possible to draw trendlines on every price chart. Sometimes the lows or highs just don't match up and it is best not to force the issue. The general rule in technical analysis is that it takes two points to draw a trendline and the third point confirms the validity.

Moving averages

The moving average is one of the most useful, objective and oldest analytical tools around. Some patterns and indicators can be somewhat subjective, where analysts may disagree on if the pattern is truly forming or if there is a deviation that is might be an illusion. The moving average is more of a cut-and-dry approach to analyzing stock charts and predicting performance, and it is one of the few that doesn't require a genius intelligence to interpret..

Moving average is an indicator that shows the average value of a security's price over a period of time.

To find the 50 day Simple Moving Average you would add up the closing prices (but not always more later) from the past 50 days and divide them by 50. And because prices are constantly changing it means the moving average will move as well.

Exponential Moving Average (EMA) - is calculated by applying a percentage of today's closing price to yesterday's moving average value. Use an exponential moving average to place more weight on recent prices. As expected, each new price has a greater impact on the EMA than it has on the SMA. And, each new price changes the moving average only once, not twice.

The most commonly used moving averages are the 15, 20, 30, 45, 50, 100, and 200 day averages. Each moving average provides a different interpretation on what the stock price will do. There really isn't just one "right" time frame. Moving averages with different time spans each tell a different story. The shorter the time span, the more sensitive the moving average will be to price changes. The longer the time span, the less sensitive or the more smoothed the moving average will be. Moving averages are used to emphasize the direction of a trend and smooth out price and volume fluctuations or "noise" that can confuse interpretation.

Different investors use moving averages for different reasons. While some use it as their primary analytic tool others simply use the moving average as confidence builder to back their investment decisions. Here are two other strategies that people use moving averages for:

Filters

Filtering is used to increase your confidence about an indicator. There are no set rules or things to look out for when filtering, just whatever makes you confident enough to invest your money. For example you might want to wait until a security crosses through its moving average and is at least 10% above the average to make sure that it is a true crossover. Remember, setting the percentile too high could result in "missing the boat" and buying the stock at its peak.

Another filter is to wait a day or two after the security crosses over, this can be used to make sure that the rise in the security isn't a fluke or unsustained. Again, the downside is if you wait too long then you could end up missing some big profits.

Crossovers

Using Crossovers isn't quite as easy as filtering. There are several different types of crossover's, but all of them involve two or more moving averages. In a double crossover you are looking for a situation where the shortest MA crosses through the longer one. This is almost always considered to be a buying signal since the longer average is somewhat of a support level for the stock price.

For extra insurance you can use a triple crossover, whereby the shortest moving average must pass through the two higher ones. This is considered to be an even stronger buying indicator.

Regression channels

Linear regression may sound intimidating, but the mathematical concept is a simple one. All this technique does is fit a straight line through a finite number of data points by minimizing the sum of the squared vertical distance between the line and each of the points. In our context, this means that if time is represented by days on the horizontal axis and the closing price
on those days is plotted as dots on the vertical axis (a normal closing price chart), then we try to fit a straight line through those closing-price dots such that the total sum of the squared vertical distance between each closing price and the line are minimized. This would then be our best-fit line.

Raff regression channel Raff Regression Channels show the range prices can be expected to deviate from a Linear Regression trend line. Developed by Gilbert Raff, the regression channel is a line study the plots directly on the price chart. The Regression Channel provides a precise quantitative way to define a price trend and its boundaries. The Regression Channel is constructed by plotting two parallel, equidistant lines above and below a Linear Regression trend line.

The distance between the channel lines to the regression line is the greatest distance that any one high or low price is from the regression line.

Raff Regression Channels contain price movement, with the bottom channel line providing support and the top channel line providing resistance. Prices may extend outside of the channel for a short period of time. However, if prices remain outside the channel for a long period of time, a reversal in trend may be imminent.

Fibonacci Retracement

Fibonacci Retracements/Extensions are displayed by first drawing a trendline between two extreme points. After selecting Fibonacci Retracement tool from **Draw** toolbar, a series of up to nine horizontal lines will be drawn at the Fibonacci levels of 0.0%, 23.6%, 38.2%, 50.0%, 61.8%, 100%, 161.8%, 261.8% and 423.6%. After a significant move (up or down), prices will often rebound and retrace a significant portion of the original move. As the price retraces, support and resistance levels will often occur near the Fibonacci Retracement levels.

Fibonacci retracement/extension tool works in 4 different modes depending on the direction of trend line drawn:

- NE - gives (old-style) retracement in up trend
- SE - gives retracement in down trend
- NW - gives extension in up trend
- SW - gives extension in down trend

A controlling trend line drawn with dotted style can be used to delete Fibonacci retracement study at once using right mouse button menu.

Fibonacci Time Zones

The Fibonacci Time Zones study consists of vertical lines at the Fibonacci intervals of 1, 2, 3, 5, 8, 13, 21, 34, etc. The interpretation of Fibonacci Time Zones involves looking for significant changes in price near the vertical lines.

Bollinger bands

Bollinger Bands are envelopes which surround the price bars on a chart. Bollinger Bands are plotted two standard deviations away from a simple short-term moving average. This is the primary difference between Bollinger Bands and envelopes. Envelopes are plotted a fixed percentage above and below a moving average. Because standard deviation is a measure of volatility, the Bollinger Bands adjust themselves to the market conditions. They widen during volatile market periods and contract during less volatile periods. Bollinger Bands become moving standard deviation bands. Bollinger Bands are displayed with a third line. This is the simple (short-term) moving average line. The time period for this moving average can vary. The default for short-term moving average in AmiBroker is 15 days.

An important thing to keep in mind is that Bollinger Bands do not generate buy and sell signals alone. They should be used with another indicator. RSI, for example, is quite good choice as a companion for Bollinger bands. When price touches one of the bands, it could indicate one of two things. It could indicate a continuation of the trend; or it could indicate a reaction the other way. So Bollinger Bands used by themselves do not provide all of what technicians need to know. Then RSI, which is an excellent indicator with respect to overbought and oversold conditions, comes with help. Generally, when price touches the upper Bollinger Band, and RSI is below 70, we have an indication that the trend will continue. Conversely, when price touches the lower Bollinger Band, and RSI is above 30, we have an indication that the trend should continue. If we run into a situation where price touches the upper Bollinger Band and RSI is above 70 (possibly approaching 80) we have an indication that the trend may reverse itself and move downward. On the other hand, if price touches the lower Bollinger Band and RSI is below 30 (possibly approaching 20) we have an indication that the trend may reverse itself and move upward. Avoid the trap of using several different indicators all working off the same input data. If you're using RSI with the Bollinger Bands, don't use MACD too. They both rely on the same inputs. You might consider using On Balance Volume, or Money Flow. RSI, On Balance Volume, and Money Flow, rely on different inputs.

Indicators

What is an indicator?

An indicator is a mathematical calculation that can be applied to a security's price and/or volume fields. The result is a value that is used to anticipate future changes in prices.

AmiBroker has following indicators built-in:

- ROC
- RSI
- MACD
- CCI
- OBV
- NVI
- MFI
- Accumulation/Distribution
- TRIX
- Chaikin
- Relative Strength
- Ultimate Oscillator
- Stochastic
- TRIN (Arms Index)
- AD-Line (Advance/Decline line)
- Volume At Price histogram (Volume Profile)
- Relative Performance

Accumulation/Distribution

Accumulation/Distribution is a momentum indicator which takes into account changes in price and volume together. The idea is that a change in price coupled with an increase in volume may help to confirm market momentum in the direction of the price move.

Note the similarity of this formula to that of the stochastic; this is basically a stochastic multiplied by volume. This means that if the security closes to its high, the volume multiplier will greater than if the security closes nearer to its low.

If the Accumulation/Distribution indicator is moving up the buyers are driving the price move and the security is being accumulated. A decreasing A/D value implies that the sellers are driving the market and the security is being distributed. If divergence occurs between the Accumulation/Distribution indicator and the price of the security a change in price direction is probable.

The Accumulation/Distribution Line formula is as follows:

$$\frac{(CLOSE - LOW) - (HIGH - CLOSE)}{HIGH - LOW} \times VOLUME + I$$

Where *I* is yesterday's Accumulation/Distribution value.

Advance-Decline line (AD-Line)

Line measuring advances and declines that reflects market breadth. In its simplest form ADLine is a summation over time of the net daily difference between the number of advancing issues and the number of declining issues. AmiBroker uses slightly improved formula which takes into account also number of unchanged issues. The exact AFL formula for AmiBroker's ADLine is:

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

This is a classical indicator which tends to give a good reading of the overall strength of the market. A break in the A/D line usually proceeds a break in prices. Look for non-confirmation and divergence.

See also AFL Function reference: AFL Function: adline()

ADX / Directional Movement Index

The ADX Indicator, otherwise known as Directional Movement Index.

The ADX is a trend following system. The average directional movement index, or ADX, determines the market trend. When used with the up and down directional indicator values, +DI and -DI, the ADX is an exact trading system. The standard interpretation for using the ADX (blue line) is to establish a long position whenever the +DI (green line) crosses above the -DI (red line). You reverse that position, liquidate the long position and establish a short position, when the -DI crosses above the +DI. In addition to the crossover rules, you must also follow the extreme point rule. When a crossover occurs, use the extreme price as the reverse point. For a short position, use the high made during the trading interval of the crossover. Conversely, reverse a long position using the low made during the trading interval of the crossover. You maintain the reverse point, the high or low, as your market entry or exit price even if the +DI and the -DI remain crossed for several trading intervals. This is supposed to keep you from getting whipsawed in the market. For some traders, the most significant use of the ADX is the turning point concept. First, the ADX must be above both DI lines. When the ADX turns lower, the market often reverses the current trend. The ADX serves as a warning for a market about to change direction. The main exception to this rule is a strong bull market during a blow-off stage. The ADX turns lower only to turn higher a few days later. According to the developer of the DMI, you should stop using any trend following system when the ADX is below both DI lines. The market is in a choppy sidewise range with no discernible trend. If you need further explanation, please refer to the author's original work. The book titled New Concepts in Technical Trading Systems by J. Welles Wilder, Jr. explains this indicator and several others.

CCI - Commodity Channel Index

A price momentum indicator developed by Donald R. Lambert - it measures price excursions from the mean price as a statistical variation. The indicator works quite well with commodities, stocks and mutual funds. It keeps trades neutral in a sideways moving market, and helps get in the market when a breakout occurs.

A description of the CCI formula is as follows:

First, Calculate each periods mean. This is the high, plus the low, plus the close, divided by 3.

Second, calculate the n period simple moving average of these means.

Third, from each periods mean price, subtract the n period simple moving average of mean prices.

Fourth, Compute the mean deviation. This is the differences between each period's mean price and the n period simple moving average of those mean prices.

Fifth, Multiply the mean deviation by .015.

Sixth, the mean price, which we calculated in step three, is divided by .015 times the mean deviations from step 5.

Ordinarily, CCI ranges in value from +100 to -100. The rules are to buy and go long when CCI crosses above +100 and close the long when CCI falls back below +100. Conversely, sell short when CCI crosses below -100 and close the short when CCI crosses back above -100.

Chaikin Oscillator

Developed by Marc Chaikin back in the early 1970's when opening prices were eliminated from many newspaper listings making it more difficult to calculate William's OBV. Chaikin substituted the average price [(HIGH+LOW)/2] for William's opening price and created an oscillator using 10-period and 3-period exponential moving averages of the resulting Accumulation/Distribution Line.

The basic premise of the Accumulation/Distribution Line is that the degree of buying or selling pressure can be determined by the location of the close, relative to the high and low for the corresponding period. There is buying pressure when a stock closes in the upper half of a period's range and there is selling pressure when a stock closes in the lower half of the period's trading range.

Bullish Signals

There are two bullish signals that can be generated from the Chaikin Oscillator: positive divergences and centerline crossovers. Because the Chaikin Oscillator is an indicator of an indicator, it is prudent to look for confirmation of a positive divergence, by a bullish moving average crossover for example, before counting this as a bullish signal.

Bearish Signals

In direct contrast to the bullish signals, there are two bearish signals that can be generated from the Chaikin Oscillator: a negative divergence and a bearish centerline crossover. Allow a negative divergence to be confirmed by a bearish centerline crossover, before a bullish signal is rendered.

The Chaikin Oscillator is good for adding momentum to the Accumulation/Distribution Line, but can sometimes add a little too much momentum and be difficult to interpret. The moving averages are both relatively short and will therefore be more sensitive to changes in the Accumulation/Distribution Line. Sensitivity is important, but one must also be able to interpret the indicator.

MACD - Moving Average Convergence/Divergence

This indicator uses three exponential moving averages, a short or fast average, a long or slow average and an exponential average of their difference, the last being used as a signal or trigger line. To fully understand the basics of MACD you must first understand simple moving averages. The Moving Average Convergence/Divergence indicator measures the intensity of public sentiment and is considered by Gerald Appel, its developer, to be a very good indicator signaling market entry points after a sharp decline. This indicator reveal overbought and oversold conditions and generates signals that predict trend or price reversals. It provides a sensitive measurement of the intensity of public sentiment and can be applied to the stock market, to individual stocks or to mutual funds. In some instances, it can provide advance warning of reversals allowing you to buy into weakness and sell into strength.

The Moving Average Convergence/Divergence indicator (MACD) is calculated by subtracting the value of 26-day exponential moving average from a 12-day exponential moving average. A 9-day exponential moving average (the "signal line") is automatically displayed on top of the MACD indicator line.

The basic MACD trading rule is to sell when the MACD falls below its 9-day signal line. Similarly, a buy signal occurs when the MACD rises above its signal line.

Money Flow Index

The Money Flow Index (MFI) attempts to measure the strength of money flowing in and out of a security. It is closely related to the Relative Strength Index (RSI). The difference between the RSI and Money Flow is that where RSI only looks at prices, the Money Flow Index also takes volume into account.

Calculating Money Flow is a bit more difficult than the RSI.
First we need the average price for the day thenwe need the Money Flow:

$$MoneyFlow = Volume \times Average\,\mathrm{Price}$$

Now, to calculate the money flow ratio you need to separate the money flows for a period into positive and negative. If the price was up in a particular day this is considered to be "Positive Money Flow". If the price closed down it is considered to be "Negative Money Flow".

$$MoneyRatio = \frac{Positive\,MoneyFlow}{Negative\,MoneyFlow}$$

It is the Money Flow Ratio which is used to calculate the Money Flow Index.

$$MFI = 100 - \frac{100}{100 + MoneyRatio}$$

The Money Flow ranges from 0 to 100. Just like the RSI, a stock is considered overbought in the 70- 80 range and oversold in the 20-30 range.

The shorter number of days you use, the more volatile the Money Flow is. The default is to use a 14 day average.

The interpretation of the Money Flow Index is as follows:

- Look for divergence/failure swings between the indicator and the price action. If the price trends higher (lower) and the MFI trends lower (higher), then a reversal may be imminent.
- Look for market tops to occur when the MFI is above a specific level (e.g., 80). Look for market bottoms to occur when the MFI is below a specific level (e.g., 20).

Negative Volume Index

This indicator makes a very important assumption. It assumes that the unsophisticated investor follows market trends thus pushing up volume as they jump in on a rising security price. On the other hand, informed buying and selling by those "in the know" occurs on quieter periods reflected by negative volume changes on days of declining volume. This is an excellent bull market trend predictor. This index simply measures the trend of prices during periods when the volume is declining.

The price index is only adjusted on those days during which the volume has decreased from the previous day. If the volume did not change or was positive, the indicator remains unchanged. If the index rises, it means simply that the price of the security has gone up on a day that the volume has dropped. A drop in the index indicates that the price of the security has gone down while the volume declined. (The change in the index is calculated as a percentage change in the price).

This indicator can be compared to its longer period averages to reflect the movement of smart money. If, for example current index readings are above a six-month average, it can very well indicate an up trend for the market or the security.

OBV - On Balance Volume

OBV was created by Joe Granville, the father of OBV analysis. This is a running total of volume that relates price changes and volume and shows accumulation and distribution action.

The classic OBV is calculated by adding today□s total volume to a cumulative total when price closes higher than yesterday□s close and subtracting today□s total volume from the cumulative when the price closes lower than yesterday□s close. If price remains the same, then the OBV is not changed. The actual amount of the price change is irrelevant and only the direction of change is significant for these calculations.

This indicator defines trends by showing underlying strength of price movements over time. A solid price trend is assumed to be accompanied with a stronger volume movement in the same direction. OBV analysis assumes that volume trends lead price trends and that OBV changes generally precede price changes. Look for divergence or non-confirmation between price and volume movements. A stock that is trending in an upward direction and starts to experience higher volume on days of lower closing prices usually indicates an end to the current trend. Look for changes or breakouts in OBV trends. Sell short when the OBV makes a downside breakout and buy long when the on OBV upside breakouts.

Parabolic SAR (Stop-And-Reverse)

Developed by Welles Wilder, creator of RSI and DMI, the Parabolic SAR sets trailing price stops for long or short positions. Also referred to as the stop-and-reversal indicator (SAR stands for "stop and reversal"), Parabolic SAR is more popular for setting stops than for establishing direction or trend. Wilder recommended establishing the trend first, and then trading with Parabolic SAR in the direction of the trend. If the trend is up, buy when the indicator moves below the price. If the trend is down, sell when the indicator moves above the price.

The formula is quite complex, but interpretation is relatively straightforward. The dotted lines below the price establish the trailing stop for a long position and the lines above establish the trailing stop for a short position. At the beginning of the move, the Parabolic SAR will provide a greater cushion between the price and the trailing stop. As the move gets underway, the distance between the price and the indicator will shrink, thus making for a tighter stop-loss as the price moves in a favorable direction.

There are two variables: the step and the maximum step. The higher the step is set, the more sensitive the indicator will be to price changes. If the step is set too high, the indicator will fluctuate above and below the price too often, making interpretation difficult. The maximum step controls the adjustment of the SAR as the price moves. The lower the maximum step is set, the further the trailing stop will be from the price. Wilder recommends setting the step at .02 and the maximum step at .20.

RS - Relative Strength (comparative)

Compares the performance trend of a stock or industry group relative to another stock, group or index. This comparison removes the emotion from the marketplace. Many times a drop in relative strength can indicate a coming drop in actual price of the security. Do not confuse with Wilders□s RSI.

The concept is to identify which stock or market sector is performing the best. Assuming that trends will continue to persist for some time, it is more probable that before a stock price will drop sharply it will first loose relative strength against other stocks. This would indicate a sell prior to such a price drop. An increase in relative strength does not necessarily indicate that the index is heading up, but it does signal a buy alert.

RSI - Relative Strength Index

A technical indicator developed by Welles Wilder to help investors gauge the current strength of a security price relative to its past performance. The RSI is an excellent overbought/oversold indicator that can be used to predict trend reversal points. Do not confuse this index with relative strength in its everyday definition as used in comparing the movement of one security, index or group against the movement of another security, index or group. Developed by J. Welles Wilder, Jr. and first described in his book "New Concepts in Technical Trading Systems", this is a momentum oscillator that measures the velocity of directional price movement.

It compares a security highest highs and lowest lows over a period of time. RSI is based upon the difference between the average of the closing price on up days vs. the average closing price on the down days.

RSI=100-[100/(1+U/D)]

U = average of upward price closes (EMA of gains)
D = average of downward price closes (EMA of losses)

The ratio between up and down closing averages is in fact the makeup of the index. The time period specified determines the volatility of the RSI. For example, a 9-day time period will be more volatile than a 21-day time span. The author (Wilder) uses an n value of 14 days but other values may be used that better fit particular securities. The 9-day and 25-day RSIs have also gained popularity. Because you can vary the number of time periods in the RSI calculation, I suggest that you experiment to find the period that works best for you.

The RSI is a price-following oscillator that ranges between 0 and 100. A popular method of analyzing the RSI is to look for a divergence in which the market index is making a new high, but the RSI is failing to surpass its previous high. This divergence would be an indication of an impending reversal. When the RSI then turns down and falls below its most recent trough, it is said to have completed a failure swing. The failure swing would be considered a confirmation of an impending reversal.

In Mr. Wilder's book, he discusses five uses of the RSI in analyzing commodity charts (these apply to indices as well):

1. Tops and Bottoms: RSI readings above 70 indicate the shares are overbought and are likely to start falling. Readings below 30 indicate the shares are oversold and a rally can be expected. (AmiBroker automatically draws horizontal lines at these levels). The RSI usually forms these tops and bottoms before the underlying price chart.

2. Chart Formations: The RSI often forms chart patterns (such as head and shoulders or rising wedges) that may or may not be visible on the price chart.

3. Failure Swings: This is where the RSI surpasses a previous high (peak) or falls below a recent low (trough).

4. Support and Resistance: The RSI shows, sometimes more clearly than the price chart, levels of support and resistance.

5. Divergence: As discussed above, this occurs when the price makes a new high (or low) that is not confirmed by a new RSI high (or low).

ROC - Price Rate Of Change

This indicator displays the rate-of-change of a security□s price. Change is displayed as a percentage rather than as a ratio.

ROC is calculated by dividing the price change over the last n-periods by the closing price n-periods ago. This gives you percentage that the price has changed in the last n-periods.

When the 10-day ROC line is above the central line, the price is higher today than it was 10 periods ago. When the ROC line is below the central line, the price is lower today than it was 10 days ago. If the ROC line is above the central line, the price is higher than it was 10 days ago. If the ROC line is below the central line but rising, the price is still lower today than it was 10 days ago, but the range is narrowing.

The 12-day ROC is best used as a short to intermediate-term overbought/oversold indicator. The higher the ROC, the more overbought the security; the lower the ROC, the more likely a rally. However, as with all overbought/oversold indicators, it is best to wait for the market to begin to correct (i.e., turn up or down) before placing your trade. A market that appears overbought may remain overbought for some time. In fact, extremely overbought/oversold readings usually imply a continuation of the current trend.

The 12-day ROC tends to be very cyclical, oscillating back and forth in a fairly regular pattern. Often, price changes can be anticipated by studying the previous cycles of the ROC and relating the previous cycles to the current market.

The optimum overbought/oversold levels (e.g., +/-5) will vary depending on the security being analyzed and overall market conditions. In strong bull markets, it is usually beneficial to use higher levels, perhaps +10 and -5.

Stochastic Slow

Stochastic is an oscillator that measures the position of a stock or security compared with its recent trading range indicating overbought or oversold conditions.

It displays current day price at a percentage relative to the security□s trading range (high/low) over the specified period of time.

$$FastStoc = \%K = \frac{\text{(today's close) - (low price in period n)}}{\text{(high price in period n) - (low price in period n)}}$$

In a Slow Stochastic, the highs and lows are averaged over a slowing period. The default is usually 3 for slow and 1 (no slowing) for fast. The line can then be smoothed using an exponential moving average, Weighted, or simple moving average %D. Confirming Buy/sell signals can be read at intersections of the %D with the %K as well.

The Stochastic Oscillator always ranges between 0% and 100%. A reading of 0% shows that the security's close was the lowest price that the security has traded during the preceding x-time periods. A reading of 100% shows that the security's close was the highest price that the security has traded during the preceding x-time periods. When the closing price is near the top of the recent trading range (above 80%), the security is in an overbought condition and may signal for a possible correction. Oversold condition exists at a point below %20. Prices close near the top of the range during uptrends and near the bottom of the range during downtrends.

TRIN - Arms Index

Trading Index, a technical measure of advances and declines within the market. TRIN takes into account the number and volume of issues that advanced in price, and the number and volume of issues that declined in price. This index measures the relative strength of volume associated with advancing stocks against the strength of volume associated with declining stocks.

Exact AFL formula for TRIN is:

```
ArmsIndex = ( AdvIssues() / DecIssues() ) / ( AdvVolume() / DecVolume ) );
```

A TRIN value of 1 indicates that the ratio of up volume to down volume is equal to the ratio of advancing issues to the declining issues and the market is in a neutral condition. A neutral condition simply means that the up volume is equally distributed over the advancing issues and that the down volume is equally distributed over declining issues for the day.

This indicator, although simple in its formulation, requires much study in its application. There are many variations applied to the TRIN. Many analysts use a 10-day moving average of TRIN as an indicator. AmiBroker plots two different averages for TRIN with the default averaging periods of 15 and 45. A reading of less than 1.0 usually indicates a bullish demand while a reading greater than 1 can signify a bearish market condition. It must be kept in mind that the indicator behavior and its reading and interpretation depends on whether the market is in a bullish or bearish phase. The actual time duration of this market phase must also be considered. Do not attempt to make and buy or sell decisions based on movements of this indicator by itself.

See also AFL Function reference: AFL Function: trin()

TRIX - TRIple eXponential

TRI-ple eXponential. TRIX displays the % rate-of-change of a triple exponentially smoothed moving average of the closing price of a security.

TRIX is calculated as a one period rate of change of the third exponential moving average pass of the closing price.

TRIX is designed to filter out insignificant cycles - those smaller than the number of moving averages specified. The TRIX indicator oscillates around a zero line. Trades should be placed when the indicator changes direction.

Ultimate Oscillator

Larry Williams, the designer of the Ultimate Oscillator, wanted to address the problems experienced with most oscillators when used over different lengths of time.
Ultimate oscillator signals are the following: divergence and a breakout in the Oscillator's trend, as well as overbought and oversold levels.
The value of other oscillators can vary greatly depending on the number of time periods used during the calculation. So, the Ultimate Oscillator, uses weighted sums of three oscillators which represent short, intermediate, and long term market cycles (7, 14, & 28-period), and it is plotted as a single line on a vertical scale of 0 to 100.

The three components are based on Williams's definitions of buying and selling "pressure."

A trade should be initiated following a divergence and a breakout in the Ultimate Oscillator's trend.

Signals:

A Buy signal is generated when:
A positive or bullish divergence occurs between the Ultimate oscillator and the price.
The Ultimate falls below 30 and then rises above the previous high established during the divergence (the actual buy signal).

A Sell Signal is offered when:
A negative or bearish divergence occurs between the Ultimate and the price.
The Ultimate rises above 70 and then falls below the previous low established during the divergence (the actual sell signal).

Closing existing positions:
Close long positions when the Ultimate exceeds 70.
Close short positions when the Ultimate goes below 30.

As with most indicators, it is good if these signals are confirmed by other indicators before being acted upon.

VAP - Volume At Price histogram

Volume At Price histogram is also known as "Volume Profile" chart.

To turn it on simply go to **Tools->Preferences** and change Type of the VAP from "NONE" to "Left-side solid area chart, behind" for example

VAP shows total volume of trading that occurred at given price level. VAP is calculated from data bars that are currently visible.

Actual algorithm involves not ONE price but High-Low price RANGE.

AmiBroker DISTRIBUTES equally bar's volume over High-Low range to produce VAP histogram. For example if bar's volume is 10000 and H-L range spans 3 'lines" of VAP histogram than each of 3 lines
involved gets added 10000/3 to produce statistics. This gives much more accurate results than using single price
as some other implementations do.

To turn VAP on/off use: Tools->Preferences->Main chart
You can also add VAP to your own custom charts using PlotVAPOverlay AFL function.

Relative Performance chart

Relative Performance chart compares the rate of price change of two or more tradable instruments. Plot starts with 0% at the very first visible bar and shows percentage change of closing price since that point for every symbol in the list. Relative perfomrance charts are great for comparing dissimimilarly priced issues (for example stocks and indices) since it displays percentage changes, not absolute values. You can easily see which instruments perform better than others and choose best performers for your trading.

You can adjust the list of symbols that are plotted in the Relative Performance chart by clicking with RIGHT mouse button over the chart and choosing "Parameters" item from the context menu. In the Parameters dialog you can enter a comma-separated list of symbols that you want to get the chart for. There is no limit on number of symbols you can enter, but please remember to separate symbols by comma and not using spaces unless symbol itself has them.

AmiBroker Formula Language (AFL)

AmiBroker is equipped with a powerful formula language allowing you to write trading system rules, define your own indicators and custom commentaries. This chapter explains the language, gives you detailed reference of built-in analysis functions and shows how to use AFL-tools such as automatic analyzer and formula editor .

See also: Tutorial: Understanding how AFL works

AFL Reference Manual

Introduction

AFL is a special programming language used to define and create custom indicators, scans, explorations, back-tests and guru commentaries.

Basics

Lexical elements

This chapter describes the different categories of word-like units (tokens) recognized by the AFL language interpreter.

Whitespace

Whitespace is the collective name given to spaces (blanks), tabs, new line characters and comments. Whitespace can serve to indicate where tokens start and end, but beyond this function, any surplus whitespace is discarded.

Comments

Comments are pieces of text used to annotate a program. Comments are for the programmer's use only; they are stripped from the source code before parsing. The are two ways to delineate comments: C-like comments and C++ like comments. A C-like comment is any sequence of characters placed after the symbol pair /*. The comment terminates at the first occurrence of the pair */ following the initial /*. The entire sequence, including the four comment-delimiter symbols, is replaced by one space. A C++ like comments are single-line comments that start by using two adjacent slashes (//) in any position within the line and extend until the next new line.

AFL does not allow nested comments.

Tokens

AFL recognizes five classes of tokens:

- identifiers
- constants
- string-literals
- operators
- punctuators (also known as separators)

Identifiers are arbitrary names of any length given to functions and variables. Identifiers can contain the letters (a-z, A-Z), the underscore character ("_"), and the digits (0-9). The first character must be a letter.
AFL identifiers are NOT case sensitive.

Constants are tokens representing fixed numeric or character values. Numeric constants consist of decimal integer and optionally: decimal point and decimal fraction part. Negative numeric constants have unary minus (-) prefixed.
String constants, also known as *string literals*, form a special category of constants used to handle fixed

sequences of characters and are written as a sequence of any number of characters surrounded by double quotes:

```
"This is literally a string"
```

The null (empty) string is written "". The characters inside the double quotes can include escape sequences ("\n" - a new line escape sequence).

A Constant expression is an expression that always evaluates to a constant. They are evaluated just as regular expressions are.

Punctuator (also known as separator) in AFL is one of the following characters:
( ) , ; = .

Parentheses (open (and close)) group expressions, isolate conditional expressions and indicate function calls and function parameters:
d = c * (a + b) /* override normal precedence */
a= (b AND c) OR (d AND e) /* conditional expression */
func() /* function call no arguments */

The *comma* (,) separates the elements of a function argument list

The *semicolon* (;) is a statement terminator. Any legal AFL expression followed by a semicolon is interpreted as a statement, known as expression statement. The expression is evaluated and its value
is discarded (except Guru Commentaries where string values are written to output window)

The *dot* (.) is a member access operator. It is used to call COM object methods. If myobj variable holds the object, using dot operator we can call the methods (functions) of myobj object:

myobj.Method();

The *equal sign* (=) separates variable declarations from initialization lists:
x = 5;
It also indicates the default value for a parameter (see built-in function description):
macd(fast = 12; slow = 26) /* default values for fast and slow arguments)

Language structure

Each formula in AFL contains of one or more expression statements. Each statement MUST be terminated by semicolon (;). In this way you are able to break long expressions into several physical lines (in order to gain clarity) and AmiBroker will still treat it like a single statement until terminating semicolon. Examples:

```
x = ( y + 3 );          /* x is assigned the value of y + 3  */

x = y = 0;              /* Both x and y are initialized to 0 */

proc( arg1, arg2 );     /* Function call, return value discarded */

y = z = ( f( x ) + 3 ); /* A function-call expression  */


my_indicator =    IIf( MACD() > 0,
               Close - MA(Close,9),
               MA( Close, 9 ) - Close );
              /* one statement in several lines */
```

Identifiers

Identifiers in AFL are used to identify variables and functions.
There are some predefined identifiers referencing built-in arrays and functions.
The most important are *price array identifiers*. They identify specific price fields that the formula should operate on. The valid price array identifiers are **open**, **high**, **low**, **close**, **volume**, **openint**, **average**. Price array identifiers can be abbreviated as shown in the following table. Note that these are not case-specific.

Long name	Abbreviation	Comment
Open	O	
High	H	
Low	L	
Close	C	
Volume	V	
OpenInt	OI	
Avg	<none available>	(High+Low+Close)/3 - so called "typical price"

Examples of the use of price array identifiers in formulas are shown below.

```
MA( Close, 10 ); IIf( H > Ref(H,-1), MA(H,20), MA(C,20) );
```

Operators

Comparision operators

Comparision operators are divided into two types:

- relational (<, >, <=, >=)
- equality (==, !=)

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

These operators give true (1) or false (0) value as a result of comparison.

Assignment operator

Symbol	Meaning
=	Store the value of the second operand in the object specified by the first operand (“simple assignment”).

The assignment operator assigns a value to a variable:

result = expression;

where *result* is variable identifier and *expression* is any numerical, array or text expression.

As the = operator behaves like other operators, expressions using it have a value in addition to assigning that value into variable. This means that you can chain assignment operators as follows:
j = k = l = 0;

j, k, and l equal zero after the example statement is executed.

Attention: please DO NOT confuse assignment operator (=) with equality check (==)

These are two different operators and you must not use assignment (=) to check for equality.

```
if( Name() = "MSFT" ) // WRONG !!! - variable assignment operator used instead of equality check
{

}

if( Name() == "MSFT" ) // CORRECT - equality operator used properly
{

}
```

This is one of common coding mistakes listed here.

Arithmetic operators

Formulas can contain the following mathematical operators:

Symbol	Meaning
+	Addition
-	Subtraction (or negative value)
*	Multiplication
/	Division
%	Modulus (or remainder) (AFL 1.7+)
^	Exponentiation (raising to a power)
|	Bit-wise "Or" (AFL 2.1+)
&	Bit-wise "And" (AFL 2.1+)

The following formulas illustrate the use of operators in a formula:

```
var1 = ( H + L ) / 2;

var2 = MA(C,10)-MA(C,20) / (H + L + C);

var3 = Close + ((1.02 * High)-High);
```

Logical operators

Symbol	Meaning
NOT	Logical "Not" - gives "True" when operand is equal to false
AND	Logical "And" - gives "True" result if BOTH operands are true at the same time
OR	Logical "Or" - gives "True" result if ANY of operands is true

If a formula requires multiple conditions, you can combine the conditions with AND and OR operators. For example, maybe you'd like to plot a +1 when the MACD is greater than zero and the RSI is greater than 70:

```
Condition = MACD() > 0 AND RSI(14) > 70;
```

You can add as many conditions within a formula as you like.

Compound assignment operators

Introduced in version 5.00, the compound operatos are specifeid in the form of:

destinvar op= expr;

where *destinvar* is the variable, *expr* is the expression, and *op* is one of the following artithmetic operators: +, -, *, /, %, &, |

The *destinvar op= expr* form behaves as:

destinvar = destinvar op expr;

This is shortcut form for common assignment statements like k = k + 2; so you can write it shorter as:

k += 2;

and it will work the same but little faster.

Full list of available assignment operators is here:

No	Symbol	Meaning
1	=	Store the value of the second operand in the object specified by the first operand (“simple assignment”).
2	*=	Multiply the value of the first operand by the value of the second operand; store the result in the object specified by the first operand.
3	/=	Divide the value of the first operand by the value of the second operand; store the result in the object specified by the first operand.

4	%=	Take modulus of the first operand specified by the value of the second operand; store the result in the object specified by the first operand.
5	+=	Add the value of the second operand to the value of the first operand; store the result in the object specified by the first operand.
6	–=	Subtract the value of the second operand from the value of the first operand; store the result in the object specified by the first operand.
7	&=	Obtain the bitwise AND of the first and second operands; store the result in the object specified by the first operand.
8	|=	Obtain the bitwise inclusive OR of the first and second operands; store the result in the object specified by the first operand

typeof() operator

The typeof operator is used in the following way:
typeof (operand)
The typeof operator returns a string indicating the type of the *unevaluated* operand. operand is the string, variable, function identifier, or object for which the type is to be returned.
When supplying identifier, it should be provided alone, without arithmetic operators, without extra arguments and without braces.
If you want to check the type of value returned by the function, you must first assign the return value to a variable and then use
typeof(variable).
Possible return values are:

- "undefined" - identifier is not defined
- "number" - operand represents a number (scalar)
- "array" - operand represents an array
- "string" - operand represents a string
- "function" - operand is a built-in function identifier
- "user function" - operand is a user-defined function
- "object" - operand represents COM object
- "member" - operand represents member function or property of COM object
- "handle" - operand represents Windows handle
- "unknown" - type of operand is unknown (should not happen)typeof operator allows among other things to detect undefined variables in the following way

```
if( typeof( somevar ) == "undefined" )
{
/// when somevar is undefined the code here will execute
}
```

The following sample COMMENTARY code shows the output of typeof() in some common situations:

```
x = MACD();
y = LastValue( x );
function testfun() { return 1; };
```

```
printf( typeof( test ) + "\n" ); // the undefined variable
printf( typeof( 1 ) + "\n"); // literal number
printf( typeof( "checking" ) + "\n"); // literal string
printf( typeof( x ) + "\n"); // array variable
printf( typeof( y ) + "\n"); // scalar variable
printf( typeof( MACD ) + "\n"); // function identifier
printf( typeof( testfun ) + "\n" ); // user function identifier
```

Operator precedence and the parentheses

AFL supports parentheses in formulas.

Parentheses can be used to control the operation precedence (the order in which the operators are calculated). AmiBroker always does operations within the innermost parentheses first. When parentheses are not used, the precedence is as follows (higher precedence listed first):

No	Symbol	Meaning
1	++	Post-increment/pre-increment (i++ works like i = i + 1)
2	--	Post-decrement/pre-decrement (i-- works like i = i - 1)
3	[ ]	Array element (subscript) operator
4	^	Exponentiation
5	**-**	Negation - Unary minus
6	*	Multiplication
7	/	Division
8	%	Reminder (Modulo operator)
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" (AFL 2.1+)
18	|	Bit-wise "Or" (AFL 2.1+)
19	**NOT**	Logical "Not"
20	**AND**	Logical "And"
21	**OR**	Logical "Or"

22	**=**	Variable assignment operator
23	***=** **/=** **%=** **+=** **-=** **& =** **|=**	Compound assignment

The expression

```
H + L / 2;
```

(without parenthesis) would be calculated by AmiBroker as "L / 2" plus "H", since division has a higher precedence. This would result in a much different value than

```
(H + L)/2;
```

A few words about increment/decrement operators. There are two kinds of them: postfix and prefix.

The unary operators (++ and --) are called “prefix” increment or decrement operators when the increment or decrement operators appear before the operand. Postfix increment and decrement has higher precedence than prefix increment and decrement operators. When the operator appears before its operand, the operand is incremented or decremented and its new value is the result of the expression.

```
i = 5;

j = ++i; // i will be incremented first and result (number 6) will be assigned to
j.
```

The result of the postfix increment or decrement operation is the value of the postfix-expression before the increment or decrement operator is applied. The type of the result is the same as that of the postfix-expression but is no longer an l-value. After the result is obtained, the value of the operand is incremented (or decremented).

```
i = 5;

j = i++; // j will be assigned the value of 5 (before incrementation) and then i
will be incremented to 6.
```

Accessing array elements: [ ] - subscript operator

An array identifier followed by an expression in square brackets ([ ]) is a subscripted representation of an element of an array object.

```
arrayidentifier [ expression ]
```

It represents the value of expression-th element of array.

BarCount constant gives the number of bars in array (such as Close, High, Low, Open, Volume, etc). Array elements are numbered from 0 (zero) to BarCount-1. BarCount does NOT change as long as your formula continues execution, but it may change between executions when new bars are added, zoom factor is changed or symbol is changed.

To get the first bar you can use array[0], to get the last bar of array you can use array[BarCount - 1];

For example:

```
Close[ 5 ];
```
Represents the sixth element (bar) of the close array.

```
Close[ 0 ];
```
Represents the very first available bar of the close array.

```
High[ BarCount - 1 ];
```
Represents the last bar of High array.

Matrices and Matrix operators

Matrices are two-dimensional arrays of numbers.

To create a matrix use:

```
my_var_name = Matrix( rows, cols, initvalue);
```

To access matrix elements, use:

```
my_var_name[ row ][ col ];
```

where
row is a row index (0... number of rows-1)
and
col is a column index (0... number of columns-1)

Matrices and their elements support all scalar (element-wise) arithmetic and logical operations.

All these standard operators are performed on matrices element-wise. For that reason for example to add two matrices they must be the same size (the number of rows and columns must be the same). If they are not the same it is up to you how to perform calculation on each element via loop.

So you can for example add, subtract, multiply, divide two matrices if they have same dimensions with one call.

```
x = Matrix( 5, 6, 9 ); // matrix 5 rows 6 columns, initial value 9
y = Matrix( 5, 6, 10 ); // matrix 5 rows 6 columns, initial value 10

z = y - x; // will give you matrix 5 rows and 6 columns filled with elements
holding value 1 (difference between 10 and 9).
```

All those operations are performed ELEMENT-WISE.

You can also apply any arithmetic and logical operation on matrix AND scalar value. This would perform element-wise operation on each element of source matrix and given scalar value.

```
m = Matrix( 10, 10, 0 ); // m will be 10x10 matrix filled with zeros
z = m; // z is now also a matrix

for( i = 0; i < 10; i++ )
{
  z[ i ][ 4 ] = i; // fill z with some other values, note that m will remain
unaffected.
}

for( i = 0; i < 10; i++ )
 _TRACEF( "%g = %g, %g, %g\n", i, m[i][1], m[ i][4], z[ i][4]);

// scalar addition (element wise)
z += 3;
m += 5;

for( i = 0; i < 10; i++ )
 _TRACEF( "%g = %g, %g, %g\n", i, m[i][1], m[ i][4], z[ i][4]);
```

There is one special operator that works only on matrices - it is **matrix product.** The operator for matrix product is @ (the 'at' sign). Matrix product is the linear algebra way to multiply matrices. If you write C = A @ B, it multiplies matrix A(n,k) by matrix B(k,m) to produce matrix C(n,m) so the number of columns in matrix A must be equal to number of rows in matrix B. For more info see: https://en.wikipedia.org/wiki/Matrix_multiplication The precedence of matrix product @ operator is the same as * (so it has higher precedence than addition and subtraction).

```
A = Matrix( 1, 3 );
B = Matrix( 3, 2 );

// matrix A = [ 1, 4, 6 ]
// matrix B =
// [ 2, 3 ]
// [ 5, 8 ]
// [ 7, 9 ]

A[ 0 ][ 0 ] = 1; A[ 0 ][ 1 ] = 4; A[ 0 ][ 2 ] = 6;

B[ 0 ][ 0 ] = 2; B[ 0 ][ 1 ] = 3;
B[ 1 ][ 0 ] = 5; B[ 1 ][ 1 ] = 8;
B[ 2 ][ 0 ] = 7; B[ 2 ][ 1 ] = 9;

X = A @ B;

_TRACEF("%g %g", X[ 0 ][ 0 ], X[ 0 ][ 1 ] );
```

Compound statements (Blocks)

A compound statement consists of zero or more statements enclosed in curly braces ({ }). A compound statement can be used anywhere a statement is expected. Compound statements are commonly called "blocks."

```
{
        statement1;
        ....
        statementN;
}
```

(this is 'borrowed' from C language, users of other programming languages are used to use BEGIN for { and END for })

```
if( Amount > 100 )
{
   _TRACE("Amount above 100");
    Balance = Balance + Amount;
}
else
    Balance = Balance - Amount;
```

Built-in Functions

In addition to mathematical operators, AmiBroker contains over 70 built-in functions that perform mathematical operations.

The following formula consists of a single function that gives the square roots of the closing prices:

```
sqrt( Close );
```

The following formula consists of a single function that gives a 14-period RSI indicator:

```
Graph0 = RSI(14);
```

The following formula consists of two functions. The result is the difference between the MACD indicator and a 9-period exponential moving average of the MACD:

```
Graph0 = MACD() - EMA(MACD(),9);
```

All function calls must consist of function identifier (name) followed by a pair of parentheses.

As has been eluded to in earlier examples, a function can be "nested" within a function. The nested function can serve as the main function's data array parameter. The following examples show functions nested within functions:

```
MA( RSI(15), 10 );

MA( EMA( RSI(15), 20), 10 );
```

The first example calculates a 10-period simple moving average of a 15-period Relative Strength Index (RSI).

The second example calculates a 20-period exponential moving average of a 15-period RSI, and then calculates a 10-period simple moving average of this moving average.

Conditional function IIF()

The iif() function is used to create **conditional assignments**. It contains three parameters as shown in the following example.

```
dynamicrsi = IIf( Close > MA(C,10), RSI(9), RSI(14) );
```

The above "iif" statement reads (in English) as follows: If today's close is greater than today's 10-day simple moving average of the close, then assign a 9-day RSI to the *dynamicrsi* variable, otherwise, assign a 14-day RSI. The next formula assigns positive volume to *volresult* variable if the close is greater than the median price. Otherwise, "negative volume" is assigned.

```
volresult = IIf( Close > (High+Low)/2, Volume, -Volume );
```

If you simply want an expression to be evaluated as either true or false, it can be done without the use of the iif() function. The following formula will result in either a 1 (true) or a 0 (false):

```
result = RSI(14) > 70;
```

The same done with iif() gives the same results, but the formula is longer.

```
result = IIf(RSI(14) > 70, 1, 0 );
```

Please note that IIF is a function - so the result of evaluation is returned by that function and should be assigned to some variable.

IIf always evaluates both TRUE_PART and FALSE_PART, even though it returns only one of them. Because of this, you should watch for undesirable side effects. **IIF function is NOT a flow-control statement**. If you need flow control (conditional execution of some code parts) you should look for **if-else** conditional statement described later in this document.

The following example shows one **common error** made with IIF function:

```
      IIf( condition, result = 7, result = 9 ); // THIS IS WRONG
```

Correct usage is:

```
      result = IIf( condition, 7, 9 );
```

/* 7 or 9 is *returned* and assigned to *result* variable depending on *condition* */

Variables

In order to shorten, simplify, enhance, and make the maintenance of complex formulas easier, you may want to use variables. In fact using variables you can significantly improve formula calculation speed. So it is strongly recommended to use variables and there is **no limit** on number of variables you can define.

A variable is an identifier that is assigned to an expression or a constant. The number of variables used in a formula is not limited. Variables must be assigned before the variable is used in the formula. Variables cannot be assigned within a function call.

User-defined variable names (identifiers) cannot duplicate names already used by functions (e.g., ma, rsi, cci, iif, etc.) or predefined array identifiers (e.g., open, high, low, close, simple, o, c, l, h, s, a).

Reserved variables

AmiBroker uses some reserved variable names in its formulas, for example in Auto-Analysis window you have to assign values to 2 variables named 'buy' or 'sell' to specify the conditions where "buy" and "sell" conditions occur. For example (system that buys when MACD rises above 0 line, and sells when MACD falls below 0 line)

```
Buy  = Cross( MACD(), 0 );
Sell = Cross( 0, MACD() );
```

AmiBroker uses the following reserved variable names. Please note that variables marked as obsolete should NOT be used in new coding. They are left for backward compatibility only and new formulas should use modern functions like Plot() to plot indicators and AddColumn() to define exploration columns.

Variable	Usage	Applies to
buy	defines "buy" (enter long position) trading rule	Automatic Analysis, Commentary
sell	defines "sell" (close long position) trading rule	Automatic Analysis, Commentary
short	defines "short" (enter short position - short sell) trading rule	Automatic Analysis
cover	defines "cover" (close short position - buy to cover) trading rule	Automatic Analysis
buyprice	defines buying price array (this array is filled in with the default values according to the Automatic Analyser settings)	Automatic Analysis
sellprice	defines selling price array (this array is filled in with the default values according to the Automatic Analyser settings)	Automatic Analysis
shortprice	defines short selling price array (this array is filled in with the default values according to the Automatic Analyser settings)	Automatic Analysis
coverprice	defines buy to cover price array (this array is filled in with the default values according to the Automatic Analyser settings)	Automatic Analysis
title	defines title text (overrides any graph*N*name)	Indicators
tooltip	**Obsolete in 5.40**. Use Data window instead or use Plot() with styleHidden if you want to add your custom values to data tooltip.	Indicators
graphxspace	defines percentage extra space added at the top and the bottom of the chart	Indicators
graphzorder	GraphZOrder variable allows to change the order of plotting indicator lines. When GraphZOrder is not defined or is zero (false) - old ordering	Indicators

	(last to first) is used, when GraphZOrder is 1 (true) - reverse ordering is applied.	
exclude	If defined, a true (or 1) value of this variable excludes current symbol from scan/exploration/back test. They are also not considered in buy and hold calculations. Useful when you want to narrow your analysis to certain set of symbols.	Automatic Analysis
roundlotsize	defines round lot sizes used by backtester (see explanations below)	Automatic Analysis (new in 4.10)
ticksize	defines tick size used to align prices generated by **built-in stops** (see explanations below) (note: it does not affect entry/exit prices specified by buyprice/sellprice/shortprice/coverprice)	Automatic Analysis (new in 4.10)
pointvalue	allows to read and modify future contract point value (see backtesting futures) CAVEAT: this AFL variable is by default set to 1 (one) regardless of contents of Information window UNLESS you turn ON futures mode (SetOption("FuturesMode", True))	Automatic Analysis (new in 4.10)
margindeposit	allows to read and modify future contract margin (see backtesting futures)	Automatic Analysis (new in 4.10)
positionsize	Allows control dollar amount or percentage of portfolio that is invested into the trade (more information available in the "Tutorial: Backtesting your trading ideas")	Automatic Analysis (new in 3.9)
positionscore	Defines the score of the position. More details: "Tutorial: Portfolio Backtesting")	Automatic analysis
numcolumns	Exploration only: defines the number of your own columns (excluding predefined ticker and date columns) and assign the column value to the variable	Automatic Analysis
filter	Exploration only: controls which symbols/quotes are accepted. If "true" (or 1) is assigned to that variable for given symbol/quote it will be displayed in the report. So, for example, the following formula will accept all symbols with closing prices greater than 50 : `filter = close > 50;`	Automatic Analysis
column*N* (obsolete)	Exploration only: defines *N*th column value. Example: `column0 = Close;`	Automatic Analysis
column*N*format (obsolete)	Exploration only: allows you to define the formatting applied to numbers. By default all variables are displayed with 2 decimal digits, but you can change this by assigning a different value to this variable: 1.5 gives 5 decimal digits, 1.0 gives no decimal digits. So, in our example, typing: `column0format = 1.4;`	Automatic Analysis

	will give closing prices displayed with 4 decimal digits. (Note for advanced users: the integer part of this number can be used to pad formatted number with spaces - 6.0 will give no decimal digits but a number space-padded upto 6 characters.)	
column*N*name (obsolete)	Exploration only: allows you to define the header name. Assigning `column0name = "Close";` will change the name of the first custom column from the default "Column 0" to more appropriate "Close".	Automatic Analysis
maxgraph (obsolete)	specifies maximum number of graphs to be drawn in custom indicator window (default=3)	Indicators
graph*N* *(obsolete)*	defines the formula for the graph number *N* (where *N* is a number 0,1,2,..., maxgraph-1)	Indicators
graph*N*name (obsolete)	defines the name of *N*th graph line. This will appear in the title of the chart pane	Indicators

graph*N*color
(obsolete)
	defines the color index of *N*th graph line (color indexes are related to the current palette - see Preferences/Color)
	
	colorCustom1 = 0
	colorCustom2 = 1
	colorCustom3 = 2
	colorCustom4 = 3
	colorCustom5 = 4
	colorCustom6 = 5
	colorCustom7 = 6
	colorCustom8 = 7
	colorCustom9 = 8
	colorCustom10 = 9
	colorCustom11 = 10
	colorCustom12 = 11
	colorCustom13 = 12
	colorCustom14 = 13
	colorCustom15 = 14
	colorCustom16 = 15
	
	colorBlack = 16
	colorBrown = 17
	colorDarkOliveGreen = 18
	colorDarkGreen = 19
	colorDarkTeal = 20
	colorDarkBlue = 21
	colorIndigo = 22
	colorDarkGrey = 23
	
	colorDarkRed = 24
	colorOrange = 25
	colorDarkYellow = 26
		Indicators

	colorGreen = 27
	colorTeal = 28
	colorBlue = 29
	colorBlueGrey = 30
	colorGrey40 = 31
	
	colorRed = 32
	colorLightOrange = 33
	colorLime = 34
	colorSeaGreen = 35
	colorAqua = 35
	colorLightBlue = 37
	colorViolet = 38
	colorGrey50 = 39
	
	colorPink = 40
	colorGold = 41
	colorYellow = 42
	colorBrightGreen = 43
	colorTurquoise = 44
	colorSkyblue = 45
	colorPlum = 46
	colorLightGrey = 47
	
	colorRose = 48
	colorTan = 49
	colorLightYellow = 50
	colorPaleGreen = 51
	colorPaleTurquoise = 52
	colorPaleBlue = 53
	colorLavender = 54
	colorWhite = 55	

graph*N*barcolor (obsolete)	defines the array that holds palette indexes for each bar drawn	Indicators

graph*N*style (obsolete)	defines the style of *N*th graph. Style is defined as a combination (sum) of one or more following flags:
	
	styleLine = 1 - normal (line) chart (default)
	styleHistogram = 2 - histogram chart
	styleThick =4 - fat (thick)
	styleDots = 8 - include dots
	styleNoLine = 16 - no line
	styleLog = 32 - semi-logarithmic scale
	styleCandle = 64 - candlestick chart
	styleBar = 128 - traditional bar chart
	styleNoDraw = 256 - no draw (perform axis scaling only)
	styleStaircase = 512 - staircase (square) chart
	styleSwingDots = 1024 - middle dots for staircase chart
	styleNoRescale = 2048 - no rescale
	styleNoLabel = 4096 - no value label
		Indicators

	stylePointAndFigure = 8192 - point and figure (new in 4.20): styleArea = 16384 - area chart (extra wide histogram) styleOwnScale = 32768 - plot is using independent scaling styleLeftAxisScale = 65536 - plot is using left axis scale (independent from right axis) Not all flag combinations make sense, for example (64+1) (candlestick + line) will result in candlestick chart (style=64) Note on candlestick/bar charts: these styles use indirectly O, H, L arrays in addition to graph*N*. So ordinary candlestick price chart formula is graph0=close; graph0style=64;. But if you want to draw something else than close price you have to assign new values to predefined O,H,L arrays.	
graph*N*barcolor (obsolete)	defines the array of color indexes for the bars and candlesticks in *N*th graph ine (color indexes are related to the current palette - see Preferences/Color)	Indicators

SEE ALSO:

- **KEYWORDS**
- **USER-DEFINABLE PROCEDURES, LOCAL/GLOBAL SCOPE**

Keywords

The following are keywords in AmiBroker Formula Language:

Loops:

- do (part of do-while statement)
- while
- for

Conditional execution / Flow control:

- if (part of if-else statement)
- else (part of if-else statement)
- switch
- break (part of the switch statement or for/while statements)
- case (part of the switch statement)
- continue (part of for/while statements)
- default (part of switch statement)

Functions:

- function
- procedure
- return
- local (variable scope)
- global (variable scope)

Variable type:

- static
- typeof

break Keyword

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.

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 );
}
```

For an example of using the break statement within the body of a switch statement, see The switch Statement.

case Keyword

The case keyword is an integral part of switch-case statement.

continue Keyword

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

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

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 );
}
```

default Keyword

The default keyword is an integral part of switch-case statement.

do Keyword

The **do** keyword is a part of **do-while** statement.

do-while Statement

The **do-while** statement lets you repeat a statement or compound statement until a specified expression becomes false.

Syntax

do *statement* **while (** *expression* **) ;**

The *expression* in a **do-while** statement is evaluated after the body of the loop is executed. Therefore, the body of the loop is always executed at least once.

The *expression* must have numeric or boolean type. Execution proceeds as follows:

1. The statement body is executed.
2. Next, *expression* is evaluated. If *expression* is false, the **do-while** statement terminates and control passes to the next statement in the program. If *expression* is true (nonzero), the process is repeated, beginning with step 1.

This is an example of the **do-while** statement:

```
x=100;
do
{
    y = sin( x );
    x--;
} while ( x > 0 );
```

In this **do-while** statement, the two statements `y = sin( x );` and `x--;` are executed, regardless of the initial value of `x`. Then `x > 0` is evaluated. If `x` is greater than 0, the statement body is executed again and `x > 0` is reevaluated. The statement body is executed repeatedly as long as `x` remains greater than 0. Execution of the **do-while** statement terminates when `x` becomes 0 or negative. The body of the loop is executed at least once.

else Keyword

The **else** keyword is an optional part of if-else statement.

if, else Statement

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

The **if** keyword executes *statement1* if *expression* is true (nonzero); if **else** is present and *expression* is false (zero), it executes *statement2*. After executing *statement1* or *statement2*, control passes to the next statement.

Example 1

```
if ( i > 0 )
    y = x / i;
else
{
    x = i;
    y = abs( x );
}
```

In this example, the statement `y = x/i;` is executed if `i` is greater than 0. If `i` is less than or equal to 0, `i` is assigned to `x` and `abs( x )` is assigned to `y`. Note that the statement forming the **if** clause ends with a semicolon.

When nesting **if** statements and **else** clauses, use braces to group the statements and clauses into compound statements that clarify your intent. If no braces are present, the compiler resolves ambiguities by associating each **else** with the closest **if** that lacks an **else**.

Example 2

```
if ( i > 0 )           /* Without braces */
    if ( j > i )
        x = j;
    else
        x = i;
```

The **else** clause is associated with the inner **if** statement in this example. If `i` is less than or equal to 0, no value is assigned to `x`.

Example 3

```
if ( i > 0 )
{             /* With braces */
    if ( j > i )
        x = j;
}
else
    x = i;
```

The braces surrounding the inner **if** statement in this example make the **else** clause part of the outer **if** statement. If `i` is less than or equal to 0, `i` is assigned to `x`.

Common misunderstandings

"New if-else problem"

Question:

Why I get the syntax error when I write: if(H > Ref(H,-1))

Answer:

if-else statement changes flow of execution (opposite to IIF function that evaluates all arguments and works on arrays) and you can not really write

```
if ( H >Ref(H,-1) )
```

because it has no meaning. It would translate to " If high array is higher than high array shifted one bar" (see tutorial below). Flow control statement (such as if-else) has to get SINGLE boolean value to make decision which execution path should be taken. If you write H (or High) it means ARRAY (entire array).
if you write H[i] - it means i-th element of the array. The subscript operator [ ] allows you to access individual array elements.

Instead you should write:

```
for( i = 1; i < BarCount; i++ )
{
   if ( High[ i ] > High[ i - 1 ] )
   {
       x[ i ] = High[ i ];
   }
   else
   {
       x[ i ] = Low[ i ];
   }
}
```

this will translate to correct one "for EVERY BAR 'i' assign i-th element of high array to the i-th element of x array if i-th element of high array is higher than the previous element, otherwise assign i-th of low array to the i-th element of x array". The rule is: new *if-else* and *while* statements need single boolean value (not array) to decide which execution path should be taken. If you want to use them with arrays you have to iterate through bars using *for* loop (as shown above).

On the other hand this can be implemented in single line using old-style array operations and IIF function:

```
x = IIf( High > Ref( High, -1 ), High, Low );
```

This works because IIF operates on ARRAYS as described in the tutorial.

As you can see in many cases old-style AFL provides much more compact form. I always tried to explain this advantage of AFL but only a few realised that. New control statements should be used where it is better to use them. As I tried to explain during last years in 80% of cases 'old-style' AFL provides the shortest formula. Only remaining 20% of cases needed script. Those 'script-only' cases now can be coded in native AFL thanks to new for/while/if-else statements. And this is correct usage of them - to replace script parts.

for Statement

The **for** statement lets you repeat a statement or compound statement a specified number of times. The body of a **for** statement is executed zero or more times until an optional condition becomes false.

Syntax

for (*init-expression* **;** *cond-expression* **;** *loop-expression* **)** *statement*

Execution of a **for** statement proceeds as follows:

1. The *init-expression*, is evaluated. This specifies the initialization for the loop. There is no restriction on the type of *init-expression*.
2. The *cond-expression*, is evaluated. This expression must have arithmetic type. It is evaluated before each iteration. Three results are possible:
    - If *cond-expression* is true (nonzero), *statement* is executed; then *loop-expression*, if any, is evaluated. The *loop-expression* is evaluated after each iteration. There is no restriction on its type. Side effects will execute in order. The process then begins again with the evaluation of *cond-expression*.
    - If *cond-expression* is false (0), execution of the **for** statement terminates and control passes to the next statement in the program.

This example illustrates the **for** statement:

```
myema[ 0 ] = Close[ 0 ];
for( i = 1; i < BarCount; i++ )
{
    myema[ i ] = 0.1 * Close[ i ] + 0.9 * myema[ i - 1 ];
}
```

This example iterates all bars of close array to calculate exponential moving average.

For loop is extremely flexible.

loop-expression can be ANY kind of expression you wish. You can produce not only regular series like this:

```
for( i = 0; i < BarCount; i = i + 3 ) // increment by 3 every iteration
```

but you can produce exponential series too:

```
for( i = 1; i < BarCount; i = i * 2 ) // produces series of 1, 2, 4, 8, 16, 32,
...
```

function Keyword

The **function** keyword begins definition of the user-function.

User-definable functions allow to encapsulate user code into easy-to-use modules that can be user in many places without 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.

A function definition establishes the name, and attributes (or parameters) of a function. A function definition must precede the call to the function. The definition starts with **function** keyword then follows function name, opening parenthesis then optional list of arguments and closing parenthesis. Later comes function body enclosed in curly braces.

A function call passes execution control from the calling function to the called function. The arguments, if any, are passed by value to the called function. Execution of a return statement in the called function returns control and possibly a value to the calling function.

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

Following is an example of function definition:

```
// the following function is 2nd order smoother

function IIR2( input, f0, f1, f2 )
{
    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 ];
    }

   return result;
}

Plot( Close, "Price", colorBlack, styleCandle );
Plot( IIR2( Close, 0.2, 1.4, -0.6 ), "function example", colorRed );
```

In this code **IIR2** is a user-defined function. **input, f0, f1, f2** are formal parameters of the functions.
At the time of function call the values of arguments are passed in these variables. Formal parameters behave like local variables.
Later we have **result** and **i** which are local variables. Local variables are visible inside function only. If any other function uses the same variable name they won't interfere between each other.

global Keyword

The **global** keyword declares global variable inside user-defined function. Global variable is the variable that is visible/accessible inside the function AND outside the function (at global formula level).

Due to the fact that AFL by default does not require to declare variables the decision whenever given variable is treated as local or global is taken depends on where it is FIRST USED.

You can however force AFL engine to require all variables to be declared using local or global keywords on formula-by-formula basis by placing SetOption("RequireDeclarations", True); at the top of the formula.

If given identifier appears first INSIDE function definition - then it is treated as LOCAL variable.
If given identifier appears first OUTSIDE function definition - then it is treated as GLOBAL variable.

This default behaviour can be however overriden using **global** and **local** keywords (introduced in 4.36) - see example 2.

Example (commentary):

```
k = 4; // this is GLOBAL variable

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 local variable in
function f

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

// Now even if we call function
// the value of our global variable z
// is not affected by function call because
// 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 )"
```

if Keyword

The **if** keyword is an required part of if-else statement.

if, else Statement

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

The **if** keyword executes *statement1* if *expression* is true (nonzero); if **else** is present and *expression* is false (zero), it executes *statement2*. After executing *statement1* or *statement2*, control passes to the next statement.

Example 1

```
if ( i > 0 )
    y = x / i;
else
{
    x = i;
    y = abs( x );
}
```

In this example, the statement `y = x/i;` is executed if `i` is greater than 0. If `i` is less than or equal to 0, `i` is assigned to `x` and `abs( x )` is assigned to `y`. Note that the statement forming the **if** clause ends with a semicolon.

When nesting **if** statements and **else** clauses, use braces to group the statements and clauses into compound statements that clarify your intent. If no braces are present, the compiler resolves ambiguities by associating each **else** with the closest **if** that lacks an **else**.

Example 2

```
if ( i > 0 )           /* Without braces */
    if ( j > i )
        x = j;
    else
        x = i;
```

The **else** clause is associated with the inner **if** statement in this example. If `i` is less than or equal to 0, no value is assigned to `x`.

Example 3

```
if ( i > 0 )
{             /* With braces */
    if ( j > i )
        x = j;
}
else
    x = i;
```

The braces surrounding the inner **if** statement in this example make the **else** clause part of the outer **if** statement. If `i` is less than or equal to 0, `i` is assigned to `x`.

Common misunderstandings

"New if-else problem"

Question:

Why I get the syntax error when I write: if(H > Ref(H,-1))

Answer:

if-else statement changes flow of execution (opposite to IIF function that evaluates all arguments and works on arrays) and you can not really write

```
if ( H >Ref(H,-1) )
```

because it has no meaning. It would translate to " If high array is higher than high array shifted one bar" (see tutorial below). Flow control statement (such as if-else) has to get SINGLE boolean value to make decision which execution path should be taken. If you write H (or High) it means ARRAY (entire array).
if you write H[i] - it means i-th element of the array. The subscript operator [ ] allows you to access individual array elements.

Instead you should write:

```
for( i = 1; i < BarCount; i++ )
{
   if ( High[ i ] > High[ i - 1 ] )
   {
       x[ i ] = High[ i ];
   }
   else
   {
       x[ i ] = Low[ i ];
   }
}
```

this will translate to correct one "for EVERY BAR 'i' assign i-th element of high array to the i-th element of x array if i-th element of high array is higher than the previous element, otherwise assign i-th of low array to the i-th element of x array". The rule is: new *if-else* and *while* statements need single boolean value (not array) to decide which execution path should be taken. If you want to use them with arrays you have to iterate through bars using *for* loop (as shown above).

On the other hand this can be implemented in single line using old-style array operations and IIF function:

```
x = IIf( High > Ref( High, -1 ), High, Low );
```

This works because IIF operates on ARRAYS as described in the tutorial.

As you can see in many cases old-style AFL provides much more compact form. I always tried to explain this advantage of AFL but only a few realised that. New control statements should be used where it is better to use them. As I tried to explain during last years in 80% of cases 'old-style' AFL provides the shortest formula. Only remaining 20% of cases needed script. Those 'script-only' cases now can be coded in native AFL thanks to new for/while/if-else statements. And this is correct usage of them - to replace script parts.

local Keyword

The **local** keyword declares local variable inside user-defined function. Local variable is the variable that is visible/accessible only inside the function.

Due to the fact that AFL by default does not require to declare variables the decision whenever given variable is treated as local or global is taken depends on where it is FIRST USED.

You can however force AFL engine to require all variables to be declared using local or global keywords on formula-by-formula basis by placing SetOption("RequireDeclarations", True); at the top of the formula.

If given identifier appears first INSIDE function definition - then it is treated as LOCAL variable.
If given identifier appears first OUTSIDE function definition - then it is treated as GLOBAL variable.

This default behaviour can be however overriden using **global** and **local** keywords (introduced in 4.36) - see example 2.

Example (commentary):

```
k = 4; // this is GLOBAL variable

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 local variable in
function f

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

// Now even if we call function
// the value of our global variable z
// is not affected by function call because
// 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 )"
```

procedure Keyword

The **procedure** keyword begins definition of the user-procedure.

Procedure is a function that does NOT return any value (does not have return statement).

Consult function keyword help for more details.

return Keyword

The **return** keyword allows to return the value from the function.

```
function RiseToAPowerOf2( x )
{
   return x ^ 2;
}
```

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

Consult function keyword help for more details.

static Keyword

The static keyword allows to declare

Syntax:

static *identifier;*

static *identifier1, identifier2, identifier3, ...;*

The **static** keyword declares identifier as static variable

Declared static variables can and should be used as 'regular' variables, no need to call functions (Static*).

IMPORTANT: The **static** keyword is only available when you use

```
#pragma enable_static_decl(prefix)
```

at the top of your formula. The *prefix* is added automatically to actual static variable name to avoid name clashes/conflicts. Prefix should be constant string literal (as in Example 1).

You can also have declared static variable with unique prefix that includes current chartID, interval or symbol. To do so use {chartid}, {interval} or {symbol} runtime tokens in the prefix string. They will get replaced in runtime by corresponding values (see Example 2).

Example 1.

```
// the pragma below is required to enable support for static declarations
// it also defines the prefix which is added to variables declared with static
keyword
// it is one prefix per formula.
#pragma enable_static_decl(prefix)

// it is really good idea to consequently use naming convention that
// distinguishes declared static variables from the others
// we highly recommend using UNDERSCORE '_' before all declared static variables
// so they can be easily spotted in the code
// The other choice could be s_ prefix

static _myvar; // this var is also accessible by "prefix_myvar"

printf( "This variable is really static %g", _myvar++ );

// keep in mind that for speed static variable is read only once at the
declaration
// and saved only once - when formula execution is completed
// This way declared static variables offer same speed as regular variables (no
speed penalty)
```

Example 2.

```
// Example 2 - if you want per-chartID declared static variables
#pragma enable_static_decl "myprefx{chartid}"
static x; // this static will really have name of myprefixIDx where ID is a
current chartID
```

More information on static keyword is included on the forum:
https://forum.amibroker.com/t/amibroker-6-27-1-beta-released/2925/

switch Statement

The switch and case statements help control complex conditional and branching operations. The switch statement transfers control to a statement within its body.

Syntax:

switch (*expression*)
{

case *constant-expression1* : *statement;*
case *constant-expression*2 : *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 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;
  }
```

More information can be found here: http://en.wikipedia.org/wiki/Switch_statement

typeof Keyword

The **typeof** keyword is a operator that returns the string describing the type of operand .

Syntax

The typeof operator is used in the following way:

typeof (*operand*)

The typeof operator returns a string indicating the type of the *unevaluated* *operand*. The *operand* is the string, variable, function identifier, or object for which the type is to be returned. When supplying identifier, it should be provided alone, without arithmetic operators, without extra arguments and without braces. If you want to check the type of value returned by the function, you must first assign the return value to a variable and then use:

typeof(variable)

Possible return values are:

- "undefined" - identifier is not defined
- "number" - operand represents a number (scalar)
- "array" - operand represents an array
- "string" - operand represents a string
- "matrix" - operand represents a matrix
- "function" - operand is a built-in function identifier
- "user function" - operand is a user-defined function
- "object" - operand represents COM object
- "member" - operand represents member function or property of COM object
- "handle" - operand represents Windows handle
- "unknown" - type of operand is unknown (should not happen)typeof operator allows among other things to detect undefined variables in the following way

```
if( typeof( somevar ) == "undefined" )
{
/// when somevar is undefined the code here will execute
}
```

The following sample COMMENTARY code shows the output of typeof() in some common situations:

```
x = MACD();
y = LastValue( x );
function testfun() { return 1; };
printf( typeof( test ) + "\n" ); // the undefined variable
printf( typeof( 1 ) + "\n"); // literal number
printf( typeof( "checking" ) + "\n"); // literal string
printf( typeof( x ) + "\n"); // array variable
printf( typeof( y ) + "\n"); // scalar variable
printf( typeof( MACD ) + "\n"); // function identifier
printf( typeof( testfun ) + "\n" ); // user function identifier
```

while Keyword

The **while** keyword is al part of while (described below) and do-while statements.

while Statement

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

Syntax

while (*expression* **)** *statement*

The *expression* must have arithmetic (numeric/boolean) type. Execution proceeds as follows:

1. The *expression* is evaluated.
2. If *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.

   If *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 - Alphabetical list of functions

1. **#include** (Miscellaneous functions) - preprocessor include command (AFL 2.2)
2. **#include_once** (Miscellaneous functions) - preprocessor include (once) command (AFL 2.70)
3. **#pragma** (Miscellaneous functions) - sets AFL pre-processor option (AFL 2.4)
4. **abs** (Math functions) - absolute value
5. **AccDist** (Indicators) - accumulation/distribution
6. **acos** (Math functions) - arccosine function
7. **AddColumn** (Exploration / Indicators) - add numeric exploration column (AFL 1.8)
8. **AddMultiTextColumn** (Exploration / Indicators) - adds exploration text column based on array (AFL 4.20)
9. **AddRankColumn** (Exploration / Indicators) - add ranking column(s) according to current sort set by SetSortColumns (AFL 5.70)
10. **AddRow** (Exploration / Indicators) - add raw text row to exploration (AFL 4.0)

11. **AddSummaryRows** (Exploration / Indicators) - add summary row(s) to the exploration output (AFL 3.2)
12. **AddTextColumn** (Exploration / Indicators) - add text exploration column (AFL 1.8)
13. **AddToComposite** (Composites) - add value to composite ticker (AFL 2.0)
14. **ADLine** (Composites) - advance/decline line (AFL 1.2)
15. **AdvIssues** (Composites) - advancing issues (AFL 1.2)
16. **AdvVolume** (Composites) - advancing issues volume (AFL 1.2)
17. **ADX** (Indicators) - average directional movement index (AFL 1.3)
18. **AlertIf** (Trading system toolbox) - trigger alerts (AFL 2.1)
19. **AlmostEqual** (Math functions) - rounding error insensitive comparison (AFL 2.80)
20. **AMA** (Moving averages, summation) - adaptive moving average (AFL 1.5)
21. **AMA2** (Moving averages, summation) - adaptive moving average (AFL 1.5)
22. **ApplyStop** (Trading system toolbox) - apply built-in stop (AFL 1.7)
23. **Asc** (String manipulation) - get ASCII code of character (AFL 2.80)
24. **asin** (Math functions) - arcsine function
25. **atan** (Math functions) - arc tan
26. **atan2** (Math functions) - calculates arctangent of y/x (AFL 2.90)
27. **ATR** (Indicators) - average true range (AFL 1.3)
28. **BarIndex** (Date/Time) - get zero-based bar number (AFL 2.3)
29. **BarsSince** (Trading system toolbox) - bars since
30. **BarsSinceCompare** (Trading system toolbox) - bars since comparision between past and present array values was true (AFL 4.40)
31. **BBandBot** (Indicators) - bottom bollinger band
32. **BBandTop** (Indicators) - top bollinger band
33. **BeginValue** (Date/Time) - Value of the array at the begin of the range (AFL 2.3)
34. **CategoryAddSymbol** (Information / Categories) - adds a symbol to a category (AFL 2.5)
35. **CategoryCreate** (Information / Categories) - add new category (such as watch list) (AFL 3.70)
36. **CategoryFind** (Information / Categories) - search for category by name (AFL 3.0)
37. **CategoryGetName** (Information / Categories) - get the name of a category (AFL 2.5)
38. **CategoryGetSymbols** (Information / Categories) - retrieves comma-separated list of symbols belonging to given category (AFL 2.5)
39. **CategoryRemoveSymbol** (Information / Categories) - remove a symbol from a category (AFL 2.5)
40. **CategorySetName** (Information / Categories) - set the name of category (group, market, watch list, industry) (AFL 3.20)
41. **CCI** (Indicators) - commodity channel index
42. **ceil** (Math functions) - ceil value
43. **Chaikin** (Indicators) - chaikin oscillator
44. **Chr** (String manipulation) - get string with given ASCII code (AFL 4.40)
45. **ClipboardGet** (Miscellaneous functions) - retrieves current contents of Windows clipboard (AFL 2.60)
46. **ClipboardSet** (Miscellaneous functions) - copies the text to the Windows clipboard (AFL 2.6)
47. **ColorBlend** (Indicators) - blends (mixes) two colors (AFL 3.30)
48. **ColorHSB** (Miscellaneous functions) - specify color using Hue-Saturation-Brightness (AFL 2.80)
49. **ColorRGB** (Miscellaneous functions) - specify color using Red-Green-Blue components (AFL 2.80)
50. **Correlation** (Statistical functions) - correlation (AFL 1.4)
51. **cos** (Math functions) - cosine
52. **cosh** (Math functions) - hyperbolic cosine function (AFL 2.80)
53. **CreateObject** (Miscellaneous functions) - create COM object (AFL 1.8)
54. **CreateStaticObject** (Miscellaneous functions) - create static COM object (AFL 1.8)
55. **Cross** (Trading system toolbox) - crossover check
56. **Cum** (Moving averages, summation) - cumulative sum
57. **CumProd** (Moving averages, summation) - cumulative product of all array elements (AFL 4.20)

58. **Date** (Date/Time) - date (AFL 1.1)
59. **DateNum** (Date/Time) - date number (AFL 1.4)
60. **DateTime** (Date/Time) - retrieves encoded date time (AFL 2.3)
61. **DateTimeAdd** (Date/Time) - adds specified number of seconds/minutes/hours/days to datetime (AFL 3.40)
62. **DateTimeConvert** (Date/Time) - date/time format conversion (AFL 2.90)
63. **DateTimeDiff** (Date/Time) - get difference in seconds between two datetime values (AFL 3.30)
64. **DateTimeFormat** (Date/Time) - converts datetime to string (AFL 4.20)
65. **DateTimeToStr** (Date/Time) - convert datetime to string (AFL 2.8)
66. **Day** (Date/Time) - day of month (AFL 1.4)
67. **DayOfWeek** (Date/Time) - day of week (AFL 1.4)
68. **DayOfYear** (Date/Time) - get ordinal number of day in a year (AFL 2.4)
69. **DaysSince1900** (Date/Time) - get number of days since January 1st, 1900 (AFL 3.20)
70. **DecIssues** (Composites) - declining issues (AFL 1.2)
71. **DecVolume** (Composites) - declining issues volume (AFL 1.2)
72. **DEMA** (Moving averages, summation) - double exponential moving average (AFL 2.0)
73. **EMA** (Moving averages, summation) - exponential moving average
74. **EnableRotationalTrading** (Trading system toolbox) - Turns on rotational-trading mode of the backtester (AFL 2.5)
75. **EnableScript** (Miscellaneous functions) - enable scripting engine
76. **EnableTextOutput** (Miscellaneous functions) - allows to enable or disable text output (AFL 2.20)
77. **EncodeColor** (Exploration / Indicators) - encodes color for indicator title (AFL 2.2)
78. **EndValue** (Date/Time) - value of the array at the end of the selected range (AFL 2.3)
79. **Equity** (Trading system toolbox) - calculate single-symbol equity line (AFL 2.0)
80. **erf** (Math functions) - calculates Gauss error function (erf) (AFL 4.40)
81. **Error** (Miscellaneous functions) - displays user-defined error message and stops the execution (AFL 3.7)
82. **EXP** (Math functions) - exponential function
83. **ExRem** (Trading system toolbox) - remove excessive signals (AFL 1.5)
84. **ExRemSpan** (Trading system toolbox) - remove excessive signals spanning given number of bars (AFL 2.0)
85. **fclose** (File Input/Output functions) - close a file (AFL 2.5)
86. **fdelete** (File Input/Output functions) - deletes a file (AFL 2.70)
87. **fdir** (File Input/Output functions) - list directory content (AFL 3.70)
88. **feof** (File Input/Output functions) - test for end-of-file (AFL 2.5)
89. **FFT** (Basic price pattern detection) - performs Fast Fourier Transform (AFL 2.90)
90. **fgetcwd** (File Input/Output functions) - get current working directory (AFL 4.10)
91. **fgets** (File Input/Output functions) - get a string from a file (AFL 2.5)
92. **fgetstatus** (File Input/Output functions) - retrieves file status/properties (AFL 2.90)
93. **FindIndex** (Miscellaneous functions) - find index of array item matching specified value (AFL 4.40)
94. **FIR** (Moving averages, summation) - Finite Impulse Response filter (AFL 3.40)
95. **FirstVisibleValue** (Indicators) - get first visible value of array (AFL 3.40)
96. **Flip** (Trading system toolbox) - (AFL 1.5)
97. **floor** (Math functions) - floor value
98. **fmkdir** (File Input/Output functions) - creates (makes) a directory (AFL 2.70)
99. **fopen** (File Input/Output functions) - open a file (AFL 2.5)
100. **Foreign** (Referencing other symbol data) - access foreign security data (AFL 1.5)
101. **fputs** (File Input/Output functions) - write a string to a file (AFL 2.5)
102. **frac** (Math functions) - fractional part
103. **frmdir** (File Input/Output functions) - removes a directory (AFL 2.70)
104. **FullName** (Information / Categories) - full name of the symbol (AFL 1.1)
105. **GapDown** (Basic price pattern detection) - gap down

106. **GapUp** (Basic price pattern detection) - gap up
107. **GetAsyncKeyState** (Miscellaneous functions) - query the current state of keyboard keys (AFL 3.60)
108. **GetBacktesterObject** (Trading system toolbox) - get the access to backtester object (AFL 2.60)
109. **GetBaseIndex** (Referencing other symbol data) - retrieves symbol of relative strength base index (AFL 2.1)
110. **GetCategorySymbols** (Information / Categories) - retrieves comma-separated list of symbols belonging to given category (AFL 2.4)
111. **GetChartBkColor** (Indicators) - get the RGB color value of chart background (AFL 3.20)
112. **GetChartID** (Exploration / Indicators) - get current chart ID (AFL 2.3)
113. **GetCursorMouseButtons** (Indicators) - get current state of mouse buttons (AFL 2.80)
114. **GetCursorXPosition** (Indicators) - get current X position of mouse pointer (AFL 2.80)
115. **GetCursorYPosition** (Indicators) - get current Y position of mouse pointer (AFL 2.80)
116. **GetDatabaseName** (Information / Categories) - retrieves folder name of current database (AFL 2.3)
117. **GetExtraData** (Miscellaneous functions) - get extra data from external data source (AFL 1.9)
118. **GetExtraDataForeign** (Miscellaneous functions) - get extra data from external data source for specified symbol (AFL 4.20)
119. **GetFnData** (Information / Categories) - get fundamental data (AFL 2.90)
120. **GetFnDataForeign** (Information / Categories) - get fundamental data for specified symbol (AFL 4.20)
121. **GetFormulaPath** (Miscellaneous functions) - get file path of current formula (AFL 3.90)
122. **GetLastOSError** (Miscellaneous functions) - get text of last operating system (Windows) error (AFL 4.30)
123. **GetObject** (Miscellaneous functions) - get handle to already running OLE object (AFL 4.40)
124. **GetOption** (Trading system toolbox) - gets the value of option in automatic analysis settings (AFL 2.60)
125. **GetPerformanceCounter** (Miscellaneous functions) - retrieves the current value of the high-resolution performance counter (AFL 2.90)
126. **GetPlaybackDateTime** (Date/Time) - get bar replay position date/time (AFL 3.0)
127. **GetPriceStyle** (Exploration / Indicators) - get current price chart style (AFL 2.70)
128. **GetRTData** (Miscellaneous functions) - retrieves the real-time data fields (AFL 2.60)
129. **GetRTDataForeign** (Miscellaneous functions) - retrieves the real-time data fields (for specified symbol) (AFL 2.80)
130. **GetScriptObject** (Miscellaneous functions) - get access to script COM object (AFL 1.8)
131. **GetTradingInterface** (Trading system toolbox) - retrieves OLE automation object to automatic trading interfac (AFL 2.70)
132. **GfxArc** (Low-level graphics) - draw an arc (AFL 3.0)
133. **GfxChord** (Low-level graphics) - draw a chord (AFL 3.0)
134. **GfxCircle** (Low-level graphics) - draw a circle (AFL 3.0)
135. **GfxDrawImage** (Low-level graphics) - draw bitmap image (AFL 4.20)
136. **GfxDrawText** (Low-level graphics) - draw a text (clipped to rectangle) (AFL 3.0)
137. **GfxEllipse** (Low-level graphics) - draw an ellipse (AFL 3.0)
138. **GfxFillSolidRect** (Low-level graphics) - fill rectangle with solid color (AFL 4.10)
139. **GfxGetTextWidth** (Low-level graphics) - get pixel width of text (AFL 2.80)
140. **GfxGradientRect** (Low-level graphics) - draw a rectangle with gradient fill (AFL 3.0)
141. **GfxLineTo** (Low-level graphics) - draw a line to specified point (AFL 3.0)
142. **GfxMoveTo** (Low-level graphics) - move graphic cursor to new position (AFL 3.0)
143. **GfxPie** (Low-level graphics) - draw a pie (AFL 3.0)
144. **GfxPolygon** (Low-level graphics) - draw a polygon (AFL 3.0)
145. **GfxPolyline** (Low-level graphics) - draw a polyline (AFL 3.0)
146. **GfxRectangle** (Low-level graphics) - draw a rectangle (AFL 3.0)
147. **GfxRoundRect** (Low-level graphics) - draw a rectangle with rounded corners (AFL 3.0)
148. **GfxSelectFont** (Low-level graphics) - create / select graphic font (AFL 3.0)
149. **GfxSelectHatchBrush** (Low-level graphics) - select hatch style brush (AFL 4.0)

150. **GfxSelectPen** (Low-level graphics) - create / select graphic pen (AFL 3.0)
151. **GfxSelectSolidBrush** (Low-level graphics) - create / select graphic brush (AFL 3.0)
152. **GfxSelectStockObject** (Low-level graphics) - select built-in graphic object (AFL 4.00)
153. **GfxSetBkColor** (Low-level graphics) - set graphic background color (AFL 3.0)
154. **GfxSetBkMode** (Low-level graphics) - set graphic background mode (AFL 3.0)
155. **GfxSetCoordsMode** (Low-level graphics) - set low-level graphics co-ordinate mode (AFL 2.80)
156. **GfxSetOverlayMode** (Low-level graphics) - set low-level graphic overlay mode (AFL 3.0)
157. **GfxSetPixel** (Low-level graphics) - set pixel at specified position to specified color (AFL 3.0)
158. **GfxSetTextAlign** (Low-level graphics) - set text alignment (AFL 3.0)
159. **GfxSetTextColor** (Low-level graphics) - set graphic text color (AFL 3.0)
160. **GfxSetZOrder** (Low-level graphics) - set current low-level graphic Z-order layer (AFL 2.80)
161. **GfxTextOut** (Low-level graphics) - writes text at the specified location (AFL 3.0)
162. **GicsID** (Information / Categories) - get GICS category information (AFL 3.40)
163. **GroupID** (Information / Categories) - get group ID/name (AFL 1.8)
164. **GuiButton** (GUI functions) - create on-chart button control (AFL 4.30)
165. **GuiCheckBox** (GUI functions) - creates on-chart checkbox control (AFL 4.30)
166. **GuiDateTime** (GUI functions) - creates on-chart date-time picker control (AFL 4.30)
167. **GuiEdit** (GUI functions) - create on-chart edit control (AFL 4.30)
168. **GuiEnable** (GUI functions) - enables or disables on-chart control (AFL 4.30)
169. **GuiGetCheck** (GUI functions) - get checked state of control (AFL 4.30)
170. **GuiGetEvent** (GUI functions) - get GUI event (AFL 4.30)
171. **GuiGetText** (GUI functions) - get text from on-chart control (AFL 4.30)
172. **GuiGetValue** (GUI functions) - get numeric value of on-chart control (AFL 4.30)
173. **GuiRadio** (GUI functions) - creates on-chart radio button control (AFL 4.30)
174. **GuiSendKeyEvents** (GUI functions) - request GUI notifications for specified keys (AFL 4.40)
175. **GuiSetCheck** (GUI functions) - set checked state of on-chart control (AFL 4.30)
176. **GuiSetColors** (GUI functions) - set colors for GUI controls (AFL 2.30)
177. **GuiSetFont** (GUI functions) - set the font for on-chart control (AFL 4.30)
178. **GuiSetRange** (GUI functions) - set slider control range (AFL 4.30)
179. **GuiSetText** (GUI functions) - set text value of on-chart control (AFL 4.30)
180. **GuiSetValue** (GUI functions) - set numeric value of on-chart control (AFL 4.30)
181. **GuiSetVisible** (GUI functions) - shows or hides on-chart control (AFL 4.30)
182. **GuiSlider** (GUI functions) - creates on-chart slider control (AFL 4.30)
183. **HHV** (Lowest/Highest) - highest high value
184. **HHVBars** (Lowest/Highest) - bars since highest high
185. **Highest** (Lowest/Highest) - highest value
186. **HighestBars** (Lowest/Highest) - bars since highest value
187. **HighestSince** (Lowest/Highest) - highest value since condition met (AFL 1.4)
188. **HighestSinceBars** (Lowest/Highest) - bars since highest value since condition met (AFL 1.4)
189. **HighestVisibleValue** (Indicators) - get the highest value within visible chart area (AFL 3.30)
190. **HMA** (Moving averages, summation) - Hull Moving Average (AFL 3.40)
191. **Hold** (Trading system toolbox) - hold the alert signal
192. **Hour** (Date/Time) - get current bar's hour (AFL 2.0)
193. **IcbID** (Information / Categories) - get ICB category information (AFL 3.60)
194. **IIf** (Trading system toolbox) - immediate IF function
195. **IIR** (Moving averages, summation) - infinite impulse response filter (AFL 4.0)
196. **IndustryID** (Information / Categories) - get industry ID / name (AFL 1.8)
197. **InGICS** (Information / Categories) - test GICS membership (AFL 3.40)
198. **InICB** (Information / Categories) - test ICB membership (AFL 3.60)
199. **Inside** (Basic price pattern detection) - inside day
200. **Int** (Math functions) - integer part
201. **InternetClose** (File Input/Output functions) - close Internet file handle (AFL 4.20)

202. **InternetGetStatusCode** (File Input/Output functions) - returns HTTP status code of last Internet call (AFL 4.40)
203. **InternetOpenURL** (File Input/Output functions) - opens Internet web resource (URL) (AFL 4.20)
204. **InternetPostRequest** (File Input/Output functions) - send HTTP Post request to Internet web resource (URL) (AFL 4.30)
205. **InternetReadString** (File Input/Output functions) - read a string from Internet resource (AFL 4.20)
206. **InternetSetAgent** (File Input/Output functions) - set agent string for Internet function (AFL 4.30)
207. **InternetSetHeaders** (File Input/Output functions) - set custom HTTP headers for subsequent web requests (AFL 4.40)
208. **InternetSetOption** (File Input/Output functions) - set HTTP option for internet session (AFL 4.40)
209. **Interval** (Date/Time) - get bar interval (in seconds) (AFL 2.1)
210. **inverf** (Math functions) - inverse Gauss erf function (AFL 4.40)
211. **InWatchList** (Information / Categories) - watch list membership test (by ordinal number)
212. **InWatchListName** (Information / Categories) - watch list membership test (by name) (AFL 3.0)
213. **IsContinuous** (Information / Categories) - checks 'continuous quotations' flag state (AFL 2.60)
214. **IsEmpty** (Miscellaneous functions) - empty value check (AFL 1.5)
215. **IsFavorite** (Information / Categories) - check if current symbol belongs to favorites (AFL 2.5)
216. **IsFinite** (Miscellaneous functions) - check if value is not infinite (AFL 2.3)
217. **IsIndex** (Information / Categories) - check if current symbol is an index (AFL 2.5)
218. **IsNan** (Miscellaneous functions) - checks for NaN (not a number) (AFL 2.3)
219. **IsNull** (Miscellaneous functions) - check for Null (empty) value (AFL 2.3)
220. **IsTrue** (Miscellaneous functions) - true value (non-empty and non-zero) check (AFL 1.5)
221. **Kurtosis** (Math functions) - calculates kurtosis (AFL 4.20)
222. **LastValue** (Trading system toolbox) - last value of the array
223. **LastVisibleValue** (Indicators) - get last visible value of array (AFL 3.40)
224. **LineArray** (Exploration / Indicators) - generate trend-line array (AFL 2.5)
225. **LinearReg** (Statistical functions) - linear regression end-point (AFL 2.2)
226. **LinRegIntercept** (Statistical functions) - (AFL 2.2)
227. **LinRegSlope** (Statistical functions) - linear regression slope (AFL 1.4)
228. **LLV** (Lowest/Highest) - lowest low value
229. **LLVBars** (Lowest/Highest) - bars since lowest low
230. **log** (Math functions) - natural logarithm
231. **log10** (Math functions) - decimal logarithm
232. **Lookup** (Date/Time) - search the array for bar with specified date/time (AFL 3.40)
233. **Lowest** (Lowest/Highest) - lowest value
234. **LowestBars** (Lowest/Highest) - bars since lowest
235. **LowestSince** (Lowest/Highest) - lowest value since condition met (AFL 1.4)
236. **LowestSinceBars** (Lowest/Highest) - barssince lowest value since condition met (AFL 1.4)
237. **LowestVisibleValue** (Indicators) - get the lowest value within visible chart area (AFL 3.30)
238. **MA** (Moving averages, summation) - simple moving average
239. **MACD** (Indicators) - moving average convergence/divergence
240. **MapCreate** (Map functions) - creates a map/dictionary object (holding key-value pairs) (AFL 4.90)
241. **MarketID** (Information / Categories) - market ID / name (AFL 1.8)
242. **Matrix** (Matrix functions) - create a new matrix (AFL 4.0)
243. **Max** (Math functions) - maximum value of two numbers / arrays
244. **MDI** (Indicators) - minus directional movement indicator (-DI) (AFL 1.3)
245. **Median** (Statistical functions) - calculate median (middle element) (AFL 2.5)
246. **MFI** (Indicators) - money flow index
247. **MicroSec** (Date/Time) - get bar's microsecond part of the timestamp
248. **MilliSec** (Date/Time) - get bar's millisecond part of the timestamp
249. **Min** (Math functions) - minimum value of two numbers / arrays
250. **Minute** (Date/Time) - get current bar's minute (AFL 2.0)

251. **Month** (Date/Time) - month (AFL 1.4)
252. **mtRandom** (Statistical functions) - Mersene Twister random number generator (AFL 3.0)
253. **mtRandomA** (Statistical functions) - Mersene Twister random number generator (array version) (AFL 3.0)
254. **MxCopy** (Matrix functions) - copy rectangular block from one matrix to another (AFL 4.40)
255. **MxDet** (Matrix functions) - calculate determinant of the matrix (AFL 4.10)
256. **MxFromString** (Matrix functions) - creates a new matrix out of string (AFL 4.10)
257. **MxGetBlock** (Matrix functions) - get rectangular block of items from matrix (AFL 4.10)
258. **MxGetSize** (Matrix functions) - get size of the matrix (AFL 4.0)
259. **MxIdentity** (Matrix functions) - create an identity matrix (AFL 4.0)
260. **MxInverse** (Matrix functions) - calculate inverse matrix (AFL 4.10)
261. **MxSetBlock** (Matrix functions) - sets values in the rectangular block of matrix cells (AFL 4.10)
262. **MxSolve** (Matrix functions) - solves linear equation system A @ X = B (AFL 4.10)
263. **MxSort** (Matrix functions) - sorts the matrix (AFL 4.10)
264. **MxSortRows** (Matrix functions) - sort the rows of the matrix (AFL 4.10)
265. **MxSum** (Matrix functions) - calculate grand sum of the matrix (AFL 4.20)
266. **MxToString** (Matrix functions) - convert matrix to string (AFL 4.10)
267. **MxTranspose** (Matrix functions) - creates transpose of an input matrix (AFL 4.0)
268. **Name** (Information / Categories) - ticker symbol (AFL 1.1)
269. **NormDist** (Statistical functions) - normal distribution function (AFL 4.20)
270. **NoteGet** (Miscellaneous functions) - retrieves the text of the note (AFL 2.6)
271. **NoteSet** (Miscellaneous functions) - sets text of the note (AFL 2.6)
272. **Now** (Date/Time) - gets current system date/time (AFL 2.3)
273. **NullCount** (Miscellaneous functions) - count consecutive Null values (AFL 3.90)
274. **NumToStr** (String manipulation) - convert number to string (AFL 2.5)
275. **NVI** (Indicators) - negative volume index
276. **Nz** (Miscellaneous functions) - Null (Null/Nan/Infinity) to zero (AFL 2.3)
277. **OBV** (Indicators) - on balance volume
278. **Optimize** (Trading system toolbox) - define optimization variable (AFL 1.7)
279. **OptimizerSetEngine** (Trading system toolbox) - select external optimization engine (AFL 3.20)
280. **OptimizerSetOption** (Trading system toolbox) - set the value of external optimizer engine parameter (AFL 3.20)
281. **OscP** (Indicators) - price oscillator
282. **OscV** (Indicators) - volume oscillator
283. **Outside** (Basic price pattern detection) - outside bar
284. **Param** (Exploration / Indicators) - add user user-definable numeric parameter (AFL 2.3)
285. **ParamColor** (Exploration / Indicators) - add user user-definable color parameter (AFL 2.3)
286. **ParamDate** (Exploration / Indicators) - add user user-definable date parameter (AFL 2.60)
287. **ParamField** (Exploration / Indicators) - creates price field parameter (AFL 2.70)
288. **ParamList** (Exploration / Indicators) - creates the parameter that consist of the list of choices (AFL 2.70)
289. **ParamStr** (Exploration / Indicators) - add user user-definable string parameter (AFL 2.3)
290. **ParamStyle** (Exploration / Indicators) - select styles applied to the plot (AFL 2.70)
291. **ParamTime** (Exploration / Indicators) - add user user-definable time parameter (AFL 2.60)
292. **ParamToggle** (Exploration / Indicators) - create Yes/No parameter (AFL 2.70)
293. **ParamTrigger** (Exploration / Indicators) - creates a trigger (button) in the parameter dialog (AFL 2.70)
294. **PDI** (Indicators) - plus directional movement indicator (AFL 1.3)
295. **Peak** (Basic price pattern detection) - peak (AFL 1.1)
296. **PeakBars** (Basic price pattern detection) - bars since peak (AFL 1.1)
297. **Percentile** (Statistical functions) - calculate percentile (AFL 2.5)
298. **PercentRank** (Indicators) - calculate percent rank (AFL 3.40)
299. **PlaySound** (Miscellaneous functions) - play back specified .WAV file (AFL 3.40)

300. **Plot** (Exploration / Indicators) - plot indicator graph (AFL 1.8)
301. **PlotForeign** (Referencing other symbol data) - plot foreign security data (AFL 2.2)
302. **PlotGrid** (Exploration / Indicators) - Plot horizontal grid line (AFL 2.3)
303. **PlotOHLC** (Exploration / Indicators) - plot custom OHLC chart (AFL 2.2)
304. **PlotShapes** (Exploration / Indicators) - plots arrows and other shapes (AFL 2.3)
305. **PlotText** (Indicators) - write text on the chart (AFL 2.80)
306. **PlotTextSetFont** (Basic price pattern detection) - write text on the chart with user-defined font (AFL 2.80)
307. **PlotVAPOverlay** (Exploration / Indicators) - plot Volume-At-Price overlay chart (AFL 2.4)
308. **PlotVAPOverlayA** (Indicators) - plot multiple-segment Volume-At-Price chart (AFL 3.20)
309. **PopupWindow** (Miscellaneous functions) - display pop-up window (AFL 3.0)
310. **Prec** (Math functions) - adjust number of decimal points of floating point number
311. **Prefs** (Miscellaneous functions) - retrieve preferences settings (AFL 1.4)
312. **PriceVolDistribution** (Matrix functions) - general-purpose distribution function (AFL 4.20)
313. **printf** (String manipulation) - Print formatted output to the output window. (AFL 2.5)
314. **Prod** (Moving averages, summation) - cumulative product of array over specified range (AFL 4.20)
315. **ProdSince** (Moving averages, summation) - cumulative product since condition is met (AFL 4.20)
316. **PVI** (Indicators) - positive volume index
317. **Random** (Statistical functions) - random number (AFL 1.9)
318. **Ref** (Trading system toolbox) - reference past/future values of the array
319. **RelStrength** (Referencing other symbol data) - comparative relative strength (AFL 1.3)
320. **Remap** (Math functions) - re-maps one range to another (AFL 4.30)
321. **RequestMouseMoveRefresh** (Indicators) - request chart to be refreshed when user moves mouse cursor (AFL 4.30)
322. **RequestTimedRefresh** (Indicators) - forces periodical refresh of indicator pane (AFL 2.90)
323. **RestorePriceArrays** (Referencing other symbol data) - restore price arrays to original symbol (AFL 2.5)
324. **Reverse** (Miscellaneous functions) - reverse the order of the elements in the array (AFL 3.90)
325. **RMI** (Indicators) - Relative Momentum Index (AFL 2.1)
326. **ROC** (Indicators) - percentage rate of change
327. **Round** (Math functions) - round number to nearest integer
328. **RSI** (Indicators) - relative strength index
329. **RWI** (Indicators) - random walk index
330. **RWIHi** (Indicators) - random walk index of highs
331. **RWILo** (Indicators) - random walk index of lows
332. **SafeDivide** (Math functions) - division with divide-by-zero protection (AFL 4.40)
333. **SAR** (Indicators) - parabolic stop-and-reverse (AFL 1.3)
334. **Say** (Text-to-Speech functions) - speaks provided text (AFL 2.90)
335. **Second** (Date/Time) - get current bar's second (AFL 2.0)
336. **SectorID** (Information / Categories) - get sector ID / name (AFL 1.8)
337. **SelectedValue** (Exploration / Indicators) - retrieves value of the array at currently selected date/time point (AFL 2.1)
338. **SendEmail** (Miscellaneous functions) - send an e-mail message (AFL 3.90)
339. **SetBacktestMode** (Trading system toolbox) - Sets working mode of the backtester (AFL 3.0)
340. **SetBarFillColor** (Indicators) - set bar/candlestick/cloud chart fill color (AFL 3.1)
341. **SetBarsRequired** (Miscellaneous functions) - set number of previous and future bars needed for script/DLL to properly execute (AFL 2.1)
342. **SetChartBkColor** (Indicators) - set background color of a chart (AFL 2.80)
343. **SetChartBkGradientFill** (Indicators) - enables background gradient color fill in indicators (AFL 2.90)
344. **SetChartOptions** (Exploration / Indicators) - set/clear/overwrite defaults for chart pane options (AFL 2.70)

345. **SetCustomBacktestProc** (Trading system toolbox) - define custom backtest procedure formula file (AFL 2.70)
346. **SetForeign** (Referencing other symbol data) - replace current price arrays with those of foreign security (AFL 2.5)
347. **SetFormulaName** (Trading system toolbox) - set the name of the formula (AFL 2.5)
348. **SetGradientFill** (Indicators) - set the colors of a gradient fill plot (AFL 3.60)
349. **SetOption** (Trading system toolbox) - sets options in automatic analysis settings (AFL 2.3)
350. **SetPositionSize** (Trading system toolbox) - set trade size (AFL 2.70)
351. **SetSortColumns** (Exploration / Indicators) - sets the columns which will be used for sorting in AA window (AFL 2.90)
352. **SetStopPrecedence** (Trading system toolbox) - set precedence of built-in stops (AFL 4.0)
353. **SetTradeDelays** (Trading system toolbox) - allows to control trade delays applied by the backtester (AFL 2.1)
354. **ShellExecute** (Basic price pattern detection) - execute a file (AFL 3.40)
355. **sign** (Math functions) - returns the sign of the number/array (AFL 2.50)
356. **Signal** (Indicators) - macd signal line
357. **sin** (Math functions) - sine function
358. **sinh** (Math functions) - hyperbolic sine function (AFL 2.80)
359. **Skewness** (Math functions) - calculate skewness (AFL 4.20)
360. **Sort** (Miscellaneous functions) - performs a quick sort of the array (AFL 3.90)
361. **SparseCompress** (Miscellaneous functions) - compress sparse array (AFL 4.0)
362. **SparseExpand** (Miscellaneous functions) - expand compressed array to sparse array (AFL 4.0)
363. **SparseInterpolate** (Miscellaneous functions) - interpolate values between sparse points given as input (AFL 4.90)
364. **sqrt** (Math functions) - square root
365. **StaticVarAdd** (Miscellaneous functions) - an "atomic" addition for static variables (AFL 4.10)
366. **StaticVarCompareExchange** (Miscellaneous functions) - atomic interlocked static variable compare-exchange operation (AFL 3.50)
367. **StaticVarCount** (Miscellaneous functions) - get the total number of static variables in memory (AFL 3.30)
368. **StaticVarGenerateRanks** (Miscellaneous functions) - generate ranking of multiple symbols and store it to static variables (AFL 3.70)
369. **StaticVarGet** (Miscellaneous functions) - gets the value of static variable (AFL 2.60)
370. **StaticVarGetRankedSymbols** (Miscellaneous functions) - retrieve a list of ranked symbols from static variables (AFL 3.70)
371. **StaticVarGetText** (Miscellaneous functions) - gets the value of static variable as string (AFL 2.60)
372. **StaticVarInfo** (Miscellaneous functions) - get the information about static variable(s) (AFL 3.60)
373. **StaticVarRemove** (Miscellaneous functions) - remove static variable (AFL 2.80)
374. **StaticVarSet** (Miscellaneous functions) - sets the value of static variable (AFL 2.60)
375. **StaticVarSetText** (Miscellaneous functions) - Sets the value of static string variable. (AFL 2.60)
376. **Status** (Miscellaneous functions) - get run-time AFL status information (AFL 1.65)
377. **StdErr** (Statistical functions) - standard error (AFL 1.4)
378. **StDev** (Statistical functions) - standard deviation (AFL 1.4)
379. **StochD** (Indicators) - stochastic slow %D
380. **StochK** (Indicators) - stochastic slow %K
381. **StrCount** (String manipulation) - count the occurrences of substring within a string (AFL 3.20)
382. **StrExtract** (String manipulation) - extracts given item (substring) from comma-separated string (AFL 2.4)
383. **StrFind** (String manipulation) - find substring in a string (AFL 2.5)
384. **StrFormat** (String manipulation) - Write formatted output to the string (AFL 2.5)
385. **StrLeft** (String manipulation) - extracts the leftmost part (AFL 2.0)
386. **StrLen** (String manipulation) - string length (AFL 1.5)

387. **StrMatch** (String manipulation) - string pattern/wildcard matching (AFL 4.0)
388. **StrMid** (String manipulation) - extracts part of the string (AFL 2.0)
389. **StrReplace** (String manipulation) - string replace (AFL 2.90)
390. **StrRight** (String manipulation) - extracts the rightmost part of the string (AFL 2.0)
391. **StrSort** (String manipulation) - sort comma-separated item list (AFL 3.90)
392. **StrToDateTime** (String manipulation) - convert string to datetime (AFL 2.80)
393. **StrToLower** (String manipulation) - convert to lowercase (AFL 2.80)
394. **StrToNum** (String manipulation) - convert string to number (AFL 2.5)
395. **StrToUpper** (String manipulation) - convert to uppercase (AFL 2.80)
396. **StrTrim** (String manipulation) - trim whitespaces from the string (AFL 3.90)
397. **Study** (Miscellaneous functions) - reference hand-drawn study (AFL 1.5)
398. **Sum** (Moving averages, summation) - sum data over specified number of bars
399. **SumSince** (Moving averages, summation) - sum of array elements since condition was tru (AFL 4.10)
400. **tan** (Math functions) - tangent function (AFL 1.0)
401. **tanh** (Math functions) - hyperbolic tangent function (AFL 2.80)
402. **TEMA** (Moving averages, summation) - triple exponential moving average (AFL 2.0)
403. **ThreadSleep** (Miscellaneous functions) - suspend thread for specified number of milliseconds (AFL 3.50)
404. **TimeFrameCompress** (Time Frame functions) - compress single array to given time frame (AFL 2.5)
405. **TimeFrameExpand** (Time Frame functions) - expand time frame compressed array (AFL 2.5)
406. **TimeFrameGetPrice** (Time Frame functions) - retrieve O, H, L, C, V values from other time frame (AFL 2.5)
407. **TimeFrameMode** (Time Frame functions) - switch time frame compression mode (AFL 2.80)
408. **TimeFrameRestore** (Time Frame functions) - restores price arrays to original time frame (AFL 2.5)
409. **TimeFrameSet** (Time Frame functions) - switch price arrays to a different time frame (AFL 2.5)
410. **TimeNum** (Date/Time) - get current bar time (AFL 2.0)
411. **TrimResultRows** (Exploration / Indicators) - trims Analysis result list to specified number of rows (AFL 4.90)
412. **Trin** (Composites) - traders (Arms) index (AFL 1.2)
413. **TRIX** (Indicators) - triple exponential smoothed price
414. **Trough** (Basic price pattern detection) - trough (AFL 1.1)
415. **TroughBars** (Basic price pattern detection) - bars since trough (AFL 1.1)
416. **TSF** (Statistical functions) - time series forecast (AFL 2.2)
417. **Ultimate** (Indicators) - ultimate oscillator
418. **UncIssues** (Composites) - unchanged issues (AFL 1.2)
419. **UncVolume** (Composites) - unchaged issues volume (AFL 1.2)
420. **ValueWhen** (Trading system toolbox) - get value of the array when condition met (AFL 1.1)
421. **VarGet** (Miscellaneous functions) - gets the value of dynamic variable (AFL 2.60)
422. **VarGetText** (Miscellaneous functions) - gets the text value of dynamic variable (AFL 2.80)
423. **VarSet** (Miscellaneous functions) - sets the value of dynamic variable (AFL 2.60)
424. **VarSetText** (Miscellaneous functions) - sets dynamic variable of string type (AFL 2.80)
425. **Version** (Miscellaneous functions) - get version info (AFL 1.9)
426. **VoiceCount** (Text-to-Speech functions) - get number of SAPI voices (AFL 4.20)
427. **VoiceSelect** (Text-to-Speech functions) - select SAPI voice (AFL 4.20)
428. **VoiceSetRate** (Text-to-Speech functions) - sets voice speech rate (AFL 4.30)
429. **VoiceSetVolume** (Text-to-Speech functions) - set the volume of speech (AFL 4.30)
430. **VoiceWaitUntilDone** (Text-to-Speech functions) - waits until TTS voice has finished speaking (AFL 4.30)
431. **Wilders** (Moving averages, summation) - Wilder's smoothing (AFL 1.4)
432. **WMA** (Moving averages, summation) - weighted moving average (AFL 2.0)
433. **WriteIf** (Exploration / Indicators) - commentary conditional text output

434. **WriteVal** (Exploration / Indicators) - converts number to string
435. **XYChartAddPoint** (Exploration / Indicators) - add point to exploration scatter (XY) chart (AFL 3.60)
436. **XYChartSetAxis** (Exploration / Indicators) - set the names of X and Y axes in exploration scatter charts (AFL 3.60)
437. **Year** (Date/Time) - year (AFL 1.4)
438. **ZIG** (Basic price pattern detection) - zig-zag indicator (AFL 1.1)
439. **_DEFAULT_NAME** (Exploration / Indicators) - retrive default name of the plot (AFL 2.70)
440. **_DT** (Date/Time) - convert string to datetime (AFL 3.40)
441. **_exit** (Miscellaneous functions) - terminate the execution of AFL formula (AFL 4.40)
442. **_N** (Exploration / Indicators) - no text output (AFL 2.1)
443. **_PARAM_VALUES** (Exploration / Indicators) - retrieve param values string (AFL 2.70)
444. **_SECTION_BEGIN** (Exploration / Indicators) - section begin marker (AFL 2.70)
445. **_SECTION_END** (Exploration / Indicators) - section end marker (AFL 2.70)
446. **_SECTION_NAME** (Exploration / Indicators) - retrieve current section name (AFL 2.70)
447. **_TRACE** (Miscellaneous functions) - print text to system debug viewer (AFL 2.4)
448. **_TRACEF** (Miscellaneous functions) - print formatted text to system debug viewer (AFL 4.0)

AFL Function Reference - Categorized list of functions

Basic price pattern detection

- **FFT** - performs Fast Fourier Transform (AFL 2.90)
- **GapDown** - gap down
- **GapUp** - gap up
- **Inside** - inside day
- **Outside** - outside bar
- **Peak** - peak (AFL 1.1)
- **PeakBars** - bars since peak (AFL 1.1)
- **PlotTextSetFont** - write text on the chart with user-defined font (AFL 2.80)
- **ShellExecute** - execute a file (AFL 3.40)
- **Trough** - trough (AFL 1.1)
- **TroughBars** - bars since trough (AFL 1.1)
- **ZIG** - zig-zag indicator (AFL 1.1)

Composites

- **AddToComposite** - add value to composite ticker (AFL 2.0)
- **ADLine** - advance/decline line (AFL 1.2)
- **AdvIssues** - advancing issues (AFL 1.2)
- **AdvVolume** - advancing issues volume (AFL 1.2)
- **DecIssues** - declining issues (AFL 1.2)
- **DecVolume** - declining issues volume (AFL 1.2)
- **Trin** - traders (Arms) index (AFL 1.2)
- **UncIssues** - unchanged issues (AFL 1.2)
- **UncVolume** - unchaged issues volume (AFL 1.2)

Date/Time

- **BarIndex** - get zero-based bar number (AFL 2.3)
- **BeginValue** - Value of the array at the begin of the range (AFL 2.3)
- **Date** - date (AFL 1.1)

- **DateNum** - date number (AFL 1.4)
- **DateTime** - retrieves encoded date time (AFL 2.3)
- **DateTimeAdd** - adds specified number of seconds/minutes/hours/days to datetime (AFL 3.40)
- **DateTimeConvert** - date/time format conversion (AFL 2.90)
- **DateTimeDiff** - get difference in seconds between two datetime values (AFL 3.30)
- **DateTimeFormat** - converts datetime to string (AFL 4.20)
- **DateTimeToStr** - convert datetime to string (AFL 2.8)
- **Day** - day of month (AFL 1.4)
- **DayOfWeek** - day of week (AFL 1.4)
- **DayOfYear** - get ordinal number of day in a year (AFL 2.4)
- **DaysSince1900** - get number of days since January 1st, 1900 (AFL 3.20)
- **EndValue** - value of the array at the end of the selected range (AFL 2.3)
- **GetPlaybackDateTime** - get bar replay position date/time (AFL 3.0)
- **Hour** - get current bar's hour (AFL 2.0)
- **Interval** - get bar interval (in seconds) (AFL 2.1)
- **Lookup** - search the array for bar with specified date/time (AFL 3.40)
- **MicroSec** - get bar's microsecond part of the timestamp
- **MilliSec** - get bar's millisecond part of the timestamp
- **Minute** - get current bar's minute (AFL 2.0)
- **Month** - month (AFL 1.4)
- **Now** - gets current system date/time (AFL 2.3)
- **Second** - get current bar's second (AFL 2.0)
- **TimeNum** - get current bar time (AFL 2.0)
- **Year** - year (AFL 1.4)
- **_DT** - convert string to datetime (AFL 3.40)

Indicators

- **AccDist** - accumulation/distribution
- **ADX** - average directional movement index (AFL 1.3)
- **ATR** - average true range (AFL 1.3)
- **BBandBot** - bottom bollinger band
- **BBandTop** - top bollinger band
- **CCI** - commodity channel index
- **Chaikin** - chaikin oscillator
- **ColorBlend** - blends (mixes) two colors (AFL 3.30)
- **FirstVisibleValue** - get first visible value of array (AFL 3.40)
- **GetChartBkColor** - get the RGB color value of chart background (AFL 3.20)
- **GetCursorMouseButtons** - get current state of mouse buttons (AFL 2.80)
- **GetCursorXPosition** - get current X position of mouse pointer (AFL 2.80)
- **GetCursorYPosition** - get current Y position of mouse pointer (AFL 2.80)
- **HighestVisibleValue** - get the highest value within visible chart area (AFL 3.30)
- **LastVisibleValue** - get last visible value of array (AFL 3.40)
- **LowestVisibleValue** - get the lowest value within visible chart area (AFL 3.30)
- **MACD** - moving average convergence/divergence
- **MDI** - minus directional movement indicator (-DI) (AFL 1.3)
- **MFI** - money flow index
- **NVI** - negative volume index
- **OBV** - on balance volume
- **OscP** - price oscillator
- **OscV** - volume oscillator
- **PDI** - plus directional movement indicator (AFL 1.3)

- **PercentRank** - calculate percent rank (AFL 3.40)
- **PlotText** - write text on the chart (AFL 2.80)
- **PlotVAPOverlayA** - plot multiple-segment Volume-At-Price chart (AFL 3.20)
- **PVI** - positive volume index
- **RequestMouseMoveRefresh** - request chart to be refreshed when user moves mouse cursor (AFL 4.30)
- **RequestTimedRefresh** - forces periodical refresh of indicator pane (AFL 2.90)
- **RMI** - Relative Momentum Index (AFL 2.1)
- **ROC** - percentage rate of change
- **RSI** - relative strength index
- **RWI** - random walk index
- **RWIHi** - random walk index of highs
- **RWILo** - random walk index of lows
- **SAR** - parabolic stop-and-reverse (AFL 1.3)
- **SetBarFillColor** - set bar/candlestick/cloud chart fill color (AFL 3.1)
- **SetChartBkColor** - set background color of a chart (AFL 2.80)
- **SetChartBkGradientFill** - enables background gradient color fill in indicators (AFL 2.90)
- **SetGradientFill** - set the colors of a gradient fill plot (AFL 3.60)
- **Signal** - macd signal line
- **StochD** - stochastic slow %D
- **StochK** - stochastic slow %K
- **TRIX** - triple exponential smoothed price
- **Ultimate** - ultimate oscillator

Information / Categories

- **CategoryAddSymbol** - adds a symbol to a category (AFL 2.5)
- **CategoryCreate** - add new category (such as watch list) (AFL 3.70)
- **CategoryFind** - search for category by name (AFL 3.0)
- **CategoryGetName** - get the name of a category (AFL 2.5)
- **CategoryGetSymbols** - retrieves comma-separated list of symbols belonging to given category (AFL 2.5)
- **CategoryRemoveSymbol** - remove a symbol from a category (AFL 2.5)
- **CategorySetName** - set the name of category (group, market, watch list, industry) (AFL 3.20)
- **FullName** - full name of the symbol (AFL 1.1)
- **GetCategorySymbols** - retrieves comma-separated list of symbols belonging to given category (AFL 2.4)
- **GetDatabaseName** - retrieves folder name of current database (AFL 2.3)
- **GetFnData** - get fundamental data (AFL 2.90)
- **GetFnDataForeign** - get fundamental data for specified symbol (AFL 4.20)
- **GicsID** - get GICS category information (AFL 3.40)
- **GroupID** - get group ID/name (AFL 1.8)
- **IcbID** - get ICB category information (AFL 3.60)
- **IndustryID** - get industry ID / name (AFL 1.8)
- **InGICS** - test GICS membership (AFL 3.40)
- **InICB** - test ICB membership (AFL 3.60)
- **InWatchList** - watch list membership test (by ordinal number)
- **InWatchListName** - watch list membership test (by name) (AFL 3.0)
- **IsContinuous** - checks 'continuous quotations' flag state (AFL 2.60)
- **IsFavorite** - check if current symbol belongs to favorites (AFL 2.5)
- **IsIndex** - check if current symbol is an index (AFL 2.5)
- **MarketID** - market ID / name (AFL 1.8)

- **Name** - ticker symbol (AFL 1.1)
- **SectorID** - get sector ID / name (AFL 1.8)

Lowest/Highest

- **HHV** - highest high value
- **HHVBars** - bars since highest high
- **Highest** - highest value
- **HighestBars** - bars since highest value
- **HighestSince** - highest value since condition met (AFL 1.4)
- **HighestSinceBars** - bars since highest value since condition met (AFL 1.4)
- **LLV** - lowest low value
- **LLVBars** - bars since lowest low
- **Lowest** - lowest value
- **LowestBars** - bars since lowest
- **LowestSince** - lowest value since condition met (AFL 1.4)
- **LowestSinceBars** - barssince lowest value since condition met (AFL 1.4)

Math functions

- **abs** - absolute value
- **acos** - arccosine function
- **AlmostEqual** - rounding error insensitive comparison (AFL 2.80)
- **asin** - arcsine function
- **atan** - arc tan
- **atan2** - calculates arctangent of y/x (AFL 2.90)
- **ceil** - ceil value
- **cos** - cosine
- **cosh** - hyperbolic cosine function (AFL 2.80)
- **erf** - calculates Gauss error function (erf) (AFL 4.40)
- **EXP** - exponential function
- **floor** - floor value
- **frac** - fractional part
- **Int** - integer part
- **inverf** - inverse Gauss erf function (AFL 4.40)
- **Kurtosis** - calculates kurtosis (AFL 4.20)
- **log** - natural logarithm
- **log10** - decimal logarithm
- **Max** - maximum value of two numbers / arrays
- **Min** - minimum value of two numbers / arrays
- **Prec** - adjust number of decimal points of floating point number
- **Remap** - re-maps one range to another (AFL 4.30)
- **Round** - round number to nearest integer
- **SafeDivide** - division with divide-by-zero protection (AFL 4.40)
- **sign** - returns the sign of the number/array (AFL 2.50)
- **sin** - sine function
- **sinh** - hyperbolic sine function (AFL 2.80)
- **Skewness** - calculate skewness (AFL 4.20)
- **sqrt** - square root
- **tan** - tangent function (AFL 1.0)
- **tanh** - hyperbolic tangent function (AFL 2.80)

Miscellaneous functions

- **#include** - preprocessor include command (AFL 2.2)
- **#include_once** - preprocessor include (once) command (AFL 2.70)
- **#pragma** - sets AFL pre-processor option (AFL 2.4)
- **ClipboardGet** - retrieves current contents of Windows clipboard (AFL 2.60)
- **ClipboardSet** - copies the text to the Windows clipboard (AFL 2.6)
- **ColorHSB** - specify color using Hue-Saturation-Brightness (AFL 2.80)
- **ColorRGB** - specify color using Red-Green-Blue components (AFL 2.80)
- **CreateObject** - create COM object (AFL 1.8)
- **CreateStaticObject** - create static COM object (AFL 1.8)
- **EnableScript** - enable scripting engine
- **EnableTextOutput** - allows to enable or disable text output (AFL 2.20)
- **Error** - displays user-defined error message and stops the execution (AFL 3.7)
- **FindIndex** - find index of array item matching specified value (AFL 4.40)
- **GetAsyncKeyState** - query the current state of keyboard keys (AFL 3.60)
- **GetExtraData** - get extra data from external data source (AFL 1.9)
- **GetExtraDataForeign** - get extra data from external data source for specified symbol (AFL 4.20)
- **GetFormulaPath** - get file path of current formula (AFL 3.90)
- **GetLastOSError** - get text of last operating system (Windows) error (AFL 4.30)
- **GetObject** - get handle to already running OLE object (AFL 4.40)
- **GetPerformanceCounter** - retrieves the current value of the high-resolution performance counter (AFL 2.90)
- **GetRTData** - retrieves the real-time data fields (AFL 2.60)
- **GetRTDataForeign** - retrieves the real-time data fields (for specified symbol) (AFL 2.80)
- **GetScriptObject** - get access to script COM object (AFL 1.8)
- **IsEmpty** - empty value check (AFL 1.5)
- **IsFinite** - check if value is not infinite (AFL 2.3)
- **IsNan** - checks for NaN (not a number) (AFL 2.3)
- **IsNull** - check for Null (empty) value (AFL 2.3)
- **IsTrue** - true value (non-empty and non-zero) check (AFL 1.5)
- **NoteGet** - retrieves the text of the note (AFL 2.6)
- **NoteSet** - sets text of the note (AFL 2.6)
- **NullCount** - count consecutive Null values (AFL 3.90)
- **Nz** - Null (Null/Nan/Infinity) to zero (AFL 2.3)
- **PlaySound** - play back specified .WAV file (AFL 3.40)
- **PopupWindow** - display pop-up window (AFL 3.0)
- **Prefs** - retrieve preferences settings (AFL 1.4)
- **Reverse** - reverse the order of the elements in the array (AFL 3.90)
- **SendEmail** - send an e-mail message (AFL 3.90)
- **SetBarsRequired** - set number of previous and future bars needed for script/DLL to properly execute (AFL 2.1)
- **Sort** - performs a quick sort of the array (AFL 3.90)
- **SparseCompress** - compress sparse array (AFL 4.0)
- **SparseExpand** - expand compressed array to sparse array (AFL 4.0)
- **SparseInterpolate** - interpolate values between sparse points given as input (AFL 4.90)
- **StaticVarAdd** - an "atomic" addition for static variables (AFL 4.10)
- **StaticVarCompareExchange** - atomic interlocked static variable compare-exchange operation (AFL 3.50)
- **StaticVarCount** - get the total number of static variables in memory (AFL 3.30)
- **StaticVarGenerateRanks** - generate ranking of multiple symbols and store it to static variables (AFL 3.70)

- **StaticVarGet** - gets the value of static variable (AFL 2.60)
- **StaticVarGetRankedSymbols** - retrieve a list of ranked symbols from static variables (AFL 3.70)
- **StaticVarGetText** - gets the value of static variable as string (AFL 2.60)
- **StaticVarInfo** - get the information about static variable(s) (AFL 3.60)
- **StaticVarRemove** - remove static variable (AFL 2.80)
- **StaticVarSet** - sets the value of static variable (AFL 2.60)
- **StaticVarSetText** - Sets the value of static string variable. (AFL 2.60)
- **Status** - get run-time AFL status information (AFL 1.65)
- **Study** - reference hand-drawn study (AFL 1.5)
- **ThreadSleep** - suspend thread for specified number of milliseconds (AFL 3.50)
- **VarGet** - gets the value of dynamic variable (AFL 2.60)
- **VarGetText** - gets the text value of dynamic variable (AFL 2.80)
- **VarSet** - sets the value of dynamic variable (AFL 2.60)
- **VarSetText** - sets dynamic variable of string type (AFL 2.80)
- **Version** - get version info (AFL 1.9)
- **_exit** - terminate the execution of AFL formula (AFL 4.40)
- **_TRACE** - print text to system debug viewer (AFL 2.4)
- **_TRACEF** - print formatted text to system debug viewer (AFL 4.0)

Moving averages, summation

- **AMA** - adaptive moving average (AFL 1.5)
- **AMA2** - adaptive moving average (AFL 1.5)
- **Cum** - cumulative sum
- **CumProd** - cumulative product of all array elements (AFL 4.20)
- **DEMA** - double exponential moving average (AFL 2.0)
- **EMA** - exponential moving average
- **FIR** - Finite Impulse Response filter (AFL 3.40)
- **HMA** - Hull Moving Average (AFL 3.40)
- **IIR** - infinite impulse response filter (AFL 4.0)
- **MA** - simple moving average
- **Prod** - cumulative product of array over specified range (AFL 4.20)
- **ProdSince** - cumulative product since condition is met (AFL 4.20)
- **Sum** - sum data over specified number of bars
- **SumSince** - sum of array elements since condition was tru (AFL 4.10)
- **TEMA** - triple exponential moving average (AFL 2.0)
- **Wilders** - Wilder's smoothing (AFL 1.4)
- **WMA** - weighted moving average (AFL 2.0)

Statistical functions

- **Correlation** - correlation (AFL 1.4)
- **LinearReg** - linear regression end-point (AFL 2.2)
- **LinRegIntercept** - (AFL 2.2)
- **LinRegSlope** - linear regression slope (AFL 1.4)
- **Median** - calculate median (middle element) (AFL 2.5)
- **mtRandom** - Mersene Twister random number generator (AFL 3.0)
- **mtRandomA** - Mersene Twister random number generator (array version) (AFL 3.0)
- **NormDist** - normal distribution function (AFL 4.20)
- **Percentile** - calculate percentile (AFL 2.5)
- **Random** - random number (AFL 1.9)
- **StdErr** - standard error (AFL 1.4)

- **StDev** - standard deviation (AFL 1.4)
- **TSF** - time series forecast (AFL 2.2)

String manipulation

- **Asc** - get ASCII code of character (AFL 2.80)
- **Chr** - get string with given ASCII code (AFL 4.40)
- **NumToStr** - convert number to string (AFL 2.5)
- **printf** - Print formatted output to the output window. (AFL 2.5)
- **StrCount** - count the occurrences of substring within a string (AFL 3.20)
- **StrExtract** - extracts given item (substring) from comma-separated string (AFL 2.4)
- **StrFind** - find substring in a string (AFL 2.5)
- **StrFormat** - Write formatted output to the string (AFL 2.5)
- **StrLeft** - extracts the leftmost part (AFL 2.0)
- **StrLen** - string length (AFL 1.5)
- **StrMatch** - string pattern/wildcard matching (AFL 4.0)
- **StrMid** - extracts part of the string (AFL 2.0)
- **StrReplace** - string replace (AFL 2.90)
- **StrRight** - extracts the rightmost part of the string (AFL 2.0)
- **StrSort** - sort comma-separated item list (AFL 3.90)
- **StrToDateTime** - convert string to datetime (AFL 2.80)
- **StrToLower** - convert to lowercase (AFL 2.80)
- **StrToNum** - convert string to number (AFL 2.5)
- **StrToUpper** - convert to uppercase (AFL 2.80)
- **StrTrim** - trim whitespaces from the string (AFL 3.90)

Trading system toolbox

- **AlertIf** - trigger alerts (AFL 2.1)
- **ApplyStop** - apply built-in stop (AFL 1.7)
- **BarsSince** - bars since
- **BarsSinceCompare** - bars since comparision between past and present array values was true (AFL 4.40)
- **Cross** - crossover check
- **EnableRotationalTrading** - Turns on rotational-trading mode of the backtester (AFL 2.5)
- **Equity** - calculate single-symbol equity line (AFL 2.0)
- **ExRem** - remove excessive signals (AFL 1.5)
- **ExRemSpan** - remove excessive signals spanning given number of bars (AFL 2.0)
- **Flip** - (AFL 1.5)
- **GetBacktesterObject** - get the access to backtester object (AFL 2.60)
- **GetOption** - gets the value of option in automatic analysis settings (AFL 2.60)
- **GetTradingInterface** - retrieves OLE automation object to automatic trading interfac (AFL 2.70)
- **Hold** - hold the alert signal
- **IIf** - immediate IF function
- **LastValue** - last value of the array
- **Optimize** - define optimization variable (AFL 1.7)
- **OptimizerSetEngine** - select external optimization engine (AFL 3.20)
- **OptimizerSetOption** - set the value of external optimizer engine parameter (AFL 3.20)
- **Ref** - reference past/future values of the array
- **SetBacktestMode** - Sets working mode of the backtester (AFL 3.0)
- **SetCustomBacktestProc** - define custom backtest procedure formula file (AFL 2.70)
- **SetFormulaName** - set the name of the formula (AFL 2.5)

- **SetOption** - sets options in automatic analysis settings (AFL 2.3)
- **SetPositionSize** - set trade size (AFL 2.70)
- **SetStopPrecedence** - set precedence of built-in stops (AFL 4.0)
- **SetTradeDelays** - allows to control trade delays applied by the backtester (AFL 2.1)
- **ValueWhen** - get value of the array when condition met (AFL 1.1)

Exploration / Indicators

- **AddColumn** - add numeric exploration column (AFL 1.8)
- **AddMultiTextColumn** - adds exploration text column based on array (AFL 4.20)
- **AddRankColumn** - add ranking column(s) according to current sort set by SetSortColumns (AFL 5.70)
- **AddRow** - add raw text row to exploration (AFL 4.0)
- **AddSummaryRows** - add summary row(s) to the exploration output (AFL 3.2)
- **AddTextColumn** - add text exploration column (AFL 1.8)
- **EncodeColor** - encodes color for indicator title (AFL 2.2)
- **GetChartID** - get current chart ID (AFL 2.3)
- **GetPriceStyle** - get current price chart style (AFL 2.70)
- **LineArray** - generate trend-line array (AFL 2.5)
- **Param** - add user user-definable numeric parameter (AFL 2.3)
- **ParamColor** - add user user-definable color parameter (AFL 2.3)
- **ParamDate** - add user user-definable date parameter (AFL 2.60)
- **ParamField** - creates price field parameter (AFL 2.70)
- **ParamList** - creates the parameter that consist of the list of choices (AFL 2.70)
- **ParamStr** - add user user-definable string parameter (AFL 2.3)
- **ParamStyle** - select styles applied to the plot (AFL 2.70)
- **ParamTime** - add user user-definable time parameter (AFL 2.60)
- **ParamToggle** - create Yes/No parameter (AFL 2.70)
- **ParamTrigger** - creates a trigger (button) in the parameter dialog (AFL 2.70)
- **Plot** - plot indicator graph (AFL 1.8)
- **PlotGrid** - Plot horizontal grid line (AFL 2.3)
- **PlotOHLC** - plot custom OHLC chart (AFL 2.2)
- **PlotShapes** - plots arrows and other shapes (AFL 2.3)
- **PlotVAPOverlay** - plot Volume-At-Price overlay chart (AFL 2.4)
- **SelectedValue** - retrieves value of the array at currently selected date/time point (AFL 2.1)
- **SetChartOptions** - set/clear/overwrite defaults for chart pane options (AFL 2.70)
- **SetSortColumns** - sets the columns which will be used for sorting in AA window (AFL 2.90)
- **TrimResultRows** - trims Analysis result list to specified number of rows (AFL 4.90)
- **WriteIf** - commentary conditional text output
- **WriteVal** - converts number to string
- **XYChartAddPoint** - add point to exploration scatter (XY) chart (AFL 3.60)
- **XYChartSetAxis** - set the names of X and Y axes in exploration scatter charts (AFL 3.60)
- **_DEFAULT_NAME** - retrive default name of the plot (AFL 2.70)
- **_N** - no text output (AFL 2.1)
- **_PARAM_VALUES** - retrieve param values string (AFL 2.70)
- **_SECTION_BEGIN** - section begin marker (AFL 2.70)
- **_SECTION_END** - section end marker (AFL 2.70)
- **_SECTION_NAME** - retrieve current section name (AFL 2.70)

File Input/Output functions

- **fclose** - close a file (AFL 2.5)

- **fdelete** - deletes a file (AFL 2.70)
- **fdir** - list directory content (AFL 3.70)
- **feof** - test for end-of-file (AFL 2.5)
- **fgetcwd** - get current working directory (AFL 4.10)
- **fgets** - get a string from a file (AFL 2.5)
- **fgetstatus** - retrieves file status/properties (AFL 2.90)
- **fmkdir** - creates (makes) a directory (AFL 2.70)
- **fopen** - open a file (AFL 2.5)
- **fputs** - write a string to a file (AFL 2.5)
- **frmdir** - removes a directory (AFL 2.70)
- **InternetClose** - close Internet file handle (AFL 4.20)
- **InternetGetStatusCode** - returns HTTP status code of last Internet call (AFL 4.40)
- **InternetOpenURL** - opens Internet web resource (URL) (AFL 4.20)
- **InternetPostRequest** - send HTTP Post request to Internet web resource (URL) (AFL 4.30)
- **InternetReadString** - read a string from Internet resource (AFL 4.20)
- **InternetSetAgent** - set agent string for Internet function (AFL 4.30)
- **InternetSetHeaders** - set custom HTTP headers for subsequent web requests (AFL 4.40)
- **InternetSetOption** - set HTTP option for internet session (AFL 4.40)

GUI functions

- **GuiButton** - create on-chart button control (AFL 4.30)
- **GuiCheckBox** - creates on-chart checkbox control (AFL 4.30)
- **GuiDateTime** - creates on-chart date-time picker control (AFL 4.30)
- **GuiEdit** - create on-chart edit control (AFL 4.30)
- **GuiEnable** - enables or disables on-chart control (AFL 4.30)
- **GuiGetCheck** - get checked state of control (AFL 4.30)
- **GuiGetEvent** - get GUI event (AFL 4.30)
- **GuiGetText** - get text from on-chart control (AFL 4.30)
- **GuiGetValue** - get numeric value of on-chart control (AFL 4.30)
- **GuiRadio** - creates on-chart radio button control (AFL 4.30)
- **GuiSendKeyEvents** - request GUI notifications for specified keys (AFL 4.40)
- **GuiSetCheck** - set checked state of on-chart control (AFL 4.30)
- **GuiSetColors** - set colors for GUI controls (AFL 2.30)
- **GuiSetFont** - set the font for on-chart control (AFL 4.30)
- **GuiSetRange** - set slider control range (AFL 4.30)
- **GuiSetText** - set text value of on-chart control (AFL 4.30)
- **GuiSetValue** - set numeric value of on-chart control (AFL 4.30)
- **GuiSetVisible** - shows or hides on-chart control (AFL 4.30)
- **GuiSlider** - creates on-chart slider control (AFL 4.30)

Low-level graphics

- **GfxArc** - draw an arc (AFL 3.0)
- **GfxChord** - draw a chord (AFL 3.0)
- **GfxCircle** - draw a circle (AFL 3.0)
- **GfxDrawImage** - draw bitmap image (AFL 4.20)
- **GfxDrawText** - draw a text (clipped to rectangle) (AFL 3.0)
- **GfxEllipse** - draw an ellipse (AFL 3.0)
- **GfxFillSolidRect** - fill rectangle with solid color (AFL 4.10)
- **GfxGetTextWidth** - get pixel width of text (AFL 2.80)
- **GfxGradientRect** - draw a rectangle with gradient fill (AFL 3.0)

- **GfxLineTo** - draw a line to specified point (AFL 3.0)
- **GfxMoveTo** - move graphic cursor to new position (AFL 3.0)
- **GfxPie** - draw a pie (AFL 3.0)
- **GfxPolygon** - draw a polygon (AFL 3.0)
- **GfxPolyline** - draw a polyline (AFL 3.0)
- **GfxRectangle** - draw a rectangle (AFL 3.0)
- **GfxRoundRect** - draw a rectangle with rounded corners (AFL 3.0)
- **GfxSelectFont** - create / select graphic font (AFL 3.0)
- **GfxSelectHatchBrush** - select hatch style brush (AFL 4.0)
- **GfxSelectPen** - create / select graphic pen (AFL 3.0)
- **GfxSelectSolidBrush** - create / select graphic brush (AFL 3.0)
- **GfxSelectStockObject** - select built-in graphic object (AFL 4.00)
- **GfxSetBkColor** - set graphic background color (AFL 3.0)
- **GfxSetBkMode** - set graphic background mode (AFL 3.0)
- **GfxSetCoordsMode** - set low-level graphics co-ordinate mode (AFL 2.80)
- **GfxSetOverlayMode** - set low-level graphic overlay mode (AFL 3.0)
- **GfxSetPixel** - set pixel at specified position to specified color (AFL 3.0)
- **GfxSetTextAlign** - set text alignment (AFL 3.0)
- **GfxSetTextColor** - set graphic text color (AFL 3.0)
- **GfxSetZOrder** - set current low-level graphic Z-order layer (AFL 2.80)
- **GfxTextOut** - writes text at the specified location (AFL 3.0)

Map functions

- **MapCreate** - creates a map/dictionary object (holding key-value pairs) (AFL 4.90)

Matrix functions

- **Matrix** - create a new matrix (AFL 4.0)
- **MxCopy** - copy rectangular block from one matrix to another (AFL 4.40)
- **MxDet** - calculate determinant of the matrix (AFL 4.10)
- **MxFromString** - creates a new matrix out of string (AFL 4.10)
- **MxGetBlock** - get rectangular block of items from matrix (AFL 4.10)
- **MxGetSize** - get size of the matrix (AFL 4.0)
- **MxIdentity** - create an identity matrix (AFL 4.0)
- **MxInverse** - calculate inverse matrix (AFL 4.10)
- **MxSetBlock** - sets values in the rectangular block of matrix cells (AFL 4.10)
- **MxSolve** - solves linear equation system A @ X = B (AFL 4.10)
- **MxSort** - sorts the matrix (AFL 4.10)
- **MxSortRows** - sort the rows of the matrix (AFL 4.10)
- **MxSum** - calculate grand sum of the matrix (AFL 4.20)
- **MxToString** - convert matrix to string (AFL 4.10)
- **MxTranspose** - creates transpose of an input matrix (AFL 4.0)
- **PriceVolDistribution** - general-purpose distribution function (AFL 4.20)

Referencing other symbol data

- **Foreign** - access foreign security data (AFL 1.5)
- **GetBaseIndex** - retrieves symbol of relative strength base index (AFL 2.1)
- **PlotForeign** - plot foreign security data (AFL 2.2)
- **RelStrength** - comparative relative strength (AFL 1.3)
- **RestorePriceArrays** - restore price arrays to original symbol (AFL 2.5)

- **SetForeign** - replace current price arrays with those of foreign security (AFL 2.5)

Text-to-Speech functions

- **Say** - speaks provided text (AFL 2.90)
- **VoiceCount** - get number of SAPI voices (AFL 4.20)
- **VoiceSelect** - select SAPI voice (AFL 4.20)
- **VoiceSetRate** - sets voice speech rate (AFL 4.30)
- **VoiceSetVolume** - set the volume of speech (AFL 4.30)
- **VoiceWaitUntilDone** - waits until TTS voice has finished speaking (AFL 4.30)

Time Frame functions

- **TimeFrameCompress** - compress single array to given time frame (AFL 2.5)
- **TimeFrameExpand** - expand time frame compressed array (AFL 2.5)
- **TimeFrameGetPrice** - retrieve O, H, L, C, V values from other time frame (AFL 2.5)
- **TimeFrameMode** - switch time frame compression mode (AFL 2.80)
- **TimeFrameRestore** - restores price arrays to original time frame (AFL 2.5)
- **TimeFrameSet** - switch price arrays to a different time frame (AFL 2.5)

#include
- preprocessor include command

Miscellaneous functions
(AmiBroker 4.20)

SYNTAX **#include**

RETURNS nothing

FUNCTION Includes external AFL files into your formula. Note 1: include statement need SINGLE backslashes in the path (this is quite the opposite to normal AFL sting
parsing)
Note 2: using #include command may slow down formula execution even
considering the fact that AmiBroker tries to include only once and cache pre-processed text
Note 3: that currently no error message is given if #include fails and this code
is experimental.
Note 4: nesting #include commands is now supported (version 5.10 and above)
Note 5: by default files #included are cached by the AmiBroker. To turn off caching use
#pragma nocache
before any #include statements. #include now accepts new way of specifying file names to include:

#include <filename.afl>

(note < > braces instead of " ") if you specify the file name this way AmiBroker will look for the file in "standard include path" that is definable using new prefs setting in Tools->Preferences->AFL It makes much shorter to write includes and you can move include folder now without changing all AFL codes using #includes.

For example if you have set standard include path to "C:\AFL\MyIncludes" and write in your formula:

#include <common.afl>

AmiBroker will look for C:\AFL\MyIncludes\common.afl file

Also now #include reports file(s) not found in regular error message box.

EXAMPLE #include "C:\Program Files\AmiBroker\AFL\common.afl"

SEE ALSO #pragma() function

References:

The **#include** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

#include_once - preprocessor include (once) command

Miscellaneous functions
(AmiBroker 4.70)

SYNTAX	**#include "formula file path"**
RETURNS	nothing
FUNCTION	Includes external AFL files into your formula. Similar to #include but #include_once performs inclusion only once per formula. So if single formula has multiple #include_once commands for the same file (for example because of drag-and-drop overlay) it prevents syntax errors that could occur due to repeated definitions of functions in included file. More information can be found in #include command docs.
EXAMPLE	#include_once "myfile.afl"
SEE ALSO	#include() function

References:

The **#include_once** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

#pragma
- sets AFL pre-processor option

Miscellaneous functions
(AmiBroker 4.40)

SYNTAX **#pragma optionname**

RETURNS NOTHING

FUNCTION Sets various AFL pre-processor options. Pre-processor is a part of AFL engine that processes formulas BEFORE they are executed. #pragma allows to change pre-processor behaviour. Preprocessor is responsible for inclusion of external files via #include command. Pre-processor #pragmas can also affect behavior of Analysis window and declared static variable prefixes

Currently #pragma supports the following commands:

- **#pragma nocache** - causes that #included files are not cached so they are re-read with every execution

  #pragma nocache must be placed before any #include commands Note: between '#pragma' and 'nocache' there must be exactly SINGLE space
  Note 2: disabling caching may slow down execution of the formula (especially in indicators) !!!
- **#pragma maxthreads N** - causes Analysis window to limit number of parallel threads to N
- **#pragma enable_static_decl "prefix_string"** - enables static variable declarations with specified prefix
- **#pragma sequence(list_of_actions)** - defines sequence of actions to be performed automatically when you press "Run Sequence" button in the Analysis window

EXAMPLE

```
#pragma nocache
#include "myfile.afl"
```

SEE ALSO #include() function

References:

The **#pragma** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

abs - absolute value

Math functions

SYNTAX	**abs(NUMBER)**
	abs(ARRAY)

RETURNS	NUMBER ARRAY
FUNCTION	Calculates the absolute value of the NUMBER or ARRAY.
EXAMPLE	The formula "abs(-15)" will return +15; the formula "abs(15)" also returns +15.
SEE ALSO	

References:

The **abs** function is used in the following formulas in AFL on-line library:

- Absolute Breadth Index
- Adaptave Zones O/B & O/S Oscillator
- Adaptive Laguerre Filter, from John Ehlers
- ADXbuy
- ADXVMA
- Against all odds
- Analytic RSI formula
- Another FIb Level
- Application of Ehler filter
- AR_Prediction.afl
- Auto Trade Step by Step
- Auto-Optimization Framework
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Better Bollinger Bands
- Bullish Percent Index 2 files combined
- Candle Identification
- Candle Pattern Function
- Candle Stick Analysis
- CandleStick Comentary--Help needed
- Candlestick Commentary
- Candlestick Commentary Modified
- Candlestick Commentary-modified
- CandleStochastics
- CCT Kaleidoscope
- Chandelier Exit
- com-out
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Cybernertic Hilbert Sine Wave
- Demand Index
- Double top detection
- Dynamtic Momentum Index
- Ed Seykota's TSP: EMA Crossover System

- Ehlers Hilbert Transformer Indicator
- Elder Triple Screen Trading System
- elliott wave manual labelling
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- Halftrend
- Harmonic Patterns
- Head & Shoulders Pattern
- Heatmap V1
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- Hilbert Study
- Hurst Constant
- John Ehler
- Kiss and Touch with the Modified True Range
- MA Difference 20 Period
- MACD commentary
- MACD indicator display
- Market Breadth Chart-In-Chart
- Market Direction
- MAVG
- MDYtoXLSerialDays and XLSerialDaysToDateNum
- Modified Head & Shoulder Pattern
- MultiCycle 1.0
- Multiple Ribbon Demo
- Multiple sinus noised
- Nadaraya-Watson Envelope
- Nonlinear Ehlers Filter
- nth (1 - 8) Order Polynomial Fit
- Option Calls, Puts and days till third friday.
- Parametric Chande Trendscore
- Pattern Recognition Exploration
- Periodically ReBalance a BUY & HOLD Portfolio
- prakash
- Range Expansion Index
- Range Filter - Trading Strategy
- Raw ADX
- Rebalancing Backtest avoiding leverage
- RSI of Weekly Price Array
- SectorRSI
- Sine Wave Indicator
- TD REI
- The Mean RSIt
- The Mean RSIt (variations)
- tomy_frenchy
- Triangle exploration using P&F Chart
- Trigonometric Fit - TrigFit with AR for cos / sin
- Tushar Chande's Projected Range
- TWS auto-export Executions-file parser

- ValueChart
- Vertical Horizontal Filter
- Vertical Horizontal Filter (VHF)
- Vic Huebner
- Visi-Trade
- Volatility Quality Index
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

AccDist - accumulation/distribution

Indicators

SYNTAX	**AccDist()**
RETURNS	ARRAY
FUNCTION	Calculates the Accumulation/ Distribution indicator.
EXAMPLE	
SEE ALSO	

References:

The **AccDist** function is used in the following formulas in AFL on-line library:

- accum/dist mov avg crossover SAR
- Bollinger band normalization

More information:

See updated/extended version on-line.

acos
- arccosine function

Math functions

SYNTAX	**acos(x)**
RETURNS	NUMBER, ARRAY
FUNCTION	Returns the arccosine of NUMBER or ARRAY. The acos function returns the arccosize of x in the range 0 to pi radians. If x is less than -1 or greater than 1, acos returns an indefinite.
EXAMPLE	
SEE ALSO	COS() function

References:

The **acos** function is used in the following formulas in AFL on-line library:

- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots

More information:

See updated/extended version on-line.

AddColumn
- add numeric exploration column

Exploration / Indicators
(AmiBroker 3.80)

SYNTAX **AddColumn(array, name, format = 1.2, textColor = colorDefault, bkgndColor = colorDefault, width = -1, barchart = Null)**

RETURNS NOTHING

FUNCTION Adds a new column to the exploration result list. The column shows *array* values and has a caption of *name.* The values are formatted using *format* specification.
By default all variables are displayed with 2 decimal digits, but you can change this by assigning a different value to this variable: 1.5 gives 5 decimal digits, 1.0 gives no decimal digits. (Note for advanced users: the integer part of this number can be used to pad formatted number with spaces - 6.0 will give no decimal digits but a number space-padded upto 6 characters.)
Next two parameters allow to modify text and background color.

special format constants:

- formatDateTime - produces date time formated according to your system settings
```
 AddColumn( DateTime(), "Date / Time", formatDateTime );
```

- formatDateTimeISO - produces date time formated using ISO standard, i.e. YYYY-MM-DD HH:MM:SS
```
 AddColumn( DateTime(), "Date / Time", formatDateTimeISO );
```

- formatChar - allows outputting single ASCII character codes:
  Example (produces signal file accepted by various other programs):
```
 Buy=Cross(MACD(),Signal());
Sell=Cross(Signal(), MACD());
Filter=Buy OR Sell;
SetOption("NoDefaultColumns", True );
AddColumn( DateTime(), "Date", formatDateTime );
AddColumn( IIf( Buy, 66, 83 ), "Signal", formatChar );
```
- width parameter allows to control pixel width of the column
- 'barchart' parameter accepts values from 0...100 represening percentage width of bar chart displayed in a cell the in-cell bar chart is drawn with bkcolor (background color).

EXAMPLE 1. Simple column showing close price

```
addcolumn( Close, "Close price", 1.4 );
```

2. Colorful output

```
Filter =1;

AddColumn( Close, "Close", 1.2 );
AddColumn( MACD(), "MACD", 1.4 , IIf( MACD() > 0, colorGreen,
colorRed ) );
AddTextColumn( FullName(), "Full name", 77 , colorDefault, IIf(
Close < 10, colorLightBlue, colorDefault ) );
```

3. Barchart example **Filter**=1;

```
AddColumn( Close, "Close" );
rank = PercentRank( Close, 100 );
Color = ColorHSB( rank * 64/100, 255, 255 );
AddColumn( rank, "100-day percent rank", 1.2, colorDefault, Color,
-1, rank );
```

SEE ALSO ADDTEXTCOLUMN() function

References:

The **AddColumn** function is used in the following formulas in AFL on-line library:

- Advanced MA system
- ADXbuy
- AFL Example
- AFL Example - Enhanced
- Alert Output As Quick Rewiev
- Appel's ROC or The Triple Momentum Timing Model
- Aroon Indicators
- Auto-Optimization Framework
- AutoTrade using an Exploration
- Average Dollar Price Volatility Exploration
- BBAreacolor&TGLCROSSNEW
- Black Scholes Option Pricing
- Bollinger Band Squeeze
- Bottom Fisher Exploration
- Bull Fear / Bear Fear
- Bullish Percent Index 2 files combined
- Calculate composites for tickers in list files
- CAMSLIM Cup and Handle Pattern AFL
- Commodity Selection Index (CSI)
- Count Tickers in Watchlist
- CVR--severe filter
- Darvas Amibroker
- Darvas Johndeo Research
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- ekeko price chart
- Elder Impulse Indicator V2
- End Of Year Trading
- Follow the Leader
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Gordon Rose
- half-automated Trading System
- IBD relative strength database Viewer
- ICHIMOKU SIGNAL TRADER
- Intraday Average Volume
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System

- JEEVAN'S SRI CHAKRA
- Market Facilitation Index VS Volume
- mitalpradip
- Momentum Volume Price (MVP) Indicator
- Monthly bar chart
- MS Darvas Box with Exploration
- Nadaraya-Watson Envelope
- New HL Scanner
- NRx Exploration
- nth (1 - 8) Order Polynomial Fit
- Open Range Breakout Trading System
- Ord Volume
- Pattern_-_Rectangle_Base_Breakout_on_High_Vol 2
- Position Sizer vers2, stocks and CFDs
- Price Persistency
- Ranking and sorting stocks
- Relative Strength
- Robert Antony
- RSI Double-Bottom
- RSI Trendlines and Wedges
- RUTVOL timing signal with BB Scoring routine
- Sainath Sidgiddi
- SAR-ForNextBarStop
- SectorRSI
- Simple Candle Exploration
- Stan Weinstein strategy
- STD_STK Multi
- StochD_StochK Single.afl
- Stops Implementation in AFS
- Strength and Weakness
- TAZ Trading Method Exploration
- testing multiple system simulataneously
- Three Day Balance Point
- Trend Detection
- Trend exploration with multiple timeframes
- Trend Exploration: Count Number of New Highs
- Trend Exploration: Slope Moving Average
- Triangle exploration using P&F Chart
- Triangular Moving Average new
- Using From and To dates from Auto Analysis in Code
- Volume - compared with Moving Avg (100%)
- Weekly chart
- Weighted Index
- William's Alligator System II

More information:

See updated/extended version on-line.

AddMultiTextColumn - adds exploration text column based on array

Exploration / Indicators
(AmiBroker 6.20)

SYNTAX **AddMultiTextColumn(ARRAY, "TextList", "Caption", format = 1.2, fgcolor = colorDefault, bkcolor = colorDefault, width = -1)**

RETURNS NOTHING

FUNCTION Adds a text column to the exploration where text displayed is choosen based on array value.

Parameters:

- ARRAY - parameter decides on bar-by-bar basis which item from TextList is choosen
- TextList - newline-separated list of texts to be displayed depending on ARRAY value. Note: newline is a "\n" symbol
  This work so, when ARRAY value is zero, then first item from TextList is choosen and displayed, when ARRAY value is 1 then second item is choosen and so on.
  This allows outputting conditional text in the exploration output, like shown in the example.
- Caption - defines column caption
- format - defines formatting (minimum number of characters used to display a string)
- fgcolor - defines foreground color
- bgcolor - defines background color
- width - defines column width

EXAMPLE

```
Buy = Cross( MACD(), Signal() );
Sell = Cross( Signal(), MACD() );

Filter = 1; // all bars

AddColumn( Buy, "Buy" );
AddColumn( Sell, "Sell" );

TextList = "No signal\nBuy\nSell\nBuy and Sell";
TextSelector = 1 * Buy + 2 * Sell; /* would give 0 if no signal, 1
if a buy, 2 if a sell, 3 if both buy and sell */
AddMultiTextColumn( TextSelector, TextList, "Which signal" );
```

SEE ALSO AddColumn() function , AddTextColumn() function

References:

The **AddMultiTextColumn** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

AddMultiTextColumn - adds exploration text column based on array *614*

AddRankColumn
- add ranking column(s) according to current sort set by SetSortColumns

Exploration / Indicators
(AmiBroker 7.70)

SYNTAX **AddRankColumn()**

RETURNS NOTHING

FUNCTION The function adds ranking column(s) according to current sort set by SetSortColumns to exploration result list.

EXAMPLE

```
Filter = 1;
AddColumn( Close, "Close" );
AddColumn( Volume, "BI" );
AddSummaryRows( 31 + 32, 1.5 );

AddRankColumn(); // without prior sorting AddRankColumn just adds
line number
SetSortColumns( -4 );
AddRankColumn(); // rank according to 4th column (descending)
SetSortColumns( 3 );
AddRankColumn(); // rank according to 3rd column (ascending)
```

SEE ALSO SetSortColumns() function

References:

The **AddRankColumn** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

AddRow
- add raw text row to exploration

Exploration / Indicators
(AmiBroker 60)

SYNTAX **AddRow("text")**

RETURNS NOTHING

FUNCTION The function adds a raw text row to the exploration (allows outputing things without respecting Filter and without being limited to number of bars). This function is preliminary and its parameters are subject to change.

EXAMPLE

```
SetOption("NoDefaultColumns", True );
Filter = 1;
AddColumn( Close, "Column1" );
AddColumn( Null, "Column2" );

for( i = 0; i < 10; i++ )
{
   AddRow( StrFormat( "row %g second column", i ) );
}
```

SEE ALSO AddColumn() function , AddTextColumn() function

References:

The **AddRow** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

AddSummaryRows - add summary row(s) to the exploration output

Exploration / Indicators
(AmiBroker 5.20)

SYNTAX **AddSummaryRows(flags, format = 0, onlycols = 0, ...)**

RETURNS NOTHING

FUNCTION AddSummaryRows automatically adds "summary" row(s) to the exploration output.
Parameters:

The flags parameter can be combination of the following:

- 1 - add TOTAL row
- 2 - add AVERAGE row
- 4 - add MIN row
- 8 - add MAX row
- 16 - add COUNT row
- 32 - add STANDARD DEVIATION row (new in 5.70)

format - defines the numeric formating in WriteVal style so 1.2 for example means 2 decimal digits.
If default value of zero is used (or parameter not specified) the default formatting of "maximum precision" is used - upto 15 digits are printed

onlycols - defines for which columns you want to display summary row values. Note that if you do not specify any columns - ALL will be printed.

If you are using onlycols, you can define upto 10 columns, columns, like in SetSortColumns are numbered starting from 1. For example:

AddSummaryRows(1, 1.2, 3, 5, 7, 9);

Display sum for columns: 3, 5, 7, and 9.

Generally you should call this funciton only once, using combination of flags desired. But it is possible to call AddSummaryRows multiple times and the result will be "accumulation" (i.e. bitwise OR) in case of "flag" parameter. format and onlycols are always overwritten by last call.

EXAMPLE
```
Filter=1;
AddColumn(V, "Volume" );
AddSummaryRows( 31, 1.2 );
// add Total, Average, Min, Max, and Count rows (1+2+4+8+16)=31 -
with two decimal places summary rows are added at the top of the
list
```

SEE ALSO AddColumn() function , AddTextColumn() function

References:

The **AddSummaryRows** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

AddTextColumn - add text exploration column

Exploration / Indicators
(AmiBroker 3.80)

SYNTAX	**AddTextColumn(string, name, format = 1.2, textColor = colorDefault, bkgndColor = colorDefault, width = -1)**
RETURNS	NOTHING
FUNCTION	Adds a new text column to the exploration result list. The column shows *text* and has a caption of *name.* Next two parameters allow to modify text and background color. Width parameter allows to control pixel width of the column
EXAMPLE	addtextcolumn(GroupID(1), "The name of the group");
SEE ALSO	ADDCOLUMN() function

Comments:

Tomasz Janeczko 2005-08-10 06:35:35	Please note that AddTextColumn takes single string as a parameter, so you can only display text that does NOT vary on bar-by-bar basis.

References:

The **AddTextColumn** function is used in the following formulas in AFL on-line library:

- Advanced Search and Find
- AFL Example
- AFL Example - Enhanced
- Alert Output As Quick Rewiev
- AutoTrade using an Exploration
- Average Dollar Price Volatility Exploration
- Backup Data of 1min Interval
- Bottom Fisher Exploration
- Darvas Amibroker
- Dave Landry PullBack Scan
- Elder Impulse Indicator V2
- Elder Triple Screen Trading System
- IBD relative strength database ranker
- ICHIMOKU SIGNAL TRADER
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- Market Facilitation Index VS Volume
- MS Darvas Box with Exploration
- New HL Scanner
- NRx Exploration
- Relative Strength
- Scan New High and New Low
- Stress with SuperSmoother
- Trend exploration with multiple timeframes

- Trend Exploration: Count Number of New Highs
- Trend Exploration: Slope Moving Average
- Triangular Moving Average new
- William's Alligator System II
- WLBuildProcess

More information:

See updated/extended version on-line.

AddToComposite
- add value to composite ticker

Composites
(AmiBroker 40)

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

RETURNS NOTHING

FUNCTION Allows you to create composite indicators with ease. More info...
Parameters:
array - the array of values to be added to "field" in "ticker" composite symbol
"ticker" - the ticker of composite symbol. It is advised to use ~comp (tilde at the beginning) newly added composites are assigned to group 253 by default and
have "use only local database" feature switched on for proper operation with external sources possible field codes: "C" - close , "O" - open, "H" - high, "L" - low, "V" - volume, "I" - open interest, "1" - Aux1 field, "2" - Aux2 field, "X" - updates all OHLC fields at once

flags - contains the sum of following values

- atcFlagDeleteValues = 1 - deletes all previous data from composite symbol at the beginning of scan (recommended)
- atcFlagCompositeGroup = 2 - put composite ticker into group 253 and EXCLUDE all other tickers from group 253 (avoids adding composite to composite)
- atcFlagTimeStamp = 4 - put last scan date/time stamp into FullName field
- atcFlagEnableInBacktest = 8 - allow running AddToComposite in backtest/optimization mode
- atcFlagEnableInExplore = 16 - allow running AddToComposite in exploration mode
- atcFlagResetValues = 32 - reset values at the beginning of scan (not required if you use atcFlagDeleteValues)
- atcFlagDefaults = 7
(this is a composition of atcFlagResetValues | atcFlagCompositeGroup | atcFlagTimeStamp flags)

- atcFlagEnableInPortfolio = 64 - allow running AddToComposite in custom portfolio backtester phase
- atcFlagEnableInIndicator = 128 - allow running AddToComposite in indicator mode
- atcFlagNormalize = 256 - performs normalization by dividing OHLCV fields by the field given as argument (either "I", "1" or "2") after scan is completed

AddToComposite function also detects the context in which it is run
(it works ONLY in scan mode, unless atcFlagEnableInBacktest or atcFlagEnableInExplore flags are specified) and does NOT affect composite ticker when run in Indicator or Commentary mode, so it is now allowed to join scan and indicator into single formula.

EXAMPLE
```
AddToComposite( MACD() > 0, "~BullMACD", "V");
Graph0 = Foreign("~BullMACD", "V");
// Now you can use the same formula in scan AND indicator
```

SEE ALSO

References:

The **AddToComposite** function is used in the following formulas in AFL on-line library:

- 30 Week Hi Indicator - Calculate
- 52 Week New High-New Low Index
- Bad Tick Trim on 5 sec database
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Calculate composites for tickers in list files
- Compare Sectors against Tickers
- Detailed Equity Curve
- Heatmap V1
- Improved NH-NH scan / indicator
- Index of 30 Wk Highs Vs Lows
- Market Direction
- Overbought issues, Oversold issues
- RUTVOL timing signal with BB Scoring routine
- SectorRSI
- Stochastic Divergences, PDI, NDI
- Stochastic OSI & OBI
- The Mean RSIt
- The Mean RSIt (variations)
- The Relative Slope Pivots
- Trending or Trading ?
- Weighted Index
- WLBuildProcess

More information:

See updated/extended version on-line.

ADLine - advance/decline line

Composites
(AmiBroker 3.20)

SYNTAX	**ADLine()**
RETURNS	ARRAY
FUNCTION	Calculates Advance/Decline line indicator
EXAMPLE	adline()
SEE ALSO	

References:

The **ADLine** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

AdvIssues - advancing issues

Composites
(AmiBroker 3.20)

SYNTAX	**AdvIssues()**
RETURNS	ARRAY
FUNCTION	Returns the number of advancing issues for a given market (the one that currently analysed stock belongs to)
EXAMPLE	advissues()
SEE ALSO	

References:

The **AdvIssues** function is used in the following formulas in AFL on-line library:

- Absolute Breadth Index
- Breadth Thrust
- McClellan Oscillator
- McClellan Summation Index

More information:

See updated/extended version on-line.

AdvVolume - advancing issues volume

Composites
(AmiBroker 3.20)

SYNTAX	**AdvVolume()**
RETURNS	ARRAY
FUNCTION	Returns the volume of advancing issues for a given market (the one that currently analysed stock belongs to)
EXAMPLE	advvolume()
SEE ALSO	

References:

The **AdvVolume** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

ADX - average directional movement index

Indicators
(AmiBroker 3.30)

SYNTAX	**adx(period = 14)**
RETURNS	ARRAY
FUNCTION	Calculates Average Directional Index indicator
EXAMPLE	adx(), adx(20)
SEE ALSO	

References:

The **ADX** function is used in the following formulas in AFL on-line library:

- Advanced MA system
- ADX Indicator - Colored
- ADXR
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Bollinger band normalization
- Bull Fear / Bear Fear
- Dave Landry Pullbacks
- DMI Spread Index
- ekeko price chart
- Gordon Rose
- Heatmap V1
- ICHIMOKU SIGNAL TRADER
- Index and ETF trading
- Mndahoo ADX
- Multiple Ribbon Demo
- pattenz
- Perceptron
- swing chart
- TAZ Trading Method Exploration
- TrendingRibbonArrowsADX

More information:

See updated/extended version on-line.

AlertIf - trigger alerts

Trading system toolbox
(AmiBroker 4.10)

SYNTAX **AlertIf(*BOOLEAN_EXPRESSION*, *command*, *text*, *type* = 0, *flags* = 1+2+4+8, *lookback* = 1);**

RETURNS nothing

FUNCTION Triggers alert action if BOOLEAN_EXPRESSION is true.

1. BOOLEAN_EXPRESSION is the expression that if evaluates to True (non zero value) triggers the alert. If it evaluates to False (zero value) no alert is triggered. Please note that only *lookback* most recent bars are considered.

2. The *command* string defines the action taken when alert is triggered. If it is empty the alert *text* is simply displayed in the Alert output window (View->Alert Output). Other supported values of *command* string are:
SOUND *the-path-to-the-WAV-file*
EMAIL
EXEC *the-path-to-the-file-or-URL*

SOUND command plays the WAV file once.
EMAIL command sends the e-mail to the account defined in the settings (Tools->Preferences->E-mail). The format of the e-mail is as follows: Subject: Alert type_name (*type*) Ticker on Date/Time
Body: *text*
EXEC command launches external application or file or URL specified after EXEC command. are attached after file name and *text* is attached at the end

3. Text defines the text that will be printed in the output window or sent via e-mail or added as argument to the application specified by EXEC command

4. *Type* defines type of the alert. Pre-defined types are 0 - default, 1 - buy, 2 - sell, 3 - short, 4- cover. YOu may specify higher values and they will get name "other"

5. *Flags* control behaviour of AlertIF function. This field is a combination (sum) of the following values:
(1 - display text in the output window, 2 - make a beep (via computer speaker), 4 - don't display repeated alerts having the same type, 8 - don't display repeated alerts having the same date/time) By default all these options are turned ON.

6. *lookback* parameter controls how many recent bars are checked

IMPORTANT: AlertIf is not mindless function, it contains internal logic (aka. finite state machine). For full understanding how AlertIf function works and how to use it, you need to read Tutorial: Using formula-based alerts.

EXAMPLE

```
Buy = Cross( MACD(), Signal() );
Sell = Cross( Signal(), MACD() );
Short = Sell;
Cover = Buy;
```

```
AlertIF( Buy, "EMAIL", "A sample alert on "+FullName(), 1 );
AlertIF( Sell, "SOUND C:WindowsMediaDing.wav", "Audio alert", 2 );
AlertIF( Short, "EXEC Calc.exe", "Launching external application", 3
);
AlertIF( Cover, "", "Simple text alert", 4 );
```

Note EXEC command uses ShellExecute function and allows not only EXE files but URLs too.

SEE ALSO

References:

The **AlertIf** function is used in the following formulas in AFL on-line library:

- AFL Example - Enhanced
- Alert Output As Quick Rewiev
- AllinOneAlerts - Module
- Basket Trading System T101
- CCI(20) Divergence Indicator
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- RI - Auto Trading System
- Stock price AlertIf
- Trading ATR 10-1

More information:

See updated/extended version on-line.

AlmostEqual
- rounding error insensitive comparison

Math functions
(AmiBroker 4.80)

SYNTAX **AlmostEqual(x, y, ulps = 5)**

RETURNS NUMBER
ARRAY

FUNCTION This is a helper function for comparing floating point numbers. It returns True if x and y are equal or almost equal upto defined accurracy (ulps). It is recommended to use this function instead of equality check (==) as it leads to more reliable comparisons and less headache caused by IEEE floating pointacurracy issues.

Parameters:

- **x, y** - the numbers or arrays to be compared,
- **ulps** stands for "units in last place" and represents maximum relative error of the comparison. Since 32 bit IEEE floating point numbers have accurracy of 7 significant digits, 1 unit in last place(ulp) represents relative error of 0.00001 %. The default value of ulps parameter is 5 which gives roughtly 0.00005% "comparison sensitivity".

Thanks to Bruce Dawson for his fast routine.

EXAMPLE

```
if( 1/3 == 0.3333333 )
{
  printf("32-bit Floating point IEEE exact equality\n");
}

if( AlmostEqual( 1/3, 0.3333333 ) )
{
  printf("Numbers are almost equal\n");
}
```

SEE ALSO

References:

The **AlmostEqual** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling

More information:

See updated/extended version on-line.

AMA
- adaptive moving average

Moving averages, summation
(AmiBroker 3.50)

SYNTAX	**ama(ARRAY, SMOOTHINGFACTOR)**
RETURNS	ARRAY
FUNCTION	calculates adaptive moving average - simliar to EMA() but smoothing factor could be time-variant (array).

EXAMPLE	The example of volatility-weighted adaptive moving average formula: graph0 = ema(close, 15);
	fast = 2/(2+1);
	slow = 2/(30+1);
	dir=abs(close-ref(close,-10));
	vol=sum(abs(close-ref(close,-1)),10);
	ER=dir/vol;
	sc =(ER*(fast-slow)+slow)^2; graph0 = **ama**(close, sc);

SEE ALSO	

Comments:

Tomasz Janeczko
 	
2006-04-26 20:13:15
	output = AMA(input, factor)
	
	is equivalent to the following looping code:
	
	for(i = 1; i < BarCount; i++)
	{
	output[i] = factor[i] * input[i] + (1 - factor[i]) * output[i - 1];
	}

References:

The **AMA** function is used in the following formulas in AFL on-line library:

- Application of Ehler filter
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- automatic trendlines using fractal patterns
- BBAreacolor&TGLCROSSNEW
- Better Bollinger Bands
- Bman's HaDiffCO
- candlestick chart for Volume/RSI/OBV
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- com-out
- Heatmap V1
- Heikin Ashi Candles
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- Hilbert Study

- IFT of RSI - Multiple TimeFrames
- INTRADAY HEIKIN ASHI new
- Intraday Volume EMA
- Linear Candle
- Pivots for Intraday Forex Charts
- shailu lunia
- Vikram's Floor Pivot Intraday System
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

AMA2 - adaptive moving average

Moving averages, summation
(AmiBroker 3.50)

SYNTAX **ama2(ARRAY, SMOOTHINGFACTOR, FEEDBACKFACTOR)**

RETURNS ARRAY

FUNCTION calculates adaptive moving average - simliar to EMA() but smoothing factor could be time-variant (array).
AMA2 has a separate control of feedbackfactor which is normally (1-SMOOTHINGGFACTOR). Internally this function works like this: today_ama = SMOOTHINGFACTOR * array + FEEDBACKFACTOR * yesterday_ama

EXAMPLE The example of volatility-weighted adaptive moving average formula: graph0 = ema(close, 15);
fast = 2/(2+1);
slow = 2/(30+1);
dir=abs(close-ref(close,-10));
vol=sum(abs(close-ref(close,-1)),10);
ER=dir/vol;
sc =(ER*(fast-slow)+slow)^2; graph0 = **ama2**(close, sc, 1-sc);

SEE ALSO

References:

The **AMA2** function is used in the following formulas in AFL on-line library:

- Candle Stick Analysis
- Coral Trend Indicator
- Cycle Period
- Intraday Volume EMA
- mitalpradip

More information:

See updated/extended version on-line.

ApplyStop
- apply built-in stop

Trading system toolbox
(AmiBroker 3.70)

SYNTAX **ApplyStop(*type, mode, amount, exitatstop, volatile = False, ReEntryDelay = 0, ValidFrom = 0, ValidTo = -1*)**

RETURNS Nothing

FUNCTION controls built-in stops from the formula level (allows optimization of stops)

Parameters:

type =
0 = stopTypeLoss - maximum loss stop,
1 = stopTypeProfit - profit target stop,
2 = stopTypeTrailing - trailing stop,
3 = stopTypeNBar - N-bar stop

mode =
0 - disable stop (stopModeDisable),
1 - amount in percent (stopModePercent), or number of bars for N-bar stop (stopModeBars),
2 - amount in points (stopModePoint);
3 - amount in percent of profit (risk)

amount =
percent/point loss/profit trigger/risk amount.
This could be a number (static stop level) or an array (dynamic stop level)

ExitAtStop

ExitAtStop = 0 - means check stops using only trade price and exit at regular trade price(1)
(if you are trading on close it means that only close price will be checked for exits and exit will be done at close price)
ExitAtStop = 1 - check High-Low prices and exit intraday on price equal to stop level on the same bar when stop was triggered
ExitAtStop = 2 - check High-Low prices but exit NEXT BAR on regular trade price.

volatile -
decides if amount (or distance) (3rd parameter) is sampled at the trade entry and remains fixed during the trade (Volatile = FALSE - old behaviour) or if can vary during the trade (Volatile = TRUE) (allows single line Chandelier exit implementation)(2)

ReEntryDelay -
how many bars to wait till entering the same stock is allowed.

ValidFrom -
defines first bar since entry when stop can generate an exit. 0 means from the very beginning

ValidTo -
defines last bar since entry when stop can generate an exit. -1 means "infinite". By default stops are valid all the time (0/-1).

ValidFrom/ValidTo can be used to create stops that get actived/deactivated in different times. This setting is independent for each stop type. It also works in conjunction with SetOption("HoldMinBars", x). HoldMinBars affects BOTH regular exits and stops, preventing ALL kind of exits during defined period. ValidFrom/ValidTo works on each stop separately and does not affect regular exits.

Note on using stops:
Scenario 1:
you trade on next bar OPEN and want to exit intraday on stop price

Correct settings:
ActivateStopsImmediately turned ON
ExitAtStop = 1
Trade delays set to one
Trade price set to open

Scenario 2:
you trade on today's close and want to exit intraday on stop price
Correct settings:
ActivateStopsImmediately turned OFF
ExitAtStop = 1
Trade delays set to zero
Trade price set to close

Scenario 3:
you trade on next day OPEN and want to exit by stop on OPEN price when PREVIOUS day H-L range hits stop
Correct settings:

ExitAtStop = 2 (NEW)
Trade delays set to one
Trade price set to open

- a) (if you want to have stops executed AFTER regular signals, so cash from stopped out positions is NOT available to enter trades the same day)
  ActivateStopsImmediately turned ON

- b) (if you want to have stops executed BEFORE regular signals, so cash from stopped out positions IS available to enter new trades the same day)
  ActivateStopsImmediately turned OFF

Scenario 4:
you trade on today's close and want to exit only when today's close price hits the stop level

Correct settings:

ActivateStopsImmediately turned OFF
ExitAtStop = 0
Trade delays set to zero
Trade price set to close

LIMITATIONS:

- [1] ExitAtStop = 0 uses SellPrice/CoverPrice variables in backtestRegular mode only, in other modes it uses trade prices from the Settings dialog (not overridable via SellPrice/CoverPrice)
- [2] Volatile stops (Volatile=True) work only in backtestRegular mode

EXAMPLE

```
/* max loss stop optimization */

ApplyStop(stopTypeLoss,
         stopModePercent,
         Optimize( "max. loss stop level", 10, 2, 30, 1 ),
         True );

/* single-line implementation of Chandelier exit */

ApplyStop(stopTypeTrailing, stopModePoint, 3*ATR(14), True, True );

/* N-bar stop */
ApplyStop( stopTypeNBar, stopModeBars, 5 );
```

SEE ALSO

Comments:

Herman van den Bergen psytek [at] magma.ca 2003-02-23 09:53:51	If you are trading at the Close with zero delays be sure to unmark "Activate Stops Immediately" in Settings.

Corey Saxe csaxe [at] nwi.net 2003-03-01 23:33:13	For visual conformation of ApplyStop function, add the following lines below your ApplyStop formula in Indicator Builder:
	
	Equity(1); // THIS EVALUATES STOPS
	
	Plot(Sell==4,"ApplyStop Sell",colorRed,1|styleOwnScale);
	Plot(Cover==4,"ApplyStop Cover",colorGreen,1|styleOwnScale);

Tomasz Janeczko tj --at-- amibroker.com 2004-08-28 03:14:12	If two or more different stops are triggered on the VERY SAME bar then they are evaluated in this fixed order: Fixed Ruin stop (loosing 99.96% of the starting capital) Max. loss stop Profit target stop Trailing stop

	N-bar stop
Graham Kavanagh gkavanagh [at] e-wire.net.au 2004-09-30 21:55:42	from equity comments Depending on kind of the stop various values are written back to sell/cover array to enable you to distinguish if given signal was generated by regular rule or by stop. 1 - regular exit 2 - max. loss 3 - profit target 4 - trailing 5 - n-bar stop 6 - ruin stop
Tomasz Janeczko tj --at-- amibroker.com 2005-03-01 17:10:39	ExitAtStop has a new meaning for N-BAR stop type. If ExitAtStop = 0 then N-bar stop has the lowest priority (so if for example profit target stop is hit on the same bar then profit target is evaluated first) If ExitAtStop = 1 then N-bar stop has highest priority and it is evaluated before all other stops. The same effect is obtained by checking "Has priority" box in AA Settings window.
Tomasz Janeczko tj-/nospam/ [at] amibroker.com 2006-01-13 11:41:32	ApplyStop function is designed to be used to simulate stop orders placed at the exchange or simulated by the brokerage Please read this how such stops operate: http://www.interactivebrokers.com/en/trading/orders/stop.php?ib_entity=uk http://www.interactivebrokers.com/en/trading/orders/trailingStops.php?ib_entity=uk

References:

The **ApplyStop** function is used in the following formulas in AFL on-line library:

- ATR Study
- AutoTrade using an Exploration
- Caleb Lawrence
- danningham penetration
- Ed Seykota's TSP: EMA Crossover System
- Follow the Leader
- Index and ETF trading
- RUTVOL timing signal with BB Scoring routine
- SectorRSI
- The D_oscillator
- The Three Day Reversal
- Trend Continuation Factor
- Vivek Jain

More information:

See updated/extended version on-line.

Asc
- get ASCII code of character

String manipulation
(AmiBroker 4.80)

SYNTAX **Asc(string, pos = 0)**

RETURNS NUMBER

FUNCTION Returns the ANSI character code corresponding to the first letter in a string (if position is not specified) or code of character at specified position. If you don't specify position (pos argument) then first character is used. Negative values of pos reference characters counting from the end of string.

Useful for creation of exploration that displays single letters for signals instead of numbers.

EXAMPLE

```
Buy = Cross(MACD(),Signal());
Sell = Cross(Signal(),MACD());

Filter = Buy OR Sell;

AddColumn( IIf( Buy, Asc("B"), Asc("S")), "Signal", formatChar );
```

SEE ALSO AddColumn() function

References:

The **Asc** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_1
- AutoTrade using an Exploration
- Ed Seykota's TSP: Support and Resistance
- New HL Scanner

More information:

See updated/extended version on-line.

asin - arcsine function

Math functions

SYNTAX	**asin(x)**
RETURNS	NUMBER, ARRAY
FUNCTION	Returns the arcsine of NUMBER or ARRAY. The asin function returns the arcsine of x in the range -pi/2 to pi/2 radians. If x is less than -1 or greater than 1, asin returns an indefinite
EXAMPLE	
SEE ALSO	SIN() function

References:

The **asin** function is used in the following formulas in AFL on-line library:

- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator

More information:

See updated/extended version on-line.

atan
- arc tan

Math functions

SYNTAX	**atan(NUMBER),** **atan(ARRAY)**
RETURNS	NUMBER ARRAY
FUNCTION	Returns the arc tangent of NUMBER or ARRAY. The value is returned in radians
EXAMPLE	The formula "atan(1.00)" returns PI/4
SEE ALSO	

References:

The **atan** function is used in the following formulas in AFL on-line library:

- Andrews Pitchfork
- Andrews PitchforkV3.3
- AR_Prediction.afl
- Cybernertic Hilbert Sine Wave
- DMI Spread Index
- Dominant Cycle Phase
- Ehlers Hilbert Transformer Indicator
- Even Better Sinewave Indicator
- Gabriel Linear Regression Angle Indicator
- Heatmap V1
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- Hilbert Study
- John Ehler
- Moving Average "Crash" Test
- Multiple sinus noised
- Schiff Lines
- Signal to Noise
- Sine Wave Indicator
- Smoothed Adaptive Momentum
- Three Pole Butterworth
- Trigonometric Fit - TrigFit with AR for cos / sin
- Voss Predictive Filter (A Peek Into the Future)
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots

More information:

See updated/extended version on-line.

atan2
- calculates arctangent of y/x

Math functions
(AmiBroker 4.90)

SYNTAX **atan2(y, x)**

RETURNS NUMBER or ARRAY

FUNCTION atan2 returns the arctangent of y/x. If x is 0, atan2 returns 0. If both parameters of atan2 are 0, the function returns 0. atan2 returns a value in the range -PI to +PI radians, using the signs of both parameters to determine the quadrant of the return value.

EXAMPLE

```
ffc = FFT(data,Len);
for( i = 0; i < Len - 1; i = i + 2 )
{
   amp[ i ] = amp[ i + 1 ] = sqrt(ffc[ i ]^ 2  +  ffc[ i + 1 ]^2);
   phase[ i ] = phase[ i + 1 ] = atan2( ffc[ i + 1], ffc[ i ] );
}
```

SEE ALSO atan() function

References:

The **atan2** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

ATR
- average true range

Indicators
(AmiBroker 3.30)

SYNTAX	**atr(period)**
RETURNS	ARRAY
FUNCTION	Calculates Average True Range indicator
EXAMPLE	atr(7)
SEE ALSO	

Comments:

Bob Jagow bjagow [at] charterr.net 2003-02-06 23:37:50	For other MAs, use ATR(1) to get the True Range. E.g., MA(ATR(1),period), WMA(ATR(1),period), etc.
Tomasz Janeczko tj --at-- amibroker.com 2005-02-03 06:46:39	Note that original formulation of ATR (the one that AmiBroker implements) uses WILDERS smoothing (not simple moving average) For more details check this page: http://stockcharts.com/education/IndicatorAnalysis/indic_ATR.html

References:

The **ATR** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Abhimanyu
- ADXVMA
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- AR_Prediction.afl
- ATR Study
- AutoTrade using an Exploration
- BB squeeze
- BBAreacolor&TGLCROSSNEW
- Bollinger - Keltner Bands
- Bollinger Band Gap
- Bow tie
- Caleb Lawrence
- Chandelier Exit
- Chandelier Exit
- Chandelier Exit or Advanced Trailing Stop
- Chandelier Plugin
- Channel/S&R and trendlines
- Commodity Selection Index (CSI)
- Double Super Trend System
- elliott wave manual labelling
- Fre

- Gann Swing Charts in 3 modes with text
- Gartley 222 Pattern Indicator
- Gordon Rose
- Halftrend
- Harmonic Pattern Detection
- Hull Range Indicator
- Keltner Channel
- Linear Candle
- MACD BB Indicator
- MACD-V
- mitalpradip
- pattenz
- Perceptron
- Peterson
- PF Chart - Close - April 2004
- Pivots And Prices
- Point & figure Chart India Securities
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- Random Walk Index, base formula included
- Raw ADX
- Renko Chart
- Revised Renko chart
- SectorRSI
- SF Entry,Stop, PT Indicator
- STARC Bands
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- STO & MACD Buy Signals with Money-Management
- Stop-loss Indicator bands
- Super Trend Indicator
- Super Trend Indicator
- Support and resistance
- Tracking Error
- Tracking Error
- Trading ATR 10-1
- Trigonometric Fit - TrigFit with AR for cos / sin
- TSV
- TTM Squeeze
- visual turtle trading system
- Vivek Jain
- Volatility Quality Index
- Volatility System
- VSTOP (2)
- VSTOP (3)
- Weinberg's The Range Indicator
- White Theme
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

BarIndex
- get zero-based bar number

Date/Time
(AmiBroker 4.30)

SYNTAX	**BarIndex()**
RETURNS	ARRAY
FUNCTION	returns zero-based bar number - the same as Cum(1)-1 but it is much faster than Cum(1) when used in Indicators New in 5.30: BarIndex() now returns values always starting from zero (even if QuickAFL is turned on). This change is required because Cum() now does not require all bars and formulas mixing Cum(1) and BarIndex would work improperly otherwise.
EXAMPLE	`ThisIsLastBar = BarIndex() == LastValue( BarIndex() );`
SEE ALSO	CUM() function

Comments:

Tomasz Janeczko
tj --at--
amibroker.com
2004-07-23
07:07:29
	When QuickAFL is ON, the BarIndex() may not be equal with array item index.
	
	Actual array item
	corresponding to bar index can be found this way:
	
	bi = BarIndex();
	arrayitem = SelectedValue(bi) - bi[0];
	"Close at selected bar:" + Close[arrayitem];

Rolly rollyzhang [at] hotmail.com 2007-11-03 14:12:22	BarIndex() returns an array and LastValue() returns a number. The API's example compares the two to give the boolean ThisIsLastBar. How could an array be compared with a number? How could it work? Please explain.
Tomasz Janeczko tj at amibroker dot com 2007-11-04 09:14:21	That's simple - it compares each array element to a number and produces array as a result. Detailed explanation in User's Guide: Tutorial: Understanding AFL

References:

The **BarIndex** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Adaptive Price Channel
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Andrews Pitchfork
- Andrews PitchforkV3.3
- Another FIb Level
- AR_Prediction.afl
- Auto-Optimization Framework
- Automatic Trendlines using multiple timeframes

- babaloo chapora
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bullish Percent Index 2004
- Caleb Lawrence
- Candle Identification
- Candle Stick Analysis
- Candle Stick Demo
- CCI(20) Divergence Indicator
- Chandelier Exit
- channel indicator
- Channel/S&R and trendlines
- Chart Zoom
- com-out
- Congestions detection
- crMathLib
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period
- Date_To_Num(), Time_To_Num()
- Elder safe Zone Long + short
- elliott wave manual labelling
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- FirstBarIndex(), LastBarIndex()
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann Swing Charts in 3 modes with text
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Harmonic Pattern Detection
- Heikin-Ashi(Koma-Ashi) with Moving Average
- High Low Detection code
- Hurst "Like" DE
- Intraday Fibonacii Trend Break System
- Intraday Range and Periods Framer
- Intraday Trend Break System
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Last Five Trades Result Dashboard – AFL code
- Least Squares Channel Indicator
- Linear Regression Line w/ Std Deviation Channels
- MACD indicator display
- Market Profile
- MFE and MAE and plot trades as indicator
- Multi-color Volume At Price (VAP)
- Multiple Ribbon Demo
- Multiple sinus noised
- Nadaraya-Watson Envelope

- Now Send Push Notifications From Amibroker
- nth (1 - 8) Order Polynomial Fit
- pattenz
- PF Chart - Close - April 2004
- Pivot Finder
- Point & figure Chart India Securities
- Polyfit Lines
- Prior Daily OHLC
- Probability Density & Gaussian Distribution
- Rea Time Daily Price Levels
- Relative Strength Multichart of up to 10 tickers
- Renko Chart
- Schiff Lines
- shailu lunia
- Square of Nine Roadmap Charts
- Support and resistance
- suresh
- TAPE READING
- tomy_frenchy
- Trend Detection
- Trend exploration with multiple timeframes
- Trend Exploration: Slope Moving Average
- Trend Lines from 2 points
- Trigonometric Fit - TrigFit with AR for cos / sin
- Updated Renko Chart
- Using From and To dates from Auto Analysis in Code
- Visible Min and Max Value Demo
- Volume based support resistance price finder
- VWAP - Volume Weighted Average Price
- White Theme
- WLBuildProcess
- Woodie's CCI Panel Full Stats
- Zig Zag Indicator with Valid Entry and Exit Points
- ZigZag Retracements

More information:

See updated/extended version on-line.

BarsSince - bars since

Trading system toolbox

SYNTAX	**BarsSince(ARRAY)**
RETURNS	ARRAY
FUNCTION	Calculates the number of bars (time periods) that have passed since ARRAY was true (or 1)
EXAMPLE	barssince(macd() < 0)
SEE ALSO	BarsSinceCompare() function

References:

The **BarsSince** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Andrews PitchforkV3.3
- AR_Prediction.afl
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- Awsome Oscilator
- Buyer Seller Force
- Caleb Lawrence
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- CCI(20) Divergence Indicator
- changing period moving avarage
- Channel/S&R and trendlines
- Cole
- Commodity Channel Index
- Connors TPS
- ConnorsRSI
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Day Bar No
- Divergences
- ekeko price chart
- Elder Impulse Indicator V2
- Envelope System
- FastStochK FullStochK-D
- Fib Fan Based on ZZ
- Follow the Leader
- Fre
- Fund Screener

- Gann Swing Chart
- Gann Swing Charts in 3 modes with text
- Halftrend
- Harmonic Pattern Detection
- Heatmap V1
- Intraday Average Volume
- IntraDay Open Marker
- Intraday Volume EMA
- Lagging MA-Xover
- MACD commentary
- MACD indicator display
- Market Profile
- mitalpradip
- Moving Averages NoX
- MO_CrashZone
- New HL Scanner
- OBV with Linear Regression
- Open Range Breakout Trading System
- pattenz
- Performance Check
- Peterson
- Positive Bars Percentage
- prakash
- Price Persistency
- Range Filter - Trading Strategy
- Rea Time Daily Price Levels
- Relative Strength Index
- Reverse MFI Crossover
- Schiff Lines
- Stochastic Divergence, negative
- Stochastic Divergence, positive
- Stochastic Divergences, PDI, NDI
- Stochastic Fast%K and Full
- Stops Implementation in AFS
- TD sequential
- TD Sequential
- The Fibonaccian behavior
- The Three Day Reversal
- Trade day of month
- Trend Analysis_Comentary
- Triangular Moving Average
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- Trix Bars number
- VWAP - Volume Weighted Average Price
- VWAP versus Average Price
- Williams Alligator system
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Zig Zag Indicator with Valid Entry and Exit Points
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount

More information:

See updated/extended version on-line.

BarsSinceCompare - bars since comparision between past and present array values was true

Trading system toolbox
(AmiBroker 6.40)

SYNTAX **BarsSinceCompare(past, comparison, current)**

RETURNS ARRAY

FUNCTION For every bar in *current* array, go back in time and compare *past* array values with *current* bar value and get number of bars since *comparison* was true.

Parameters:

- *past* - the array holding "past" values to compare to
- *comparison* - string that defines comparison operator, available choices are ">", "=", "<="
- *current* - the array holding "current" values to compare to

For more discussion about that function see:
https://forum.amibroker.com/t/duplicate-bars-since-price-was-this-high-low/28355/14

EXAMPLE

```
// get bars since close was last time higher than current close
BarsSinceCompare( Close, ">", Close );

// get bars since low was last time higher than current close
BarsSinceCompare( Low, ">", Close );

// get bars since low was last time higher than or equal than
current close
BarsSinceCompare( Low, ">=", Close );

// get bars since high was lower than or equal than current low
BarsSinceCompare( High, "<=", Low );
```

SEE ALSO BarsSince() function

References:

The **BarsSinceCompare** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

BBandBot - bottom bollinger band

Indicators

SYNTAX	**BBandBot(ARRAY, *periods* = 15, *width* = 2)**
RETURNS	ARRAY
FUNCTION	Calculates the bottom Bollinger Band of ARRAY shifted down w*idth* standard deviations (using *periods* averaging range).
EXAMPLE	bbandbot(close, 10, 2)
SEE ALSO	

References:

The **BBandBot** function is used in the following formulas in AFL on-line library:

- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Bman's HaDiffCO
- bolingerbands
- Bollinger - Keltner Bands
- Bollinger band normalization
- Bollinger Band Squeeze
- bonlinger bands
- Congestions detection
- ekeko price chart
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- MACD BB Indicator
- prakash
- RUTVOL timing signal with BB Scoring routine
- TSV
- TTM Squeeze
- Volatility Breakout with Bollinger Bands

More information:

See updated/extended version on-line.

BBandTop - top bollinger band

Indicators

SYNTAX **BBandTop(ARRAY, *periods* = 15, *width* = 2)**

RETURNS ARRAY

FUNCTION Calculates the top Bollinger Band of ARRAY shifted up *width* standard deviations (using *periods* averaging range).

EXAMPLE bbandtop(close, 10, 2)

SEE ALSO

References:

The **BBandTop** function is used in the following formulas in AFL on-line library:

- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Bman's HaDiffCO
- bolingerbands
- Bollinger - Keltner Bands
- Bollinger Band Gap
- Bollinger band normalization
- Bollinger Band Squeeze
- bonlinger bands
- Congestions detection
- ekeko price chart
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- MACD BB Indicator
- prakash
- RUTVOL timing signal with BB Scoring routine
- TSV
- TTM Squeeze
- Volatility Breakout with Bollinger Bands

More information:

See updated/extended version on-line.

BeginValue
- Value of the array at the begin of the range

Date/Time
(AmiBroker 4.30)

SYNTAX **BeginValue(ARRAY)**

RETURNS NUMBER

FUNCTION This function gives the single value (number) of the ARRAY at the beginning of the selected range. If no range is marked then the value at the first bar is returned.

To select the range you have to double click in the chart at the beginning of the range and then double click in the chart at the end of the range. Then > < markers will appear above date axis.

EXAMPLE 1. Simple commentary:

```
WriteVal( BeginValue( DateTime() ), formatDateTime );
WriteVal( EndValue( DateTime() ), formatDateTime );
"Precentage change of close is " +
WriteVal( 100 * (EndValue( Close ) - BeginValue( Close
))/BeginValue( Close ) ) + "%";
```

2. Get the number of bars in the range and calculate some stats for that range:

```
Period = EndValue( BarIndex() ) - BeginValue( BarIndex() );
StandardDeviationInTheRange = EndValue( StDev( Close, Period ) );
```

SEE ALSO ENDVALUE() function

References:

The **BeginValue** function is used in the following formulas in AFL on-line library:

- Adaptive Price Channel
- Another FIb Level
- Congestions detection
- Elder safe Zone Long + short
- Futures - Dollar Move Indicator
- Multi-color Volume At Price (VAP)
- nth (1 - 8) Order Polynomial Fit
- Relative Strength Multichart of up to 10 tickers
- Trend Lines from 2 points

More information:

See updated/extended version on-line.

CategoryAddSymbol - adds a symbol to a category

Information / Categories
(AmiBroker 4.50)

SYNTAX **CategoryAddSymbol(*symbol*, *category*, *number*)**

RETURNS NOTHING

FUNCTION The **CategoryAddSymbol** function adds the *symbol* to given category, note that for markets, groups, industries 'adding' means moving from one category to another, since the symbol is assigned always to one and only one market, group, industry and sector. This limitation does not apply to watchlists, favorites, and index categories. When *symbol* string is empty ("") then current symbol is used.

category is one of the following:

- categoryMarket
- categoryGroup
- categorySector
- categoryIndustry
- categoryWatchlist
- categoryFavorite
- categoryIndex
- categoryGICS
- categoryICB

number is a market/group/industry/sector/watchlist number:
0..255 for categoryMarket, categoryGroup, categoryIndustry
0..63 for categorySector
no limit for categoryWatchlist.
ignored for categoryFavorite, categoryIndex

The meaning of index parameter is different for GICS and ICB categories - the index for categoryGICS and categoryICB is actually GICS/ICB code. Such as 10 for energy sector or 351010 for "Health Care Equipment & supplies" industry. The codes are fixed even if new classifications are added at some point in the future. This means that you won't need to change AFL codes even if new classifications are added. But it is important to understand that these codes work in hierarchical way. So

GetCategorySymbols(categoryGICS, 10)

will return all symbols belonging to energy sector, including those in 10101010 - Oil & Gas Drilling sector as well as 10102050 - Coal & Consumable Fuels; for example. See http://en.wikipedia.org/wiki/Global_Industry_Classification_Standard for more details on GICS system
http://en.wikipedia.org/wiki/Industry_Classification_Benchmark for more details on ICB system

EXAMPLE

```
// the code adds symbols with last day volume > 100000
// to the watch list number 1
if( LastValue( V ) > 100000 )
{
   CategoryAddSymbol( "", categoryWatchlist, 1 );
}
```

SEE ALSO CategoryGetName() function , CategoryGetSymbols() function

References:

The **CategoryAddSymbol** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically
- Calculate composites for tickers in list files
- Heatmap V1
- WLBuildProcess

More information:

See updated/extended version on-line.

CategoryCreate - add new category (such as watch list)

Information / Categories
(AmiBroker 5.70)

SYNTAX **CategoryCreate("name", category)**

RETURNS STRING

FUNCTION CategoryCreate("name", category) - adds new category (currently supports only adding watchlists)

```
newwl = CategoryCreate( "name", categoryWatchlist );
```

This will add (create new) watch list with "name" (if it does not exist) or will return existing "name" watch list

newwl will hold new watch list index.

Currently only creation of watch list is supported by this function.

EXAMPLE

```
newwl = CategoryCreate( "name", categoryWatchlist );
```

SEE ALSO CategoryAddSymbol() function , CategoryFind() function , CategoryGetName() function , CategoryGetSymbols() function , CategoryRemoveSymbol() function , CategorySetName() function

References:

The **CategoryCreate** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically

More information:

See updated/extended version on-line.

CategoryFind
- search for category by name

Information / Categories
(AmiBroker 50)

SYNTAX **CategoryFind("name", category)**

RETURNS NUMBER

FUNCTION It allows to search for category by name. It takes category name and kind as parameters and returns INDEX (ordinal number).

For example it allows to find watch list index by name.

Supported categories:

- categoryMarket
- categoryGroup
- categorySector
- categoryIndustry
- categoryWatchlist
- categoryFavorite
- categoryIndex
- categoryGICS
- categoryICB

The index (in the example below watch list number) can be later used in functions that need the index (like CategoryGetSymbols).

EXAMPLE

```
wlnumber = CategoryFind("MyWatch List 1", categoryWatchlist );
mysymbols = CategoryGetSymbols(categoryWatchlist, wlnumber );
```

SEE ALSO CategoryAddSymbol() function , CategoryGetName() function , CategoryGetSymbols() function , CategoryRemoveSymbol() function

References:

The **CategoryFind** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically

More information:

See updated/extended version on-line.

CategoryGetName - get the name of a category

Information / Categories
(AmiBroker 4.50)

SYNTAX **CategoryGetName(*category*, *number*)**

RETURNS STRING

FUNCTION The **CategoryGetName** function retrieves the name of category.

category is one of the following:

- categoryMarket
- categoryGroup
- categorySector
- categoryIndustry
- categoryWatchlist
- categoryFavorite
- categoryIndex
- categoryGICS
- categoryICB

number is a market/group/industry/sector/watchlist number:
0..255 for categoryMarket, categoryGroup, categoryIndustry
0..63 for categorySector
no limit for categoryWatchlist.
ignored for categoryFavorite, categoryIndex

The meaning of index parameter is different for GICS and ICB categories - the index for categoryGICS and categoryICB is actually GICS/ICB code. Such as 10 for energy sector or 351010 for "Health Care Equipment & supplies" industry. The codes are fixed even if new classifications are added at some point in the future. This means that you won't need to change AFL codes even if new classifications are added. But it is important to understand that these codes work in hierarchical way. So

GetCategorySymbols(categoryGICS, 10)

will return all symbols belonging to energy sector, including those in 10101010 - Oil & Gas Drilling sector as well as 10102050 - Coal & Consumable Fuels; for example. See http://en.wikipedia.org/wiki/Global_Industry_Classification_Standard for more details on GICS system
http://en.wikipedia.org/wiki/Industry_Classification_Benchmark for more details on ICB system

EXAMPLE

```
CategoryGetName( categoryWatchlist, 1 );
```

SEE ALSO CategoryGetSymbols() function

References:

The **CategoryGetName** function is used in the following formulas in AFL on-line library:

- In Watch List

More information:

See updated/extended version on-line.

CategoryGetSymbols - retrieves comma-separated list of symbols belonging to given category

Information / Categories
(AmiBroker 4.50)

SYNTAX **CategoryGetSymbols(category, index, mode = 0)**

RETURNS STRING

FUNCTION Retrieves comma-separated list of symbols belonging to given category Supported categories:

- categoryMarket
- categoryGroup
- categorySector
- categoryIndustry
- categoryWatchlist
- categoryFavorite
- categoryIndex
- categoryGICS
- categoryICB
- categoryAll (new in 5.50) - means all symbols in the database (

index is a market/group/industry/sector/watchlist number:
0..255 for categoryMarket, categoryGroup, categoryIndustry
0..63 for categorySector
no limit for categoryWatchlist.
ignored for categoryFavorite, categoryIndex

mode - (new in 5.50) mode parameter decides what field is retrieved:

- 0 (default value) - ticker symbol
- 1 - full name

The meaning of index parameter is different for GICS and ICB categories - the index for categoryGICS and categoryICB is actually GICS/ICB code. Such as 10 for energy sector or 351010 for "Health Care Equipment & supplies" industry. The codes are fixed even if new classifications are added at some point in the future. This means that you won't need to change AFL codes even if new classifications are added. But it is important to understand that these codes work in hierarchical way. So

GetCategorySymbols(categoryGICS, 10)

will return all symbols belonging to energy sector, including those in 10101010 - Oil & Gas Drilling sector as well as 10102050 - Coal & Consumable Fuels; for example. See http://en.wikipedia.org/wiki/Global_Industry_Classification_Standard for more details on GICS system
http://en.wikipedia.org/wiki/Industry_Classification_Benchmark for more details on ICB system

Use **StrExtract** function to extract individual symbols from the list.

EXAMPLE

```
/* note: if given watch list contains lots of symbols
** performance may be poor
** AVOID SUCH CODES IN REAL-TIME MODE !
*/
function CreateAverageForWatchList( listnum )
{
   // retrive comma-separated list of symbols in watch list
   list = CategoryGetSymbols( categoryWatchlist, listnum );

   Average = 0; // just in case there are no watch list members

   for( i = 0; ( sym = StrExtract( list, i ) ) != ""; i++ )
   {
      f = Foreign( sym, "C" );
      if( i == 0 ) Average = f;
      else Average = Average + f;
   }
   return Average / i; // divide by number of components
}

Plot( CreateAverageForWatchList( 1 ), "Avg of WL 1", colorGreen );
```

Example 2, retrieving all symbols tickers and full names `// to get all ticker symbols`

```
CategoryGetSymbols( categoryAll, 0 );
//to get full names of all symbols use:
CategoryGetSymbols( categoryAll, 0, 1 );
```

SEE ALSO GETCATEGORYSYMBOLS() function , StrExtract() function , INWATCHLIST() function , CategoryGetName() function

References:

The **CategoryGetSymbols** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_1
- Basket Trading System T101
- Calculate composites for tickers in list files
- Graphical sector stock amalysis
- Heatmap V1
- IB Backfiller
- IBD relative strength database ranker
- Optimal Weights
- Peter Cooper
- Relative Strength

More information:

See updated/extended version on-line.

CategoryRemoveSymbol - remove a symbol from a category

Information / Categories
(AmiBroker 4.50)

SYNTAX **CategoryRemoveSymbol(*symbol*, *category*, *number*)**

RETURNS NOTHING

FUNCTION Removes the *symbol* from given *category*.

Note that for markets, groups, industries 'removing' means moving from given category to category with number zero, since the symbol is assigned always to one and only one market, group, industry and sector. This limitation does not apply to watchlists, favorites, and index categories. When symbol string is empty ("") then current symbol is used.

category is one of the following:

- categoryMarket
- categoryGroup
- categorySector
- categoryIndustry
- categoryWatchlist
- categoryFavorite
- categoryIndex
- categoryGICS
- categoryICB

number is a market/group/industry/sector/watchlist number:
0..255 for categoryMarket, categoryGroup, categoryIndustry
0..63 for categorySector
no limit for categoryWatchlist.
ignored for categoryFavorite, categoryIndex

The meaning of index parameter is different for GICS and ICB categories - the index for categoryGICS and categoryICB is actually GICS/ICB code. Such as 10 for energy sector or 351010 for "Health Care Equipment & supplies" industry. The codes are fixed even if new classifications are added at some point in the future. This means that you won't need to change AFL codes even if new classifications are added. But it is important to understand that these codes work in hierarchical way. So

GetCategorySymbols(categoryGICS, 10)

will return all symbols belonging to energy sector, including those in 10101010 - Oil & Gas Drilling sector as well as 10102050 - Coal & Consumable Fuels; for example. See http://en.wikipedia.org/wiki/Global_Industry_Classification_Standard for more details on GICS system
http://en.wikipedia.org/wiki/Industry_Classification_Benchmark for more details on ICB system

EXAMPLE

```
// the code removes the symbols with last day volume < 1000
// from the watch list number 1
if( LastValue( V ) < 1000 )
```

```
{
   CategoryRemoveSymbol( "", categoryWatchlist, 1 );
}
```

SEE ALSO CategoryAddSymbol() function , CategoryGetName() function , CategoryGetSymbols() function

References:

The **CategoryRemoveSymbol** function is used in the following formulas in AFL on-line library:

- Calculate composites for tickers in list files
- Heatmap V1
- WLBuildProcess

More information:

See updated/extended version on-line.

CategorySetName
- set the name of category (group, market, watch list, industry)

Information / Categories
(AmiBroker 5.20)

SYNTAX **CategorySetName(name, category, number)**

RETURNS NOTHING

FUNCTION Function sets the name of category (group,market, watch list, industry)

Arguments;

- name - a new name for the category (in case of watch lists it has to be unique)
- category - type of category, one of the following: categoryMarket, categoryGroup, categorySector, categoryIndustry, categoryWatchlist, categoryGICS, categoryICB
- number - the number (index) of the category 0.255 for market, group industry, 0..32 for sectors, 0...unlimited for watch lists

Please note that the function will NOT create watch list of given index if one does not exist.

The meaning of index parameter is different for GICS and ICB categories - the index for categoryGICS and categoryICB is actually GICS/ICB code. Such as 10 for energy sector or 351010 for "Health Care Equipment & supplies" industry. The codes are fixed even if new classifications are added at some point in the future. This means that you won't need to change AFL codes even if new classifications are added. But it is important to understand that these codes work in hierarchical way. So

GetCategorySymbols(categoryGICS, 10)

will return all symbols belonging to energy sector, including those in 10101010 - Oil & Gas Drilling sector as well as 10102050 - Coal & Consumable Fuels; for example. See http://en.wikipedia.org/wiki/Global_Industry_Classification_Standard for more details on GICS system
http://en.wikipedia.org/wiki/Industry_Classification_Benchmark for more details on ICB system

EXAMPLE CategorySetName("Testing", categoryWatchList, 1);

SEE ALSO CategoryAddSymbol() function , CategoryFind() function , CategoryGetName() function , CategoryGetSymbols() function , CategoryRemoveSymbol() function

References:

The **CategorySetName** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

CCI - commodity channel index

Indicators

SYNTAX	**CCI(** ***periods = 14*** **)**
	CCIa(***array, periods = 14*** **)**

RETURNS	ARRAY
FUNCTION	Calculates the Commodity Channel Index (using *periods* averaging range). Second version (CCIa) accepts input array, so CCI can be applied to array different than close. (CCIa exists in AFL 2.2+ only (v.4.20+))

EXAMPLE	CCI(14)
	CCIa(High, 14);

SEE ALSO	

Comments:

Tomasz Janeczko
tj --at-- amibroker.com
2003-09-03 04:24:35
	CCI uses internally 'Avg' built-in price array.
	'Avg' is also known as typical price:
	Avg = (H + L + C) /3
	
	So
	CCI(period) is equivalent to CCia(Avg, period).
	
	Therefore if you want to calculate CCI from Foreign ticker you should overwrite Avg array,
	instead of OHLC:
	
	Avg = (Foreign("!VIX", "H") + Foreign("!VIX", "L") + Foreign("!VIX", "C")) / 3;
	cc = CCI(period);
	
	Alternativelly use CCIa that takes array directly:
	
	cc = CCIa(Foreign("!VIX", "H") + Foreign("!VIX", "L") + Foreign("!VIX", "C")) / 3, period);

References:

The **CCI** function is used in the following formulas in AFL on-line library:

- Adaptave Zones O/B & O/S Oscillator
- Auto-Optimization Framework
- BBAreacolor&TGLCROSSNEW
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- CCI(20) Divergence Indicator
- CCI/DI+- COMBO indicator
- Commodity Channel Index
- JEEVAN'S SRI CHAKRA

- Price with Woodies Pivots
- RSIS
- Spearman Rank Correlation Coefficient
- The Stochastic CCI
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodies CCI

More information:

See updated/extended version on-line.

ceil
- ceil value

Math functions

SYNTAX	**ceil(*number*)**
	ceil(*array*)

RETURNS	NUMBER, ARRAY
FUNCTION	Calculates the lowest integer that is greater than NUMBER or ARRAY.
EXAMPLE	The formula *ceil(6.2)* returns 7; the formula *ceil(-6.2)* returns -6.
SEE ALSO	FLOOR() function , INT() function , ROUND() function

References:

The **ceil** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- For Auto Trading Setup
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- JEEVAN'S SRI CHAKRA
- Market Profile
- Murrey Math Price Lines
- N-period candlesticks (time compression)
- New HL Scanner
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- PF Chart - Close - April 2004
- Point & figure Chart India Securities
- Renko Chart
- Renko Chart
- Revised Renko chart
- Triangle exploration using P&F Chart
- Trigonometric Fit - TrigFit with AR for cos / sin
- Updated Renko Chart

More information:

See updated/extended version on-line.

Chaikin
- chaikin oscillator

Indicators

SYNTAX	**chaikin(** ***fast*** **= 9,** ***slow*** **= 14)**
RETURNS	ARRAY
FUNCTION	Calculates the Chaikin Oscillator with averaging parameters: *fast, slow*
EXAMPLE	
SEE ALSO	

References:

The **Chaikin** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Chr
- get string with given ASCII code

String manipulation
(AmiBroker 6.40)

SYNTAX	**Chr(code)**
RETURNS	STRING
FUNCTION	The function returns string representing single character of given ASCII *code*.
EXAMPLE	Chr(32); // will return " " (a string consisting of single space)
SEE ALSO	

References:

The **Chr** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

ClipboardGet - retrieves current contents of Windows clipboard

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX	**ClipboardGet()**
RETURNS	STRING
FUNCTION	Retrieves current contents of Windows clipboard

EXAMPLE

```
"Contents of the Windows clipboard" + ClipboardGet();
```

SEE ALSO ClipboardSet() function

References:

The **ClipboardGet** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

ClipboardSet - copies the text to the Windows clipboard

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX	**ClipboardSet("Text");**
RETURNS	NUMBER
FUNCTION	Copies the "text" to the Windows clipboard. Returns True (1) on success and 0 on failure

EXAMPLE

```
// this can be used to put dynamically-constructed texts into
// clipboard
//
ClipboardSet( "The price of " + FullName() + " is " + Close );
```

SEE ALSO ClipboardGet() function

References:

The **ClipboardSet** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

ColorBlend
- blends (mixes) two colors

Indicators
(AmiBroker 5.30)

SYNTAX **ColorBlend(colorFrom, colorTo, factor = 0.5)**

RETURNS NUMBER

FUNCTION The function blends (mixes) colorFrom with colorTo with 'factor' proportion using the following algorithm

RGB = (1 - factor) * RGB(colorFrom) + factor * RGB(colorTo);

So factor = 0 means use colorFrom only, factor = 1 means use colorTo only. All in-between values mean create mix of colors. The lower the factor value means more colorFrom.

EXAMPLE This function makes it easy to lighten or darken colors like this:

```
function ColorLighten( color )
{
  return ColorBlend( color, colorWhite, 0.5 );
}

function ColorDarken( color )
{
 return ColorBlend( color, colorBlack, 0.5 );
}
```

SEE ALSO ColorRGB() function , ColorHSB() function

References:

The **ColorBlend** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- BBAreacolor&TGLCROSSNEW
- Color Combination
- Fib Fan Based on ZZ
- Gfx Toolkit
- Guppy Cloud
- Ichimoku Kinko Hyo
- Intraday Trend Break System
- MACD BB Indicator
- Momentum Volume Price (MVP) Indicator
- TSV
- White Theme

More information:

See updated/extended version on-line.

ColorHSB - specify color using Hue-Saturation-Brightness

Miscellaneous functions
(AmiBroker 4.80)

SYNTAX **ColorHSB(hue, saturation, brightness)**

RETURNS NUMBER

FUNCTION The function allows to specify color out of 16 million color (24 bit) palette using Hue, Saturation and Brightness parameters.

The return value is a number that can be used in Plot, PlotOHLC, PlotForeign, AddColumn, AddTextColumn functions to specify chart or column color.

Parameters:

- **hue** - represents gradation of color within the optical spectrum (as in rainbow)
- **saturation** represents "vibrancy" of the color
- **brightness** represents brightness.

Each parameter ranges from 0 to 255, where 0 represents 0% saturation/brightness or 0 degree hue in HSV color wheel, and 255 represents 100% saturation/brightness or 360degrees hue in HSV color wheel.

When you modify hue from 0 to 255 you will see consecutive rainbow colors starting from red, through yellow and green to blue and violet.

For more information about HSB color space please read:
http://en.wikipedia.org/wiki/HSB_color_space

EXAMPLE

```
// Example 1:
// 3-d multicolor multiple moving average cloud chart
side = 1;
increment = Param("Increment",2, 1, 10, 1 );
for( i = 10; i < 80; i = i + increment )
{
     up = MA( C, i );
     down = MA( C, i + increment );

   if( ParamToggle("3D effect?", "No|Yes" ) )
     side = IIf(up<=down AND Ref( up<=down, 1 ), 1, 0.6 );

   PlotOHLC( up,up,down,down, "MA"+i, ColorHSB( 3*(i - 10),
   Param("Saturation", 128, 0, 255 ),
     side * Param("Brightness", 255, 0, 255 ) ), styleCloud |
styleNoLabel );
}

// Example 2:
///////
//Color-parade exploration
```

```
Filter=1;
for( i = 0; i < 256; i = i + 16 )
  AddColumn( C, "C", 1.2, colorDefault, ColorHSB( ( BarIndex() + i )
% 256, 255-i, 255 ) );
```

SEE ALSO ColorRGB() function , PLOT() function , AddColumn() function

References:

The **ColorHSB** function is used in the following formulas in AFL on-line library:

- Animated BackGround
- Animated BackGround 1.1
- Brian Wild
- Color Combination
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- IchimokuBrianViorelRO
- Market Profile
- Nadaraya-Watson Envelope
- pattenz
- Stan Weinstein strategy

More information:

See updated/extended version on-line.

ColorRGB - specify color using Red-Green-Blue components

Miscellaneous functions
(AmiBroker 4.80)

SYNTAX	**ColorRGB(red, green, blue)**
RETURNS	NUMBER
FUNCTION	The function allows to specify color out of 16 million color (24 bit) palette using Red, Green, Blue components. The return value is a number that can be used in Plot, PlotOHLC, PlotForeign, AddColumn, AddTextColumn functions to specify chart or column color. Parameters: red, green, blue - represent color component values in range 0..255 each For more information about RGB color model please read: http://en.wikipedia.org/wiki/RGB_color_model
EXAMPLE	Plot(MA(C,10), "Light Red", ColorRGB(255, 128, 128)); Plot(MA(C,20), "Light Green", ColorRGB(128, 255, 128)); Plot(MA(C,30), "Light Blue", ColorRGB(128, 128, 255));
SEE ALSO	ColorHSB() function , PLOT() function , AddColumn() function

References:

The **ColorRGB** function is used in the following formulas in AFL on-line library:

- ALJEHANI
- Alternative ZIG function
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trendlines using multiple timeframes
- BBAreacolor&TGLCROSSNEW
- Caleb Lawrence
- channel indicator
- Channel/S&R and trendlines
- Color Combination
- Continuous Contract Rollover
- Cycle Period
- Day Bar No
- Fib Fan Based on ZZ
- For Auto Trading Setup
- Get Moneycontrol News Snippets into Amiboker
- Gfx Toolkit
- Harmonic Pattern Detection
- Heatmap V1
- High Low Detection code
- How to add IB Option Symbols
- Hurst "Like" DE
- Intraday Range and Periods Framer

- Least Squares Channel Indicator
- Linear Candle
- MACD BB Indicator
- Meu Sistema de Trading - versão 1.0
- Momentum Volume Price (MVP) Indicator
- Multiple Ribbon Demo
- Nadaraya-Watson Envelope
- New HL Scanner
- Perceptron
- Polyfit Lines
- Probability Density & Gaussian Distribution
- Rebalancing Backtest avoiding leverage
- Renko Chart
- Support and resistance
- TD Sequential
- TSV
- Updated Renko Chart
- Volume Divided Histogram
- Volume Spread for VSA
- VSTOP (2)
- VSTOP (3)
- VWAP versus Average Price
- White Theme
- WILSON RELATIVE PRICE CHANNEL
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

Correlation - correlation

Statistical functions
(AmiBroker 3.40)

SYNTAX **correlation(ARRAY1, ARRAY2, *periods*)**

RETURNS ARRAY

FUNCTION Calculates correlation between ARRAY1 and ARRAY2 using *periods* range

For more information about correlation please check this:

http://en.wikipedia.org/wiki/Correlation

EXAMPLE

```
// the line below calculates
// Correlation between Close price AND AND Close price 5 days back
Correlation( Close, Ref( Close, -5 ), 5 );

// Built-in correlation can be re-coded with
// basic AFL functions like MA (moving average) - which
// is equivalent for "expected value" statistic term
// and few basic arithmetic operations

function Correl( x, y, number )
{
nom= MA( x * y, number ) - MA( x, number ) * MA( y, number );
denom = sqrt( MA( x ^ 2, number ) - MA( x, number ) ^ 2 ) *
sqrt( MA( y ^ 2, number ) - MA( y, number ) ^ 2 );
return nom/denom;
}

Graph0=Correlation( C, Ref( H, -2 ), 10 ); // built-in

Graph1=Correl( C, Ref( H, -2 ), 10 ); // re-coded;
```

SEE ALSO

References:

The **Correlation** function is used in the following formulas in AFL on-line library:

- Alpha and Beta and R_Squared Indicator
- correlerror
- crMathLib
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Inter-market Yield Linear Regression Divergence
- R-Squared
- Trend Exploration: Count Number of New Highs

More information:

See updated/extended version on-line.

cos - cosine

Math functions

SYNTAX	**cos(NUMBER)**
	cos(ARRAY)

RETURNS	NUMBER ARRAY
FUNCTION	Returns the cosine of NUMBER or ARRAY. Assumes that the NUMBER or ARRAY values are in radians.
EXAMPLE	cos(C)
SEE ALSO	atan() function , SIN() function

References:

The **cos** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Cybernertic Hilbert Sine Wave
- Cycle Period
- Ehlers Hilbert Transformer Indicator
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- Even Better Sinewave Indicator
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- John Ehler
- Luna Phase
- Sine Wave Indicator
- Smoothed Adaptive Momentum
- Stress with SuperSmoother
- Three Pole Butterworth
- Trigonometric Fit - TrigFit with AR for cos / sin
- Voss Predictive Filter (A Peek Into the Future)
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots

More information:

See updated/extended version on-line.

cosh
- hyperbolic cosine function

Math functions
(AmiBroker 4.80)

SYNTAX	**cosh(NUMBER)**
	cosh(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Returns the hyperbolic cosine of NUMBER or ARRAY. This function assumes that the ARRAY values are in radians.
EXAMPLE	
SEE ALSO	

References:

The **cosh** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

CreateObject - create COM object

Miscellaneous functions
(AmiBroker 3.80)

SYNTAX	**createobject(**
RETURNS	OBJECT
FUNCTION	Creates the instance of *"Server.Class"* COM object. The return value should be assigned to a variable that is used latter for calling the methods of the object. Note: this function creates the instance of the object everytime the formula is executed (the object is released automatically at the end of the formula - no explicit freeing is necessary)
EXAMPLE	myobj = CreateObject("MyOwnActiveX.Class1"); myobj.Method(1, 2, Close); // call the method of myobj COM object
SEE ALSO	CREATESTATICOBJECT() function

References:

The **CreateObject** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically
- AFL to Python COM Link
- AFL-Excel
- AllinOneAlerts - Module
- Auto Export to Gif
- Calculate composites for tickers in list files
- Chart Zoom
- Heatmap V1
- How to add IB Option Symbols
- LoadAB.vbs
- Now Send Push Notifications From Amibroker
- SectorRSI

More information:

See updated/extended version on-line.

CreateStaticObject - create static COM object

Miscellaneous functions
(AmiBroker 3.80)

SYNTAX	**createstaticobject(**
RETURNS	OBJECT
FUNCTION	Creates the single static instance (one per AmiBroker session) of *"Server.Class"* COM object. The return value should be assigned to a variable that is used latter for calling the methods of the object. This function is useful for "heavyweight" COM object like QuotesPlus ActiveX for example. Note: this function creates the instance of the object only once when the formula is executed for the first time. Then the object is cached internally for all consecutive calls. It is also shared if multiple formulas use the same object using CreateStaticObject call. The object is automatically released when AmiBroker is closed.
EXAMPLE	myobj = CreateStaticObject("MyOwnActiveX.Class1"); myobj.Method(1, 2, Close); // call the method of myobj COM object
SEE ALSO	CreateObject() function

References:

The **CreateStaticObject** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Cross - crossover check

Trading system toolbox

SYNTAX	**Cross(ARRAY1, ARRAY2)**
RETURNS	ARRAY
FUNCTION	Gives a "1" or true on the day that ARRAY1 crosses above ARRAY2. Otherwise the result is "0". To find out when ARRAY1 crosses **below** ARRAY2, use the formula cross(ARRAY2, ARRAY1)
EXAMPLE	cross(close, ema(close,9))
SEE ALSO	

References:

The **Cross** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- Adaptive Centre of Gravity
- Adaptive Cyber Cycle
- Adaptive Relative Vigour Index
- AFL Example
- AFL Example - Enhanced
- AFL-Excel
- Against all odds
- ALJEHANI
- Alphatrend
- Aroon The Advisor
- AR_Prediction.afl
- ATR Study
- Auto-Optimization Framework
- automatic trendlines using fractal patterns
- AutoTrader Basic Flow - updated Nov 18, 2008
- Awsome Oscilator
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- Bow tie
- Brian Wild
- Bull Fear / Bear Fear
- Caleb Lawrence
- CandleStick Comentary--Help needed
- CCI(20) Divergence Indicator
- Centre of Gravity
- Chandelier Exit
- Chandelier Exit
- Chandelier Exit or Advanced Trailing Stop
- Channel/S&R and trendlines
- com-out
- Commodity Channel Index
- Connors TPS

- correlerror
- Customised Avg. Profit %, Avg. Loss % etc
- Cyber Cycle
- DateNum_DateStr
- Demand Index
- Dinapoli Guru Commentary
- DMI Spread Index
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Ed Seykota's TSP: EMA Crossover System
- Effective Swing Indicator
- ekeko price chart
- Ema bands
- EMA Crossover
- Envelope System
- FastStochK FullStochK-D
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator
- Fisher Relative Vigour Index
- Follow the Leader
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Fund Screener
- Gann HiLo Indicator and System
- Gann Swing chart v41212
- hassan
- Heatmap V1
- Heikin Ashi Delta
- Hilbert Sine Wave Support & Resistance
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- IchimokuBrianViorelRO
- Index and ETF trading
- Indicator Explorer (ZigZag)
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- Lagging MA-Xover
- Last Five Trades Result Dashboard – AFL code
- Linear Candle
- MACD BB Indicator
- MACD commentary
- MACD optimize
- Meu Sistema de Trading - versão 1.0
- mitalpradip
- Modified Head & Shoulder Pattern
- Multiple Ribbon Demo
- Nadaraya-Watson Envelope
- New HL Scanner
- Open Range Breakout Trading System
- Performance Check
- Plot the Equity Curve without Backtesting?

- prakash
- PVT Trend Decider
- regavg
- Relative Momentum Index (RMI)
- Relative Strength Index
- Relative Vigour Index
- Renko Chart
- Reverse MFI Crossover
- RI - Auto Trading System
- RUTVOL timing signal with BB Scoring routine
- Sainath Sidgiddi
- Scale Out: Futures
- Sine Wave Indicator
- Stan Weinstein strategy
- STO & MACD Buy Signals with Money-Management
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic Fast%K and Full
- Stochastic optimize
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stochastics Trendlines
- Stock price AlertIf
- Stops Implementation in AFS
- Stops on percentages
- Stress with SuperSmoother
- Super Trend Indicator
- Support and Resistance
- swing chart
- TD Moving Average 2
- The D_oscillator
- Trailing Stop Loss
- Trend Analysis_Comentary
- Trend exploration with multiple timeframes
- Trend Exploration: Slope Moving Average
- Trend Following System
- Trend Trigger Factor
- TrendingRibbonArrowsADX
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- TSV
- Updated Renko Chart
- Using From and To dates from Auto Analysis in Code
- VAMA
- visual turtle trading system
- Visualization of stoploses and profit in chart
- Vivek Jain
- Volatility
- Volatility Breakout with Bollinger Bands
- Volume Occilator
- Volume Oscillator
- Volume Zone Oscillator

- Williams Alligator system
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

Cum
- cumulative sum

Moving averages, summation

SYNTAX	**Cum(ARRAY)**
	Cum(Value)

RETURNS	ARRAY
FUNCTION	Calculates a cumulative sum of the ARRAY from the first period in the chart. Note: Starting from AmiBroker 5.30, the Cum() function does NOT force using all bars. In the past versions Cum() functions effectively turned OFF QuickAFL feature by requesting all bars to be processed. Since Cum() function was popular it caused that many legacy formulas that used it were not benefiting from QuickAFL. To enable QuickAFL with such formulas now Cum() function does NOT affect required bars count (previously it forced all bars). This change may lead to different results when comparing with old versions. If you are interested in getting old behaviour and use all bars just add: SetBarsRequired(sbrAll) anywhere in your formula.
EXAMPLE	The formula cum(1) calculates an indicator that rises one point for each day since the beginning of the chart - this is an equivalent of bar number - especially useful if you want to detect the last bar: ThisIsLastBar = cum(1) == lastvalue(cum(1));
SEE ALSO	SUM() function

Comments:

Graham Kavanagh
gkavanagh [at]
e-wire.net.au
2004-08-09 07:49:35
	Sum adds up the last "n" number of bars. It sums whatever you put into the first part of the sum formula.
	
	Cum(1) adds 1 to the previous value of Cum, so the first bar is 1 and it just keeps adding one to the last bar value of cum(1).
	You can use Cum to add anything, like how many times you get rising days in the entire chart:
	
	Rise = C>O; //this gives results of 0 or 1
	TotalRise = Cum(Rise);
	
	You could limit this as well to time periods, or any other condition Example would be one for total rise days since 1995:
	
	RecentRise = C>O and Year()>=1995; //this gives results of 0 or 1
	TotalRise = Cum(RecentRise);
	
	If you wanted to know how many rising days in the last 12 bars you would use:

	LastRises = Sum(Rise,12); Hope this helps

References:

The **Cum** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic Trend-line
- automatic trendlines using fractal patterns
- Baseline Relative Performance Watchlist charts V2
- Buff Volume Weighted Moving Averages
- Bullish Percent Index 2 files combined
- Candle Stick Analysis
- Color Display.afl
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Dave Landry PullBack Scan
- Demand Index
- Double top detection
- ekeko price chart
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fib Fan Based on ZZ
- Fibonacci Internal and External Retracements
- Fre
- Fund Screener
- Gann Swing Chart
- Head & Shoulders Pattern
- Hurst Constant
- Intraday Fibonacii Trend Break System
- Last Five Trades Result Dashboard – AFL code
- Linear Regression Line & Bands
- Linear Regression Line w/ Std Deviation Channels
- McClellan Summation Index
- Modified Head & Shoulder Pattern
- Modified Momentum Finder DDT-NB
- Modified-DVI
- Monthly bar chart
- Moving Average "Crash" Test
- Multiple sinus noised
- N-period candlesticks (time compression)
- nth (1 - 8) Order Polynomial Fit
- Pattern Recognition Exploration
- Periodically ReBalance a BUY & HOLD Portfolio
- Prashanth
- Price Persistency
- Projection Oscillator

- PVT Trend Decider
- QP2 Float Analysis
- R-Squared
- Random Walk
- Regression Analysis Line
- RSI Trendlines and Wedges
- RSIS
- Square of Nine Roadmap Charts
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Stochastics Trendlines
- TD Sequential
- Trend Exploration: Count Number of New Highs
- Triangle exploration using P&F Chart
- Triangular Moving Average
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- Volatility Quality Index
- Weekly chart
- Williams Alligator system
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

CumProd - cumulative product of all array elements

Moving averages, summation
(AmiBroker 6.20)

SYNTAX **CumProd(factor)**

RETURNS ARRAY

FUNCTION The function returns cumulative product of all array elements from first bar till current bar

EXAMPLE

```
y = CumProd( 1.05 ); // 1% per-bar growth
Plot( y, "exponential growth", colorRed );
```

SEE ALSO

References:

The **CumProd** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Date - date

Date/Time
(AmiBroker 3.10)

SYNTAX	**date()**
RETURNS	STRING
FUNCTION	It is used to display the selected date in commentary / interpretation window
EXAMPLE	date()
SEE ALSO	

References:

The **Date** function is used in the following formulas in AFL on-line library:

- AccuTrack
- ADX Indicator - Colored
- AFL Example
- AFL Example - Enhanced
- AFL_Glossary_1
- Against all odds
- Another FIb Level
- BBAreacolor&TGLCROSSNEW
- Binance Data Download
- Bullish Percent Index 2004
- Candle Stick Demo
- candlestick chart for Volume/RSI/OBV
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- Chart Zoom
- Coinbase Data Download
- Color Coded Short Term Reversal Signals
- Commodity Channel Index
- Congestions detection
- Continuous Contract Rollover
- DateNum_DateStr
- Dinapoli Guru Commentary
- Double Smoothed Stochastic from W.Bressert
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's SafeZone Stop
- ElderSafeZoneStopLong
- ElderSafeZoneStopShort
- EMA Crossover Price
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- For Auto Trading Setup
- FTX Data Download
- Future Plotting of Time and Price
- Future Plotting of Time and Price

- Futures - Dollar Move Indicator
- Futures - Dollar Move Today Indicator
- Halftrend
- hassan
- Heikin Ashi Candles
- Heinkin-Ashi
- High Low Detection code
- How to add IB Option Symbols
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- Javascript Formulas for skalp and update portfolio
- JEEVAN'S SRI CHAKRA
- Last Five Trades Result Dashboard – AFL code
- lastNDaysBeforeDate
- Linear Candle
- MA Difference 20 Period
- MACD commentary
- MACD Histogram - Change in Direction
- Main price chart with Rainbow & SAR
- Market Facilitation Index VS Volume
- MFE and MAE and plot trades as indicator
- Nick
- PF Chart - Close - April 2004
- Pivot Point and Support and Resistance Points
- Pivots for Intraday Forex Charts
- Plot the Equity Curve without Backtesting?
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- Price with Woodies Pivots
- Probability Calculator
- Range Filter - Trading Strategy
- Relative Strength Index
- Relative Strength Multichart of up to 10 tickers
- Remove EOD data from intraday database
- Shares To Buy Price Graph
- Super Trend Indicator
- Support Resistance levels
- Trading ATR 10-1
- Triangular Moving Average new
- TSV
- Volume Divided Histogram
- VolumeDivAvgV3_Study_BrS
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- Woodies CCI
- Zig Zag Indicator with Valid Entry and Exit Points
- ZigZag Retracements

More information:

See updated/extended version on-line.

DateNum - date number

Date/Time
(AmiBroker 3.40)

SYNTAX	**datenum()**
RETURNS	ARRAY
FUNCTION	Returns the array with numbers that represent quotation dates coded as follows: 10000 * (year - 1900) + 100 * month + day, so 2001-12-31 becomes 1011231 and 1995-12-31 becomes 951231
EXAMPLE	datenum();
SEE ALSO	

References:

The **DateNum** function is used in the following formulas in AFL on-line library:

- BEANS-Summary of Holdings
- Bullish Percent Index 2004
- Continuous Contract Rollover
- DateNum_DateStr
- Date_To_Num(), Time_To_Num()
- Day Bar No
- Fund Screener
- Gordon Rose
- High Low Detection code
- IBD relative strength database ranker
- IBD relative strength database Viewer
- Intraday Range and Periods Framer
- lastNDaysBeforeDate
- MDYtoXLSerialDays and XLSerialDaysToDateNum
- MFE and MAE and plot trades as indicator
- PF Chart - Close - April 2004
- Rea Time Daily Price Levels
- Renko Chart
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- suresh
- TWS auto-export Executions-file parser
- Updated Renko Chart
- Weekly chart
- Zig Zag Indicator with Valid Entry and Exit Points

More information:

See updated/extended version on-line.

DateTime
- retrieves encoded date time

Date/Time
(AmiBroker 4.30)

SYNTAX **DateTime()**

RETURNS ARRAY

FUNCTION Returns array of encoded date/time values suitable for using with AddColumn and formatDateTime constant to produce date time formated according to your system settings.

VERSION 5.27 and above: It is important to understand that DateTime is not a simple number but rather bitset and two datetime values can only be reliably compared for equlity or inequality using == or != operators. Any other comparisions (less than/greater then) using normal operators > < can lead to wrong results, therefore to compare two datetime numbers you should use DateTimeDiff(arg1, arg2) which will return positive values if arg1 > arg2 and negative values if arg1 < arg2.

EXAMPLE 1. Simple date/time column

```
AddColumn( DateTime(), "Date / Time", formatDateTime );
```

2. Example (produces signal file accepted by various other programs):

```
Buy=Cross(MACD(),Signal());
Sell=Cross(Signal(), MACD());
Filter=Buy OR Sell;
SetOption("NoDefaultColumns", True );
AddColumn( DateTime(), "Date", formatDateTime );
AddColumn( IIf( Buy, 66, 83 ), "Signal", formatChar );
```

SEE ALSO

References:

The **DateTime** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_1
- AutoTrade using an Exploration
- babaloo chapora
- BEANS-Summary of Holdings
- Chart Zoom
- Congestions detection
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- elliott wave manual labelling
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Futures - Dollar Move Indicator
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- Gordon Rose
- High Low Detection code
- IBD relative strength database ranker
- Improved NH-NH scan / indicator

- Intraday Range and Periods Framer
- New HL Scanner
- PF Chart - Close - April 2004
- Pivot Finder
- Polyfit Lines
- Relative Strength Multichart of up to 10 tickers
- Scale Out: Futures
- shailu lunia
- TAPE READING
- Trades per second indicator
- Trend Exploration: Count Number of New Highs
- Trend Exploration: Slope Moving Average
- TWS trade plotter
- ValueChart

More information:

See updated/extended version on-line.

DateTimeAdd
- adds specified number of seconds/minutes/hours/days to datetime

Date/Time
(AmiBroker 5.40)

SYNTAX **DateTimeAdd(datetime, amount, interval = inDaily)**

RETURNS NUMBER or ARRAY

FUNCTION The function adds specified amount of 'intervals' to input datetime array. Allows for example to calculate datetime 5 days back or 13 months into the future.

INPUT:

- datetime - scalar or array - the datetime value to add to (for example returned by Now(), DateTime() or StrToDateTime/_DT or ConvertDateTime functions)
- amount - number of 'intervals' to add (seconds/minutes/hours/days/etc - depending on interval parameter)
- interval - specifies unit you want to add, supported units are in1Second, in1Minute, inHourly, inDaily, inWeekly, inMonthly, inQuarterly, inYearly

RETURNS:
datetime (scalar or array) - returned type depends on input

EXAMPLE

```
// Example 1:

x = Now( 5 ) ;
printf("11 days from now " + DateTimeToStr( DateTimeAdd( x, 11,
inDaily ) ) ) ;

// Example 2 (commentary):

x = Now(5);
printf( " now is " + DateTimeToStr( x ) );

for( shift = -12; shift <= 12; shift++ )
{
 printf("%g seconds from now is " + DateTimeToStr( DateTimeAdd( x,
shift, 1 ) ) + "n", shift );
 printf("%g minutes from now is " + DateTimeToStr( DateTimeAdd( x,
shift, in1Minute ) ) + "n", shift );
 printf("%g hours from now is " + DateTimeToStr( DateTimeAdd( x,
shift, inHourly ) ) + "n", shift );
 printf("%g days from now is " + DateTimeToStr( DateTimeAdd( x,
shift, inDaily ) ) + "n", shift );
 printf("%g weeks from now is " + DateTimeToStr( DateTimeAdd( x,
shift, inWeekly ) ) + "n", shift );
 printf("%g months from now is " + DateTimeToStr( DateTimeAdd( x,
shift, inMonthly ) ) + "n", shift );
 printf("%g quarters from now is " + DateTimeToStr( DateTimeAdd( x,
shift, inQuarterly ) ) + "n", shift );
 printf("%g years from now is " + DateTimeToStr( DateTimeAdd( x,
shift, inYearly ) ) + "nn", shift );
}
```

SEE ALSO DateTimeDiff() function , DateTimeToStr() function , DateTimeConvert() function , DateTime() function

References:

The **DateTimeAdd** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

DateTimeConvert - date/time format conversion

Date/Time
(AmiBroker 4.90)

SYNTAX **DateTimeConvert(format, date, time = Null)**

RETURNS NUMBER or ARRAY

FUNCTION The function allows to convert from DateTime format to DateNum and TimeNum and vice versa.

format parameter controls the direction of conversion:

- format = 0 - converts DateTime format to DateNum format, example

  mydatenum = DateTimeConvert(0, DateTime());// - this returns DateNum
  date argument should be in datetime formattime argument in this case should not be used
- format = 1 - converts DateTime format to TimeNum format, example:

  mytimenum = DateTimeConvert(1, DateTime()); // - returns timenum
  date argument should be in datetime formattime argument in this case should not be used
- format = 2 - converts from DateNum and optionally TimeNum to DateTime format, example:

  mydatetime = DateTimeConvert(2, DateNum(), TimeNum());
  date argument should be in datenum formattime argument (optional) should be in timenum format. In case of EOD data you can skip time argument:

  mydatetime = DateTimeConvert(2, DateNum());
- format = 3 - (AB5.0 or higher) converts DateTime format to Seconds (00..59) example:

  myseconds = DateTimeConvert(3, DateTime());

  date argument should be in datetime, formattime argument in this case should not be used
- format = 4 - (AB5.0 or higher) converts DateTime format to Minutes(00..59) example:

  myminute = DateTimeConvert(4, DateTime());

  date argument should be in datetime, formattime argument in this case should not be used
- format = 5 - (AB5.0 or higher) converts DateTime format to Hours (00..23) example:

  myhour = DateTimeConvert(5, DateTime());

  date argument should be in datetime, formattime argument in this case should not be used

EXAMPLE

```
mydatenum = DateTimeConvert( 0, DateTime() );// - this returns
DateNum
mytimenum = DateTimeConvert( 1, DateTime() ); // - returns timenum
mydatetime = DateTimeConvert( 2, DateNum(), TimeNum() );
mydatetime = DateTimeConvert( 2, DateNum() );
```

SEE ALSO DateNum() function , DateTime() function , DateTimeToStr() function , Day() function , DayOfWeek() function , DayOfYear() function , TIMENUM() function , MONTH() function , YEAR() function , HOUR() function , MINUTE() function , SECOND() function

References:

The **DateTimeConvert** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

DateTimeDiff
- get difference in seconds between two datetime values

Date/Time
(AmiBroker 5.30)

SYNTAX **DateTimeDiff(arg1, arg2)**

RETURNS NUMBER or ARRAY

FUNCTION DateTimeDiff(arg1, arg2) which will return positive values if arg1 > arg2 and negative values if arg1 < arg2.

The difference is given in seconds.

The function can operate on scalar and array arguments, returning scalar if both inputs are scalars, and array otherwise.

It is important to understand that DateTime is not a simple number but rather bitset and two datetime values can only be reliably compared for equlity or inequality using == or != operators. Any other comparisions (less than/greater then), using normal operators > < may sometimes lead to wrong results (if one of dates compared is pre-1964), therefore to compare two datetime numbers reliably you should use DateTimeDiff.

EXAMPLE

SEE ALSO DateTime() function

References:

The **DateTimeDiff** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- Improved NH-NH scan / indicator
- TAPE READING
- Trades per second indicator

More information:

See updated/extended version on-line.

DateTimeFormat - converts datetime to string

Date/Time
(AmiBroker 6.20)

SYNTAX **DateTimeFormat("formatstr", datetime)**

RETURNS STRING

FUNCTION The function converts datetime to string according to user-specified format.

The *formatstr* argument contains formatting string that specifies how date time should be converted to string.

Available formatting sequences are exactly like in strftime() C-runtime function strftime:

- %a - Abbreviated weekday name
- %A - Full weekday name
- %b - Abbreviated month name
- %B - Full month name
- %c - Date and time representation appropriate for locale
- %d - Day of month as decimal number (01 – 31)
- %H - Hour in 24-hour format (00 – 23)
- %I - Hour in 12-hour format (01 – 12)
- %j - Day of year as decimal number (001 – 366)
- %m - Month as decimal number (01 – 12)
- %M - Minute as decimal number (00 – 59)
- %p - Current locale’s A.M./P.M. indicator for 12-hour clock
- %S - Second as decimal number (00 – 59)
- %U - Week of year as decimal number, with Sunday as first day of week (00 – 53)
- %w - Weekday as decimal number (0 – 6; Sunday is 0)
- %W - Week of year as decimal number, with Monday as first day of week (00 – 53)
- %x - Date representation for current locale
- %X - Time representation for current locale
- %y - Year without century, as decimal number (00 – 99)
- %Y - Year with century, as decimal number
- %z, %Z - Time-zone name or abbreviation; no characters if time zone is unknown
- %% - Percent sign

Extra # flag may prefix any formatting code. In that case, the meaning of the format code is changed as follows.

Format Code Meaning

- %#a, %#A, %#b, %#B, %#p, %#X, %#z, %#Z, %#% # flag is ignored.
- %#c Long date and time representation, appropriate for current locale. For example: “Tuesday, March 14, 1995, 12:41:29”.
- %#x Long date representation, appropriate to current locale. For example: “Tuesday, March 14, 1995”.
- %#d, %#H, %#I, %#j, %#m, %#M, %#S, %#U, %#w, %#W, %#y, %#Y Remove leading zeros (if any).

EXAMPLE

```
// For example to get full week day name of date time use
```

```
str = DateTimeFormat( "%A", dt );

// To get YYYYMMDD without separtors use:
str = DateTimeFormat("%Y%m%d", dt );
```

SEE ALSO DateTimeToStr() function

References:

The **DateTimeFormat** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

DateTimeToStr - convert datetime to string

Date/Time
(AmiBroker 4.80)

SYNTAX **DateTimeToStr(NUMBER, mode = 0)**

RETURNS STRING

FUNCTION Converts number representing date/time value (for example obtained using GetCursorXPosition(), Now(), DateTime() functions) to the corresponding STRING (text).

The *mode* parameter defines the output mode:

- 0 - convert both date and time portion
- 1 - only date
- 2 - only time.
- 3 - iso date and time YYYY-MM-DD HH:MM:SS) HH:MM:SS part is only included for non-EOD records
- 4 - iso date only YYYY-MM-DD
- 5 - iso time only HH:MM:SS

Note that mode 2 would give you empty string when applied on chart with daily or longer interval

EXAMPLE

```
ToolTip="X="+DateTimeToStr(GetCursorXPosition())
+"nY="+GetCursorYPosition();
```

SEE ALSO DATETIME() function , NOW() function , StrToDateTime() function , DateTimeFormat() function

References:

The **DateTimeToStr** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_1
- AutoTrade using an Exploration
- Chart Zoom
- Congestions detection
- elliott wave manual labelling
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- GFX ToolTip
- Scale Out: Futures
- TAPE READING

More information:

See updated/extended version on-line.

Day
- day of month

Date/Time
(AmiBroker 3.40)

SYNTAX	**day()**
RETURNS	ARRAY
FUNCTION	Returns the array with days (1-31)
EXAMPLE	writeif(day() < 3, "Beginning of the month", "The rest of the month");
SEE ALSO	

References:

The **Day** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Auto Trade Step by Step
- Backup Data of 1min Interval
- changing period moving avarage
- CoinToss ver 1
- Daily High Low in Advance
- Days to Third Friday
- Expiry day/days - Last thursday of month
- Expiry Thursday for Indian markets
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- For Auto Trading Setup
- Heatmap V1
- High Low Detection code
- How to add IB Option Symbols
- Intraday Average Volume
- IntraDay Open Marker
- Intraday Strength
- Intraday Volume EMA
- Luna Phase
- Lunar Phases - original
- LunarPhase
- Market Profile
- N-period candlesticks (time compression)
- New HL Scanner
- Next Date Format
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- pattenz
- Positive Bars Percentage
- Relative Strength Multichart of up to 10 tickers
- Support and resistance
- Trade day of month
- VWAP - Volume Weighted Average Price
- VWAP with standard deviation bands

- White Theme
- Wolfe Wave Patterns

More information:

See updated/extended version on-line.

DayOfWeek
- day of week

Date/Time
(AmiBroker 3.40)

SYNTAX	**dayofweek()**
RETURNS	ARRAY
FUNCTION	Returns the array with day of week (0-6): 0 - Sunday 1 - Monday ... 5 - Friday 6- Saturday

EXAMPLE	buy = dayofweek() == 1; // buy on Monday
	sell = dayofweek() == 5; // sell on Friday

SEE ALSO	

References:

The **DayOfWeek** function is used in the following formulas in AFL on-line library:

- Basket Trading System T101
- Days to Third Friday
- Ed Seykota's TSP: EMA Crossover System
- elliott wave manual labelling
- Expiry day/days - Last thursday of month
- Expiry Thursday for Indian markets
- Fast Refreshed KAGI Swing Charts (Price Swing)
- For Auto Trading Setup
- High Low Detection code
- How to add IB Option Symbols
- MO_CrashZone
- New HL Scanner
- Option Calls, Puts and days till third friday.
- Periodically ReBalance a BUY & HOLD Portfolio
- RSI of Weekly Price Array
- Sainath Sidgiddi
- Stochastic of Weekly Price Array
- Time Frame Weekly Bars
- Weekly chart
- Weekly Trend in Daily Graph
- White Theme

More information:

See updated/extended version on-line.

DayOfYear
- get ordinal number of day in a year

Date/Time
(AmiBroker 4.40)

SYNTAX	**DayOfYear()**
RETURNS	ARRAY
FUNCTION	Returns the calendar day number counting from beginning of the year January 1st is 1. Maximum number returned is 366
EXAMPLE	`Filter=1;` `AddColumn(DayOfYear(),"Day of Year");`
SEE ALSO	DAYOFWEEK() function

References:

The **DayOfYear** function is used in the following formulas in AFL on-line library:

- End Of Year Trading

More information:

See updated/extended version on-line.

DaysSince1900 - get number of days since January 1st, 1900

Date/Time
(AmiBroker 5.20)

SYNTAX **DaysSince1900()**

RETURNS ARRAY

FUNCTION The function returns the number of days that passed since January 1st, 1900, counting from 2. January 1, 1900 is serial number 2, and January 1, 2008 is serial number 39448.

Technically is equal to Windows OLEDATE and Excel's DATEVALUE function. As to why it starts counting from 2 (two) - it is to get the same values as Excel DATEVALUE. Excel's DATEVALUE function starts counting from one but it includes Feb 29, 1900 which did not exist and this adds extra one day for all dates starting from Mar 1st, 1900.

The function can be used for calculations that involve calendar days as opposed to trading days and replaces previously proposed AFL solution
http://www.amibroker.com/kb/2007/03/15/calendar-day-index/

Now RefDays can be implemeted as follows (see example)

EXAMPLE

```
SetBarsRequired( 365, 0 );
function RefDays( Array, Days )
{
   td = DaysSince1900();
   result = Null;

  if( Days < 0 )
   {
     for( i = BarCount -1; i >= -Days; i = i - 1 )
     {
       backday = td[ i ] + Days; // Days is negative
       for( j = -Days/2; j < i; j++ )
       {
          if( td[ i - j ] <= backday )
           {
             result[ i ] = Array[ i - j ];
             break;
           }
       }
     }
   }
  return result;
}

Plot( C, "C", colorRed );
Plot( Ref( C, -252 ), "Close 252 bars back", colorBlue );
Plot( RefDays( C, -365 ), "Close 365 days back", colorGreen );
```

SEE ALSO Date() function , DateNum() function , DateTime() function , DateTimeConvert() function

References:

The **DaysSince1900** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

DecIssues - declining issues

Composites
(AmiBroker 3.20)

SYNTAX	**DecIssues()**
RETURNS	ARRAY
FUNCTION	Returns the number of declining issues for a given market (the one that currently analysed stock belongs to)
EXAMPLE	DecIssues()
SEE ALSO	

References:

The **DecIssues** function is used in the following formulas in AFL on-line library:

- Absolute Breadth Index
- Breadth Thrust
- McClellan Oscillator
- McClellan Summation Index

More information:

See updated/extended version on-line.

DecVolume - declining issues volume

Composites
(AmiBroker 3.20)

SYNTAX	**DecVolume()**
RETURNS	ARRAY
FUNCTION	Returns the volume of declining issues for a given market (the one that currently analysed stock belongs to)
EXAMPLE	DecVolume()
SEE ALSO	

References:

The **DecVolume** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

DEMA
- double exponential moving average

Moving averages, summation
(AmiBroker 40)

SYNTAX	**dema(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates double exponentially smoothed average - DEMA. The function accepts time-variable *periods.*
EXAMPLE	DEMA(Close, 5)
SEE ALSO	MA(), EMA(), WMA(), TEMA()

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2003-02-06 13:51:31	DEMA can be implemented via EMA:
	
	Len=10;
	Graph0= 2 * EMA(C, len) - EMA(EMA(C, len), Len);
	
	// for comparison only
	Graph1=DEMA(C,Len);

Tomasz Janeczko tj --at-- amibroker.com 2003-04-27 15:43:17	Note to the comment above: EMA and DEMA use different initialization method. DEMA[0] is initialized with first value of input array, while EMA[len] is initialized with simple moving average to match output with Metastock. Therefore they will converge at 2 * Len bars from Graph0 start (6 * Len bars since beginning of the data).

Tomasz Janeczko tj --at-- amibroker.com 2003-04-27 15:48:11	DEMA can also be implemented using new for looping:
	
	Len = 20;
	Plot(DEMA(Close, Len), "Built-in DEMA", colorRed);
	
	factor = 2 / (Len + 1);
	
	e1 = e2 = Close[0]; // initialize
	
	for(i = 0; i < BarCount; i++)
	{
	e1 = factor * Close[i] + (1 - factor) * e1;
	e2 = factor * e1 + (1 - factor) * e2;
	
	myDema[i] = 2 * e1 - e2;
	}
	
	Plot(myDema, "Dema in loop", colorBlue);

Tomasz Janeczko	... and can be implemented using AMA:

tj --at-- amibroker.com 2003-04-27 15:51:03	Len = 20;
	Factor = 2/(Len+1);
	
	e1 = AMA(Close, Factor);
	e2 = AMA(e1, Factor);
	Plot(DEMA(Close, Len), "Built-in DEMA", colorRed);
	Plot(2*e1 - e2, "AMA-implemened DEMA", colorBlue);

Tomasz Janeczko tj --at-- amibroker.com 2003-04-27 16:26:06	For more information on DEMA see: Stocks & Commodities V. 12:1 (11-19): Smoothing Data With Faster Moving Averages by Patrick G. Mulloy.

References:

The **DEMA** function is used in the following formulas in AFL on-line library:

- AFL to Python COM Link
- Auto-Optimization Framework
- Average Price Crossover
- babaloo chapora
- DEBAJ
- Dynamic Momentum Index
- Dynamic Momentum Index
- Hull with DEMA
- JEEVAN'S SRI CHAKRA
- Lagging MA-Xover
- Linear Candle
- Moving Averages NoX
- Support Resistance levels
- The D_oscillator
- The Saturation Indicator D_sat
- Trend Detection

More information:

See updated/extended version on-line.

EMA
- exponential moving average

Moving averages, summation

SYNTAX	**ema(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates a *periods* exponential moving average of ARRAY
EXAMPLE	ema(close, 5)
SEE ALSO	MA() function , TEMA() function , AMA() function , AMA2() function , DEMA() function , WMA() function , WILDERS() function

Comments:

Nigel Rowe rho [at] bigpond.com 2003-04-27 18:05:14	See the comments attached to DEMA for a discussion on the differences in the way EMA and others are initialised. EMA is initialised from a simple MA of equivalent length. (For compatability with some other strange TA software.) The others are initialised from the first value.

References:

The **EMA** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- accum/dist mov avg crossover SAR
- AccuTrack
- Advanced MA system
- ADXbuy
- AFL Example
- AFL Example - Enhanced
- AFL to Python COM Link
- Against all odds
- Andrews PitchforkV3.3
- AR_Prediction.afl
- ATR Study
- Auto-Optimization Framework
- Average Price Crossover
- B-Xtrender
- babaloo chapora
- Bad Tick Trim on 5 sec database
- Balance of Power
- balance of power
- BB squeeze
- BBAreacolor&TGLCROSSNEW
- BMTRIX Intermediate Term Market Trend Indicator
- Bollinger - Keltner Bands
- Bollinger band normalization
- Breadth Thrust
- Bull/Bear Volume
- candlestick chart for Volume/RSI/OBV

- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- CCT FibAccordion
- CCT Kaleidoscope
- Chaikin's Volatility
- Chandelier Exit
- Color Price Bar - Impulse System
- Compare Sectors against Tickers
- Connors TPS
- Coppock Curve
- Coppock Histogram
- Dahl Oscillator modified
- Dave Landry PullBack Scan
- Dave Landry Pullbacks
- DEBAJ
- Demand Index
- Demand Index
- Derivative Oscillator
- Divergences
- Double Smoothed Stochastic from W.Bressert
- DT Oscillator
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Effective Swing Indicator
- Elder Bear Power
- Elder Bull Power
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Ray - Bull Bear
- Elder Ray Oscillator with MA
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- Elliott Wave Oscillator
- Ema bands
- EMA Crossover
- EMA Crossover Price
- Ergodic Oscillator
- Fibonacci Moving averages
- Force index
- Forward/Reverse EMA by John Ehlers
- Fund Screener
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- Guppy Cloud
- Heatmap V1
- Heikin Ashi Delta
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- Hilbert Study
- Hull Multiple Moving Averages

- Hull with DEMA
- ICHIMOKU SIGNAL TRADER
- Index and ETF trading
- Inter-market Yield Linear Regression Divergence
- INTRADAY HEIKIN ASHI new
- JEEVAN'S SRI CHAKRA
- Lagging MA-Xover
- Laugerre PPO Oscillator
- Linear Candle
- MACD BB Indicator
- MACD Histogram - Change in Direction
- MACD-V
- Market Direction
- MAVG
- McClellan Oscillator
- McClellan Summation Index
- Meu Sistema de Trading - versão 1.0
- mfimacd
- MOCS
- Modified-DVI
- Moving Averages NoX
- MultiCycle 1.0
- Multiple Ribbon Demo
- Noor_Doodie
- Percentage Price Oscillator
- Peterson
- Pivots for Intraday Forex Charts
- Plot the Equity Curve without Backtesting?
- Polarized Fractal Efficiency
- Price with Woodies Pivots
- Range Filter - Trading Strategy
- Relative Strength
- Reverse EMA function
- RSI + Avgs
- RSIS
- RUTVOL timing signal with BB Scoring routine
- Schiff Lines
- Sector Tracking
- SectorRSI
- SF Entry,Stop, PT Indicator
- shailu lunia
- SIROC Momentum
- STD_STK Multi
- STO & MACD Buy Signals with Money-Management
- Stochastic %J - KDJ
- Stochastic of Weekly Price Array
- StochD_StochK Single.afl
- Sun&Cloud
- Support Resistance levels
- T3
- T3 Function
- TAZ Trading Method Exploration

- The D_oscillator
- The Relative Slope
- The Relative Slope Pivots
- The Saturation Indicator D_sat
- tomy_frenchy
- Trend Detection
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- TRIX
- TSV
- TTM Squeeze
- Varexlist
- Vertical Horizontal Filter (VHF)
- visual turtle trading system
- Vivek Jain
- Volatility
- Volume Occilator
- Volume Oscillator
- Volume Zone Oscillator
- Weekly Trend in Daily Graph
- Weinberg's The Range Indicator
- William's Alligator System II
- Williams Alligator system
- WILSON RELATIVE PRICE CHANNEL
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- Woodie's Price Panel With Woodie's Pivots
- z-distance from vwap
- ZeroLag MACD(p,q,r)
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

EnableRotationalTrading - Turns on rotational-trading mode of the backtester

Trading system toolbox
(AmiBroker 4.50)

SYNTAX **EnableRotationalTrading()**

RETURNS NOTHING

FUNCTION When placed on the top of system formula it turns on rotational-trading (aka. fund-switching) mode of the backtester.

Note: this function is now marked as obsolete. Use SetBacktestMode(backtestRotational) in new formulas.

IMPORTANT NOTE: Unless you specifically want to implement fund-switching/rotational trading system you should NOT use this mode.

Rotational trading is popular method for trading mutual funds. It is also known as fund-switching or scoring&ranking. Its basic permise is to **rotate symbols all the time** so only top N issues ranked according to some user-definable score are traded. The number of positions open depend on "Max. open positions" setting and available funds / position size. Once position is entered in remains in place until security's rank drops below WorstRankHeld (settable via SetOption("WorstRankHeld", 5)). **Regular buy/sell/short/cover signals are not used at all.**

The rotational mode uses only score variable (PositionScore) to rank and rotate securities. This idea has been implemented earlier in PortfolioTrader AFL formula written by Fred Tonetti with GUI written by Dale Wingo.

To enter this mode you have to call **EnableRotationalTrading()** function at the very beginning of your formula. From then on using of buy/sell/short/cover variables is not allowed. Only PositionScore variable will be used to rank securities and trade top N securities..

A simple rotational trading formula (stocks with high RSI are best candidates for shorting while stocks with low RSI are best candidates for long positions):

```
EnableRotationalTrading();
SetOption("WorstRankHeld",5);

PositionSize = -25; // invest 25% of equity in single security
PositionScore = 50 - RSI(); // PositionScore has the same meaning as
rScore in PT
```

The score (PositionScore) for all securities is calculated first. Then all scores are **sorted according to absolute value of PositionScore**. Then top N are choosen to be traded. N depends on available funds and "max. open positions" setting. Backtester successively enters the trades starting from highest ranked security until the number of positions open reaches "**max. open positions**" or there are no more funds available. The score has the following meaning:

- higher positive score means better candidate for entering long trade

- lower negative score means better candidate for entering short trade
- the score of zero means no trade (exit the trade if there is already open position on given symbol)
- the score equal to **scoreNoRotate** constant means that already open trades should be kept and no new trades entered
- the score equal to **scoreExitAll** constant causes rotational mode backtester to exit all positions regardless of HoldMinBars. Note that this is global flag and it is enough to set it for just any single symbol to exit all currently open positions, no matter on which symbol you use scoreExitAll (it may be even on symbol that is not currently held). By setting PositionScore to scoreExitAll you exit all positions immediatelly regardless of HoldMinBars setting

Exits are generated automatically when security's rank drops below "worst rank held". There is no real control over when exits happen except of setting low score to force exits. You can also set the score on any (at least one) security to value of scoreNoRotate to prevent rotation (so already open positions are kept). But this is global and does not give you individual control.

Important:
The rotational trading mode uses "buy price" and "buy delay" from the Settings | Trade page as trade price and delay for both entries and exits (long and short)

EXAMPLE

```
EnableRotationalTrading();
SetOption("WorstRankHeld",5);

PositionSize = -25; // invest 25% of equity in single security
PositionScore = 50 - RSI(); // PositionScore has the same meaning as
rScore in PT
```

SEE ALSO SetBacktestMode() function

References:

The **EnableRotationalTrading** function is used in the following formulas in AFL on-line library:

- Relative Strength

More information:

See updated/extended version on-line.

EnableScript - enable scripting engine

Miscellaneous functions

SYNTAX	**EnableScript("enginename")**
RETURNS	NOTHING
FUNCTION	Enables AFL scripting host.

Parameter: *enginename* - specifies which scripting language will be used. Allowable values:

- "jscript" - standard JScript engine (the same as in pre-6.40 versions),
- "vbscript" - VBScript engine
- "chakra" - (new in AmiBroker 6.40) - newer version of JScript engine featuring native JSON object.

See also: AFL scripting host

EXAMPLE

```
EnableScript( "jscript" );
EnableScript("vbscript");
```

SEE ALSO

References:

The **EnableScript** function is used in the following formulas in AFL on-line library:

- accum/dist mov avg crossover SAR
- AFL-Excel
- AllinOneAlerts - Module
- AR_Prediction.afl
- Bullish Percent Index 2 files combined
- danningham penetration
- Heatmap V1
- Hilbert Study
- Kagi Chart
- Monthly bar chart
- Now Send Push Notifications From Amibroker
- nth (1 - 8) Order Polynomial Fit
- P&F chart with range box sizes
- Polarized Fractal Efficiency
- Prashanth
- QP2 Float Analysis
- Standard Error Bands
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Float Channel Lines
- tomy_frenchy
- Trend Continuation Factor
- Triangle exploration using P&F Chart
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

EnableTextOutput
- allows to enable or disable text output

Miscellaneous functions
(AmiBroker 4.20)

SYNTAX	**EnableTextOutput(mode)**
RETURNS	NOTHING
FUNCTION	Enables or disables text output in commentary/interpretation/report charts.

mode parameter has the following meaning:

- 0 - disable output of string literals and assignment in interpretation/commentary
- 1 - enable output of string literals and assignment in interpretation/commentary
- 2 - reserved for future use
- 3 - enable HTML text output in Report Charts (See Profit Table.afl for example code)

EXAMPLE

SEE ALSO

References:

The **EnableTextOutput** function is used in the following formulas in AFL on-line library:

- Another FIb Level
- Auto Trade Step by Step
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- Elder Impulse Indicator V2
- Heatmap V1
- Manual Bracket Order Trader
- Profit Table (Color Coded)
- Square of Nine Roadmap Charts
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

EncodeColor - encodes color for indicator title

Exploration / Indicators
(AmiBroker 4.20)

SYNTAX **EncodeColor(*colorIndex*)**

RETURNS STRING

FUNCTION Converts color index to string escape sequence that changes color of text output in chart title. Color escape sequence uses cXX sequence where XX is 2 digit number specifying color index c38 - defines violet, there is a special sequence c-1 that resets to default axis color.

EXAMPLE Title = "This is written in " + **EncodeColor**(colorViolet) + "violet color " + **EncodeColor**(colorGreen) + "and this in green"; **SEE ALSO** Using colors in Indicator Builder

SEE ALSO

References:

The **EncodeColor** function is used in the following formulas in AFL on-line library:

- 'R' Channel
- ADX Indicator - Colored
- Andrews PitchforkV3.3
- Another FIb Level
- automatic trendlines using fractal patterns
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- channel indicator
- Color Display.afl
- Coppock Trade Signal on Price Chart
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- EMA Crossover Price
- FastStochK FullStochK-D
- Fibonacci Internal and External Retracements
- For Auto Trading Setup
- Fre
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Futures - Dollar Move Indicator
- Futures - Dollar Move Today Indicator
- Graphical sector analysis
- Graphical sector stock amalysis
- High Low Detection code
- How to add IB Option Symbols
- Hurst "Like" DE
- IBD relative strength database Viewer
- Improved NH-NH scan / indicator
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- Last Five Trades Result Dashboard – AFL code

- Least Squares Channel Indicator
- Linear Candle
- MACD BB Indicator
- mitalpradip
- N Line Break Chart
- New HL Scanner
- P&F chart with range box sizes
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- Price with Woodies Pivots
- Relative Strength Multichart of up to 10 tickers
- Robert Antony
- RSI of Weekly Price Array
- RUTVOL timing signal with BB Scoring routine
- Schiff Lines
- Shares To Buy Price Graph
- Square of Nine Roadmap Charts
- Stochastic of Weekly Price Array
- Super Trend Indicator
- Support Resistance levels
- TD Sequential
- Trading ATR 10-1
- Triangular Moving Average
- Triangular Moving Average new
- TSV
- Vertical Horizontal Filter (VHF)
- Volume Occilator
- Volume Spread for VSA
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- ZigZag Hi Lo Barcount

More information:

See updated/extended version on-line.

EndValue
- value of the array at the end of the selected range

Date/Time
(AmiBroker 4.30)

SYNTAX **EndValue(ARRAY)**

RETURNS NUMBER

FUNCTION This function gives the single value (number) of the ARRAY at the end of the selected range. If no range is marked then the value at the last bar is returned.

To select the range you have to double click in the chart at the beginning of the range and then double click in the chart at the end of the range. Then > < markers will appear above date axis.

EXAMPLE 1. Simple commentary:

```
WriteVal( BeginValue( DateTime() ), formatDateTime );
WriteVal( EndValue( DateTime() ), formatDateTime );
"Precentage change of close is " +
WriteVal( 100 * (EndValue( Close ) - BeginValue( Close
))/BeginValue( Close ) ) + "%";
```

2. Get the number of bars in the range and calculate some stats for that range:

```
Period = EndValue( BarIndex() ) - BeginValue( BarIndex() );
StandardDeviationInTheRange = EndValue( StDev( Close, Period ) );
```

SEE ALSO BEGINVALUE() function

References:

The **EndValue** function is used in the following formulas in AFL on-line library:

- Adaptive Price Channel
- Another FIb Level
- CAMSLIM Cup and Handle Pattern AFL
- Congestions detection
- Futures - Dollar Move Indicator
- Multi-color Volume At Price (VAP)
- nth (1 - 8) Order Polynomial Fit
- Relative Strength Multichart of up to 10 tickers
- Trend Lines from 2 points

More information:

See updated/extended version on-line.

Equity
- calculate single-symbol equity line

Trading system toolbox
(AmiBroker 40)

SYNTAX **equity(*Flags* = 0, *RangeType* = -1, *From* = 0, *To* = 0)**

RETURNS ARRAY

FUNCTION NOTE: This function is left here for backward compatibility and is using old, single-security backtester. New coding should rather use portfolio-level equity (special ~~~EQUITY ticker).

Function:
Returns single-security Equity line based on buy/sell/short/cover rules, buy/sell/short/coverprice arrays, all apply stops, and all other backtester settings. *Flags* - defines the behaviour of Equity function

0 : (default) Equity works as in 3.98 - just calculates the equity array
1 : works as 0 but additionally updates buy/sell/short/cover arrays so all redundant signals are removed exactly as it is done internally by the backtester plus all exits by stops are applied so it is now possible to visualise ApplyStop() stops.
2 : (advanced) works as 1 but updated signals are not moved back to their original positions if buy/sell/short/cover delays set in preferences are non-zero. Note: this value of flag is documented but in 99% of cases should not be used in your formula. Other values are reserved for the future.

RangeType - defines quotations range being used:

-1 : (default) use range set in the Automatic analysis window
0 : all quotes
1 : n last quotes (n defined by 'From' parameter)
2 : n last days (n defined by 'From' parameter)
3 : From/To dates

From : defines start date (datenum) (when RangeType == 3) or "n" parameter (when RangeType == 1 or 2)
To: defines end date (datenum) (when RangeType == 3) otherwise ignored

datenum defines date the same way as DateNum() function as YYYMMDD
where YYY is (year - 1900), MM is month, DD is day

December 31st, 1999 has a datenum of 991231
May 21st, 2001 has a datenum of 1010521 All these parameters are evaluated at the time of the call of Equity function. Complete equity array is generated at once. Changes to buy/sell/short/cover rules made after the call have no effect. Equity function can be called multiple times in single formula.

IMPORTANT NOTE: Equity() function uses so called "old" single-security backtester that offers only subset of features of new backtester. To retrieve value of portfolio-level equity generated by new backtester use Foreign("~~~EQUITY", "C").

EXAMPLE
```
Buy = //your Buy rule;
```

```
            Sell = //your Sell rule;
            Graph0 = Equity();
```

SEE ALSO

Comments:

Herman van den Bergen psytek [at] magma.ca 2003-02-23 09:46:19	When the Equity function is called multiple times in a single formula one must be carefull when using it with ApplyStop(). Tomasz wrote: "Equity(1) changes buy/sell variables (evaluates stops - and writes them back to buy/sell arrays). If you are using non-zero delays both Equity calls will return different values because in first case exits are generated by stops (not delayed) and in second case STOP signals written back to buy/sell arrays are delayed (opposite to the first case). Equity(1) affects the buy/sell variables. It is not a "no-operation" function. If you want a "no-op" you should use Equity(0) to generate equity line. This is by design and described in the User's Guide. AFL reference: Equity function and chart
Tomasz Janeczko tj --at-- amibroker.com 2003-05-21 17:56:46	Using Equity(1) evaluates stops and writes BACK signals to sell/cover arrays. Equity(1) also removes all extra signals. Depending on kind of the stop various values are written back to sell/cover array to enable you to distinguish if given signal was generated by regular rule or by stop. 1 - regular exit 2 - max. loss 3 - profit target 4 - trailing 5 - n-bar stop 6 - ruin stop ... your rules... ApplyStop(stopTypeTrail, stopModePercent, 10, True); Equity(1); WriteIf(sell == 1, "Regular exit", WriteIf(sell == 4, "Trailing stop", ""));
Tomasz Janeczko tj --at-- amibroker.com 2003-05-29 05:27:07	When your formula uses Equity(1) you should avoid using built-in delays. Here is a story why: Only BACKTESTER implements delays while EXPOLORATION and other modes do NOT. Therefore Equity(1) must not delay signals by itself. However in order to perform equity calculations delays must be applied to match backtester output, so AmiBroker when it encounters Equity(1) applies the delays (even in exploration, indicator,

	etc) but just before end of the equity call AmiBroker must ADJUST BACK the signals, so Equity-adjusted buy/sell/short/cover arrays do NOT have delay applied. This involves shifting updated bars back and this may cause problem if signal occurs on the very last bar (because it is moved by delay outside the range). To disable this shifting back in exploration (so exploration matches the output of backtester with NON-ZERO delays) you need to use Equity(2) On the other hand using Equity(2) in backtest formula causes double delay. (one added by the equity function, second added by the backtester pass). Built-in delays are designed to be used in BACKTESTER ONLY. The intention is as follows: set non zero delays in the settings now SINGLE formula can be used to BACKTEST and to get TODAY SIGNALS for trading for tommorrow (in SCAN) mode. Solution 1: Embbed delays in the AFL code itself: Buy = Ref(Buy, -1); Sell = Ref(Sell, -1); Solution 2: When using Equity AND EXPLORATION Use EQUITY(2) but except of backtest mode if(Status("action") != 5) e = Equity(2);
Tomasz Janeczko tj --at-- amibroker.com 2006-08-13 08:30:15	IMPORTANT NOTE: Equity() function is using OLD backtester that is missing some recently added features such as multiple-currency handling and scaling in/out. New code should rather use new, portfolio-level backtester, i.e. ~~~EQUITY special ticker. Refer to: http://www.amibroker.com/guide/a_equity.html for details about differences between new and old backtesters.

References:

The **Equity** function is used in the following formulas in AFL on-line library:

- 'R' Channel
- Add Nifty 50 IB Equity Symbol Automatically
- AFL-Excel
- Auto-Optimization Framework
- Chandelier Exit or Advanced Trailing Stop
- CoinToss ver 1
- Ed Seykota's TSP: EMA Crossover System
- Kelly criterion

- Last Five Trades Result Dashboard – AFL code
- Plot the Equity Curve without Backtesting?
- Scale Out: Futures
- SectorRSI
- Visualization of stoploses and profit in chart

More information:

See updated/extended version on-line.

erf
- calculates Gauss error function (erf)

Math functions
(AmiBroker 6.40)

SYNTAX	**erf(x)**
RETURNS	NUMBER or ARRAY
FUNCTION	Returns Gauss error function (erf) for given argument x. For the details on Error Function see: https://en.wikipedia.org/wiki/Error_function
EXAMPLE	
SEE ALSO	inverf() function

References:

The **erf** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Error
- displays user-defined error message and stops the execution

Miscellaneous functions
(AmiBroker 5.70)

SYNTAX **Error("text")**

RETURNS NOTHING

FUNCTION The function stops the formula execution in the place of the call and displays user-specified error message. It may be also uses from plugins to display error messages.

EXAMPLE

```
if( LowestVisibleValue( Close ) < 0 ) )
{
  Error( "Close price is negative");
}
```

SEE ALSO

References:

The **Error** function is used in the following formulas in AFL on-line library:

- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Backup Data of 1min Interval
- Button trading using AB auto trading interface
- Create a list of functions in your program
- Manual Bracket Order Trader
- Tracking Error
- Tracking Error

More information:

See updated/extended version on-line.

EXP
- exponential function

Math functions

SYNTAX	**exp(NUMBER)**
	exp(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Calculates **e** raised to the NUMBER or ARRAY power.
EXAMPLE	
SEE ALSO	The log() function

References:

The **EXP** function is used in the following formulas in AFL on-line library:

- Andrews Pitchfork
- Andrews PitchforkV3.3
- Arnaud Legoux Moving Average (ALMA)
- AR_Prediction.afl
- Auto-Optimization Framework
- Black Scholes Option Pricing
- Demand Index
- Demand Index
- Ehlers Fisher Transform
- Ehlers Hilbert Transformer Indicator
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- Even Better Sinewave Indicator
- Log Time Scale
- MultiCycle 1.0
- Multiple Ribbon Demo
- Nadaraya-Watson Envelope
- Neural Network Powered Smooth/Predictive RSI V2
- Option Calls, Puts and days till third friday.
- Probability Calculator
- Probability Density & Gaussian Distribution
- Schiff Lines
- Smoothed Adaptive Momentum
- Stress with SuperSmoother
- Three Pole Butterworth
- Trigonometric Fit - TrigFit with AR for cos / sin
- Volume Occilator

More information:

See updated/extended version on-line.

ExRem
- remove excessive signals

Trading system toolbox
(AmiBroker 3.50)

SYNTAX	**exrem(ARRAY1, ARRAY2)**
RETURNS	ARRAY
FUNCTION	removes excessive signals: returns 1 on the first occurence of "true" signal in Array1 then returns 0 until Array2 is true even if there are "true" signals in Array1

EXAMPLE	buy = ExRem(buy, sell);
	sell = ExRem(sell, buy);

SEE ALSO	

References:

The **ExRem** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- 4% Model - Determine Stock Market Direction
- Abhimanyu
- Advanced MA system
- ADXbuy
- AFL Example
- AFL Example - Enhanced
- Alphatrend
- An n bar Reversal Indicator
- Andrews PitchforkV3.3
- ATR Study
- Auto-Optimization Framework
- Automatic Trendlines using multiple timeframes
- Awsome Oscilator
- BBAreacolor&TGLCROSSNEW
- Bollinger band normalization
- Bull Fear / Bear Fear
- Caleb Lawrence
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- Chandelier Exit or Advanced Trailing Stop
- Channel/S&R and trendlines
- Connors TPS
- Demand Index
- DMI Spread Index
- Double Super Trend System
- Ema bands
- Envelope System
- FastStochK FullStochK-D
- Fib Fan Based on ZZ
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Gann HiLo Indicator and System
- Halftrend

- Harmonic Pattern Detection
- Hilbert Study
- Hull Moving Average
- ICHIMOKU SIGNAL TRADER
- JEEVAN'S SRI CHAKRA
- Nadaraya-Watson Envelope
- OBV with Linear Regression
- Open Range Breakout Trading System
- Perceptron
- Peterson
- Pivot End Of Day Trading System
- Plot the Equity Curve without Backtesting?
- Pullback System No. 1
- Range Filter - Trading Strategy
- Rebalancing Backtest avoiding leverage
- Relative Momentum Index (RMI)
- Reverse MFI Crossover
- RSI Double-Bottom
- RSIS
- RUTVOL timing signal with BB Scoring routine
- Schiff Lines
- SectorRSI
- Stan Weinstein strategy
- STD_STK Multi
- Stochastic Fast%K and Full
- StochD_StochK Single.afl
- Stress with SuperSmoother
- Super Trend Indicator
- TD sequential
- TD Sequential
- The D_oscillator
- The Three Day Reversal
- Trading ATR 10-1
- Trend Continuation Factor
- Triangular Moving Average new
- TTM Squeeze
- Using From and To dates from Auto Analysis in Code
- Visualization of stoploses and profit in chart
- Vivek Jain
- Williams Alligator system
- Woodie's CCI Panel Full Stats
- Zig Zag Indicator with Valid Entry and Exit Points
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

ExRemSpan
- remove excessive signals spanning given number of bars

Trading system toolbox
(AmiBroker 40)

SYNTAX **exremspan(ARRAY1, *numbars*)**

RETURNS ARRAY

FUNCTION Removes excessive signals that span *numbars* bars since initial signal.
(In other words first non-zero bar passes through, then all subsequent non-zero bars are ignored (zero is returned) until *numbars* bars have passed since initial signal. From then on a new signal may pass through)

This function is marked as obsolete.

To implement N-bar stop you should use ApplyStop function instead.

EXAMPLE

SEE ALSO ApplyStop() function

References:

The **ExRemSpan** function is used in the following formulas in AFL on-line library:

- ekeko price chart
- SectorRSI

More information:

See updated/extended version on-line.

fclose
- close a file

File Input/Output functions
(AmiBroker 4.50)

SYNTAX **fclose(filehandle)**

RETURNS NOTHING

FUNCTION Closes a file.

The filehandle (NUMBER) should be the handle returned by **fopen** function.

EXAMPLE

```
fh = fopen( "myfile.txt", "w");
if( fh )
{
   fputs( "Testing", fh );
   fclose( fh );
}
```

SEE ALSO fopen() function , fputs() function , fgets() function

References:

The **fclose** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- Advanced Search and Find
- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrade using an Exploration
- Backup Data of 1min Interval
- Continuous Contract Rollover
- Create a list of functions in your program
- elliott wave manual labelling
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- Extract specific lines of code from your program
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Herman
- IBD relative strength database ranker
- IBD relative strength database Viewer
- Say Notes
- TWS auto-export Executions-file parser
- TWS trade plotter
- WLBuildProcess

More information:

See updated/extended version on-line.

fdelete
- deletes a file

File Input/Output functions
(AmiBroker 4.70)

SYNTAX	**fdelete("filename")**
RETURNS	NUMBER
FUNCTION	This function deletes a file. "filename" is path to the file name (relative or full path). If just file name without path is specified then AmiBroker directory is used, returns TRUE if file successfully deleted, FALSE otherwise
EXAMPLE	
SEE ALSO	fopen() function , fclose() function

References:

The **fdelete** function is used in the following formulas in AFL on-line library:

- Backup Data of 1min Interval
- TWS auto-export Executions-file parser

More information:

See updated/extended version on-line.

fdir
- list directory content

File Input/Output functions
(AmiBroker 5.70)

SYNTAX **fdir("wildcard", flags = 1)**

RETURNS STRING

FUNCTION The function returns comma separated directory list (like "DIR" command)

"wildcard" is a path with a wildcard pattern to match.

For example "C:\*.*" will give you all files in C: drive, but "C:\*.txt" will give you only files with .txt extension.

flags - controls what is returned the default is 1 - only files, 2 - only directories, 3 - both files and directories

EXAMPLE

```
printf( "Only files: " );
_N( list = fdir( "c:\*.*", 1 ) );

for ( i = 0; ( filename = StrExtract( List, i ) ) != ""; i++ )
{
    printf( filename + " " );
}

printf( " Only directories: " );

_N( list = fdir( "c:\*.*", 2 ) );

for ( i = 0; ( filename = StrExtract( List, i ) ) != ""; i++ )
{
    printf( filename + " " );
}

printf( " Both files and directories: " );

_N( list = fdir( "c:\*.*", 3 ) );

for ( i = 0; ( filename = StrExtract( List, i ) ) != ""; i++ )
{
    printf( filename + " " );
}
```

SEE ALSO

References:

The **fdir** function is used in the following formulas in AFL on-line library:

- Backup Data of 1min Interval

More information:

See updated/extended version on-line.

feof
- test for end-of-file

File Input/Output functions
(AmiBroker 4.50)

SYNTAX **feof(*filehandle*)**

RETURNS NUMBER

FUNCTION The feof function returns a nonzero value after the first read operation that attempts to read past the end of the file. It returns 0 if the current position is not end of file. There is no error return value.

filehandle is a file handle returned by **fopen** function.

EXAMPLE

```
//
// The following code (commentary) reads
// all lines from the external file and
// displays it in commentary window

fh = fopen( "quotes.csv", "r");
if( fh )
{
   while( ! feof( fh ) )
   {
      printf( fgets( fh ) );
   }
}
else
{
   printf("ERROR: file can not be found (does not exist)");
}
```

SEE ALSO fopen() function , fclose() function , fgets() function

References:

The **feof** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- AFL_Glossary_Converter
- Calculate composites for tickers in list files
- Create a list of functions in your program
- Extract specific lines of code from your program
- Herman
- IBD relative strength database Viewer
- MFE and MAE and plot trades as indicator
- Say Notes
- TWS auto-export Executions-file parser
- TWS trade plotter
- WLBuildProcess

More information:

See updated/extended version on-line.

FFT
- performs Fast Fourier Transform

Basic price pattern detection
(AmiBroker 4.90)

SYNTAX **FFT(array, len = 0)**

RETURNS ARRAY

FUNCTION The function performs FFT (Fast Fourier Transform) on last 'len' bars of the array, if len is set to zero, then FFT is performed on entire array. len parameter must be even.

Result:

function returns array which holds FFT bins for first 'len' bars. There are len/2 FFT complex bins returned, where bin is a pair of numbers (complex number): first is real part of the complex number and second number is the imaginary part of the complex number.

result = FFT(array, 256);

where:

- 0th bin (result[0] and result[1]) represents DC component,
- 1st bin (result[1] and result[2]) represents real and imaginary parts of lowest frequency range and so on upto result[len - 2] and result[len - 1]

remaining elements of the array are set to zero.

IMPORTANT note: input array for FFT must NOT contain any Null values. Use Nz() function to convert Nulls to zeros if you are not sure that input array is free from nulls.

FFT bins are complex numbers and do not represent real amplitude and phase. To obtain amplitude and phase from bins you need to convert inside the formula. The following code snipplet does that:

```
ffc = FFT(data,Len);
for( i = 0; i < Len - 1; i = i + 2 )
{
   amp[ i ] = amp[ i + 1 ] = sqrt(ffc[ i ]^ 2  +  ffc[ i + 1 ]^2);
   phase[ i ] = phase[ i + 1 ] = atan2( ffc[ i + 1], ffc[ i ] );
}
```

EXAMPLE

```
SetBarsRequired(100000,100000);

Len = Param("FFT Length", 1024, 64, 10000, 10 );

Len = Min( Len, BarCount );

x = BarIndex();
x1 = x - BarCount + Len;


input = C;
```

```
a = LastValue( LinRegIntercept( input, Len - 1 ) );
b = LastValue( LinRegSlope( input, Len - 1 ) );


Lr = a + b * x1;

data = input - Lr;// de-trending

ffc = FFT(data,Len);

for( i = 0; i < Len - 1; i = i + 2 )
{
   amp[ i ] = amp[ i + 1 ] = sqrt(ffc[ i ]^ 2  +  ffc[ i + 1 ]^2);
   phase[ i ] = phase[ i + 1 ] = atan2( ffc[ i + 1], ffc[ i ] );
}

auto = ParamToggle("Auto dominant cycle", "No|Yes", 1 );
sbar = Param( "Which FFT bin", 1, 0, 50 );

skipbin1 = ParamToggle("Skip 1st FFT bin", "No|Yes", 1 );

if( auto )
{
   sbar = int( LastValue(ValueWhen( amp == LastValue(Highest( IIf(
skipbin1 AND x < 4, 0 , amp ) )), x / 2 )) );
}

fv = Status("firstvisiblebar");

thisbar = Ref( int(x/2) == sbar, -fv);
Plot( Ref(amp,-fv),
"amplitude (bin " + Ref( int(x/2), -fv ) +")", IIf( thisbar,
colorRed, colorBlack ),styleArea);

Plot( IIf( BarCount - BarIndex() < Len, data, Null ) ,
"de-trended input ("+Len+" bars)", colorOrange, styleLeftAxisScale
);
Plot( cos( phase[ sbar * 2 ] + (sbar) * x1 * 2 * 3.1415926 / Len ),
" dominant cycle "+ Len/(sbar) + "(" + sbar + " bin) bars",
colorBlue, styleOwnScale );

GraphZOrder=1;
GraphXSpace = 10;
```

SEE ALSO atan2() function

References:

The **FFT** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

fgetcwd - get current working directory

File Input/Output functions
(AmiBroker 6.10)

SYNTAX	**fgetcwd()**
RETURNS	STRING
FUNCTION	The fgetcwd function returns a full path of current working directory. Current working directory is set by "Start in" property of program icon and it is used for all relative paths within the program.
EXAMPLE	
SEE ALSO	fopen() function , fclose() function , fdir() function

References:

The **fgetcwd** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

fgets
- get a string from a file

File Input/Output functions
(AmiBroker 4.50)

SYNTAX **fgets(*filehandle*)**

RETURNS STRING

FUNCTION The **fgets** function reads a string from the input file (defined by *filehandle* argument) and returns it as a result.

fgets reads characters from the current file position to and including the first newline character, or to the end of the file whichever comes first. The newline character, if read, is included in the returned string.

The *filehandle* argument is a number returned by **fopen** function. The file has to be opened with "r" flag (for reading).

NOTE: fgets() reads maximum 1024 characters per line. Lines longer than that can be read using sequential calls to fgets() and "adding" (concatenating) results.

EXAMPLE

```
//
// The following code (commentary) reads
// all lines from the external file and
// displays it in commentary window

fh = fopen( "quotes.csv", "r");
if( fh )
{
   while( ! feof( fh ) )
   {
      printf( fgets( fh ) );
   }
}
else
{
   printf("ERROR: file can not be found (does not exist)");
}
```

SEE ALSO fopen() function , fclose() function

References:

The **fgets** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- AFL_Glossary_Converter
- Calculate composites for tickers in list files
- Create a list of functions in your program
- Extract specific lines of code from your program
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Herman

- IBD relative strength database Viewer
- MFE and MAE and plot trades as indicator
- Say Notes
- TWS auto-export Executions-file parser
- TWS trade plotter
- WLBuildProcess

More information:

See updated/extended version on-line.

fgetstatus
- retrieves file status/properties

File Input/Output functions
(AmiBroker 4.90)

SYNTAX **fgetstatus(filename, what, format = 0)**

RETURNS NUMBER or STRING

FUNCTION The function that retrieves file properties/status.

Returns NUMBER or STRING depending on format parameter. If file does not exist it returns Null.

Parameters:

- filename - the name of the file (with or without full path) to query
- what - specifies what file property to retrieve, allowable values
  0 - the date/time the file was created
  - 1 - the date/time the file was last modified
  - 2 - the date/time the file was last accessed for reading
  - 3 - the file size in bytes
  - 4 - attribute byte of the file
- format - specifies return format of date/time values (format specifications are the same as in Now() function): allowed values:
  - 0 - returns string containing date/time formatted according to system settings
  - 1 - returns string containing date only formatted according to system settings
  - 2 - returns string containing time only formatted according to system settings
  - 3 - returns DATENUM number with date
  - 4 - returns TIMENUM number with time
  - 5 - returns DATETIME number with date/time
  - 6 - returns date DAY (1..31)
  - 7 - returns date MONTH (1..12)
  - 8 - returns date YEAR (four digit)
  - 9 - returns date DAY OF WEEK (1..7, where 1=Sunday, 2=Monday, and so on)
  - 10 - returns date DAY OF YEAR (1..366)

  Note that Windows supports only 2 second resolution of file date/time stamps.

EXAMPLE
```
// get modification date string of portfolio.afl file
fgetstatus("formulas\\Equity\\portfolio.afl",1,0);
```

SEE ALSO fclose() function , fdelete() function , feof() function , fgets() function , fmkdir() function , fopen() function , fputs() function , frmdir() function

References:

The **fgetstatus** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

FindIndex
- find index of array item matching specified value

Miscellaneous functions
(AmiBroker 6.40)

SYNTAX **FindIndex(array, value, start_from = 0, dir = 1)**

RETURNS NUMBER

FUNCTION Find an index of array item matching specified value

Returns:
NUMBER - the index of matching array item if value is not found in the array, the function returns -1

Parameters:

- array - input data
- value - what we are looking for
- start - the index to start the search from
- dir - the direction of the search, 1 is up, -1 is down

Notes: start can be positive - then it refers to index number counting from the beginning of array or it can be negative then it refers to index counting from the END of the array

EXAMPLE

```
// example find all indices when condition was true

condition = Cross( C, MA( C, 20 ) );

// search forwards
printf("Search forwards:\n");
for( index = 0; ( index = FindIndex( condition, True, index ) ) !=
-1; index++ )
{
   printf("condition is true at index %g\n", index );
}

// search backwards
printf("Search backwards:\n");
for( index = -1; ( index = FindIndex( condition, True, index, -1 ) )
!= -1; index-- )
{
   printf("condition is true at index %g\n", index );
}
```

SEE ALSO

References:

The **FindIndex** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

FIR - Finite Impulse Response filter

Moving averages, summation
(AmiBroker 5.40)

SYNTAX **FIR(array, coefficients, size)**

RETURNS ARRAY

FUNCTION This function implements general-purpose Finite Impulse Response filter (FIR) http://en.wikipedia.org/wiki/Finite_impulse_response

The output is convolution of input aray with coefficents table (impulse response table).

The function performs automatic normalization of coefficient table if necessary (if its sum is not 1)

INPUTS:

- array - input array to be smoothed
- coefficients - array of FIR coefficients
- size - number of coefficients (filter_order + 1)

It is functional (but 2+ times faster) equivalent of following AFL:

```
function FIR_AFL( input, coeff, size )
{
 result = 0;
 sumcoeff = 0;
 for( i = 0; i < Min( size, BarCount ); i++ )
 {
   sumcoeff += coeff[ i ];
   result += coeff[ i ] * Ref( input, - size + i + 1 );
 }

 return result/sumcoeff;
}
```

EXAMPLE

```
// for example 5 bar weighted moving average can be written:
coeff[ 0 ] = 1;
coeff[ 1 ] = 2;
coeff[ 2 ] = 3;
coeff[ 3 ] = 4;
coeff[ 4 ] = 5;

FIR( Close, coeff, 5 );

// Another example: ALMA which is FIR Filter with
// shifted Gaussian coefficents can be implemented as follows:

function ALMA_AFL( input, range, Offset, sigma )
{
  local m, im, s, Coeff;
```

```
 m = floor( Offset * (range - 1) );
 s = range / sigma;

 for( i = 0; i < Min( range, BarCount ); i++ )
 {
   im = i - m;
   Coeff[ i ] = exp( - ( im * im )/ ( 2 * s * s ) );
 }

 return FIR( input, Coeff, range );
}
```

SEE ALSO

References:

The **FIR** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

FirstVisibleValue - get first visible value of array

Indicators
(AmiBroker 5.40)

SYNTAX **FirstVisibleValue(array)**

RETURNS NUMBER

FUNCTION When used in charts / indicators /interpretation the function returns the values of array at first visible bar.

In other (non-indicator) modes the function returns array elements with subscripts of 0.

Note that these functions do not affect QuickAFL, so LastVisibleValue may be used instead of LastValue. These functions are intended to complement functionality already available via HighestVisibleValue and LowestVisibleValue.

EXAMPLE

```
x = C;
Plot( x, "x", colorRed );
Plot( FirstVisibleValue( x ), "fvv", colorGreen );
Plot( LastVisibleValue( x ), "lvv", colorBlue );
```

SEE ALSO HighestVisibleValue() function , LowestVisibleValue() function , LastVisibleValue() function

References:

The **FirstVisibleValue** function is used in the following formulas in AFL on-line library:

- Congestions detection
- Harmonic Pattern Detection
- High Low Detection code
- Multi-color Volume At Price (VAP)
- Nadaraya-Watson Envelope
- Renko Chart
- Support and resistance
- TAPE READING
- Updated Renko Chart
- White Theme

More information:

See updated/extended version on-line.

Flip
-

Trading system toolbox
(AmiBroker 3.50)

SYNTAX	**flip(ARRAY1, ARRAY2)**
RETURNS	ARRAY
FUNCTION	works as a flip/flop device or "latch" (electronic/electric engineers will know what I mean) returns 1 from the first occurence of "true" signal in Array1 until a "true" occurs in Array2 which resets the state back to zero unil next "true" is detected in Array1...
EXAMPLE	buy = ExRem(buy, sell); buy = **Flip**(buy, sell); // this essentially reverts the process of ExRem - multiple signals are back again
SEE ALSO	

References:

The **Flip** function is used in the following formulas in AFL on-line library:

- Alphatrend
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Caleb Lawrence
- Candle Stick Demo
- Channel/S&R and trendlines
- Connors TPS
- Darvas Wisestocktrader
- Fre
- Halftrend
- Heatmap V1
- MACD BB Indicator
- Multiple Ribbon Demo
- Open Range Breakout Trading System
- Range Filter - Trading Strategy
- Renko Chart
- RUTVOL timing signal with BB Scoring routine
- Stops Implementation in AFS
- TD Sequential
- Updated Renko Chart

More information:

See updated/extended version on-line.

floor
- floor value

Math functions

SYNTAX	**floor(NUMBER)**
	floor(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Calculates the highest integer that is less than NUMBER or ARRAY.
EXAMPLE	The function `floor( 18.9 )` returns 18. The formula `floor( -19.9 )` returns -20.
SEE ALSO	CEIL() function

References:

The **floor** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Arnaud Legoux Moving Average (ALMA)
- AR_Prediction.afl
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Color Display.afl
- elliott wave manual labelling
- Gann level plotter
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- Graphical sector analysis
- Improved NH-NH scan / indicator
- JEEVAN'S SRI CHAKRA
- Luna Phase
- Murrey Math Price Lines
- Option Calls, Puts and days till third friday.
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- PF Chart - Close - April 2004
- Point & figure Chart India Securities
- Renko Chart
- Renko Chart
- Revised Renko chart
- tomy_frenchy
- Triangle exploration using P&F Chart
- Trigonometric Fit - TrigFit with AR for cos / sin
- Updated Renko Chart
- Visi-Trade
- Volume Charts

More information:

See updated/extended version on-line.

fmkdir
- creates (makes) a directory

File Input/Output functions
(AmiBroker 4.70)

SYNTAX **fmkdir("dirname")**

RETURNS NUMBER

FUNCTION Creates (makes) a directory.

"dirname" specifies path of the directory to be created. Please note that this function creates only ONE directory at a time. So if you want to create nested directory tree you have to call it multiple times, for example to create C:MyDirectoryMySubDirectory folder you have to call it twice:

```
fmkdir( "C:\\MyDirectory" );
fmkdir( "C:\\MyDirectory\\MySubDirectory" );
```

Note also that it is safe to call it even if directory already exists (then no change to file system is applied)

Returns TRUE if directory successfully created, FALSE otherwise

EXAMPLE

SEE ALSO

References:

The **fmkdir** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- Advanced Search and Find
- Backup Data of 1min Interval
- Export EOD or Intraday to .csv file
- IBD relative strength database ranker

More information:

See updated/extended version on-line.

fopen - open a file

File Input/Output functions
(AmiBroker 4.50)

SYNTAX **fopen("filename", "mode", shared = False)**

RETURNS FILE handle

FUNCTION Opens file, returns filehandle (NUMBER).

File handle is non-zero if file opened successfully, zero on failure.

Parameters:

- *filename* - STRING - contains the path to the file name. Please note that single backslash in path must be written in AFL as (double backslash)
- *mode* - STRING - access mode can be "r" - for reading, "w" for writing, "a" for appending (and all other regular C-runtime library modes)
- *shared* (new in 5.80) - BOOLEAN - False - open file without checking for sharing, True - open file in share-aware mode.

When sharing parameter is True the following things happen:

- when mode is "w" (writing) or "a" (appending) then file is open with a sharing flag that denies others to read and/or write at the same time
- when mode is "r" (reading) then file is open with a sharing flag that denies others to write to the file (shared reads are allowed)
- when open is unsuccessful due to sharing violation (other processes / threads have that file open and deny reads/writes) then fopen would wait for 250ms and retry automatically 4 times. After 4 unsuccessful retries it fails with file handle == Null.

When sharing parameter is set to False (default) no such checking occurs and file is open anyway without denying others to read/write. This may cause data corruption if file is written to from multiple threads/external processes at the same time. If you want to use this mode (sharing set to False) in multithreaded environment to write data, you need to care about synchronization yourself for example using critical section.

EXAMPLE

```
fh = fopen( "myfile.txt", "w");
if( fh )
{
   fputs( "Testing", fh );
}
else
{
   printf("Error opening file");
}
```

SEE ALSO fclose() function , fputs() function , fgets() function

References:

The **fopen** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System

- Advanced Search and Find
- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrade using an Exploration
- Backup Data of 1min Interval
- Calculate composites for tickers in list files
- Continuous Contract Rollover
- Create a list of functions in your program
- elliott wave manual labelling
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- Extract specific lines of code from your program
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Herman
- IBD relative strength database ranker
- IBD relative strength database Viewer
- MFE and MAE and plot trades as indicator
- Say Notes
- TWS auto-export Executions-file parser
- TWS trade plotter
- WLBuildProcess

More information:

See updated/extended version on-line.

Foreign - access foreign security data

Referencing other symbol data
(AmiBroker 3.50)

SYNTAX **foreign(TICKER, DATAFIELD, fixup = 1)**

RETURNS ARRAY

FUNCTION Allows referencing other (than current) tickers in the AFL formulas. TICKER is a string that holds the symbol of the stock. DATAFIELD defines which array is referenced. Allowable data fields: "O" (open), "H" (high), "L" (low), "C" (close), "V" (volume), "I" (open Interest), and for v5.29 and above: "1" (aux1), "2" (aux2)
The last parameter - fixup - accepts following values

- 0 - the holes are not fixed
- 1 - **default value** - missing data bar OHLC fields are all filled using previous bar Close and volume is set to zero.

  Note: you can use Foreign/RelStrength without specifying last parameter: Foreign("ticker", "field"), RelStrength("ticker") - then the holes will be fixed.
- 2 - (old pre-4.90 behaviour) - causes filling the holes in the data with previous O, H, L, C, V values

Unless you know what you are doing you should use DEFAULT value of fixup parameter (Fixup=1). If you do not use fixup=1, data holes will have the value of Null that you would need to handle by yourself.

EXAMPLE

```
// EXAMPLE 1:
// Plotting spread between currently selected symbol and another one
Graph0 = Close - Foreign( "MSFT", "Close" ) ;



// EXAMPLE 2:
// Built-in relative performance chart
_N( TickerList = ParamStr("Tickers", "^DJI,MSFT,GE") );
NumBars = 20;
fvb = Status("firstvisiblebar");
Plot( 100 * ( C - C[ fvb ] ) / C[ fvb ], Name(), colorBlue );
for( i = 0; ( symbol = StrExtract( TickerList, i ) ) != ""; i++ )
{
  fc = Foreign( symbol, "C" );

  if( ! IsNull( fc[ 0 ] ) )
  {
     Plot( 100 * ( fc - fc[ fvb ] )/ fc[ fvb ],
          symbol,
           colorLightOrange + ( (2*i) % 15 ),
          styleLine );
  }
}
PlotGrid( 0, colorYellow );
```

```
_N( Title = "{{NAME}} - Relative Performance [%]: {{VALUES}}" );
```

SEE ALSO PLOTFOREIGN() function , SetForeign() function

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2003-08-07 20:28:41	Foreign function synchronizes the data file you are referencing with the currently selected symbol. Synchronization makes sure that EACH bar of FOREIGN data matches exactly with each bar of currently selected symbol. So if DateNum() function returns 990503 for given bar then Close array represents the CLOSE of currently selected symbol at May 3, 1999 and Foreign("SYMBOL", "C") represents close of foreign symbol at May 3, 1999 TOO. This is absolutely necessary because otherwise you won't be able to do ANY meaningful operations involving both selected symbol and foreign symbol. This also needed for the display so when you mark the quote with vertical line it will always match the date displayed regardless if you use Foreign or not. Please note that if you have data holes in currently selected symbol then in order to synchronize bars Foreign function will remove bars that exist in Foreign symbol but do not exist in currently selected symbol.

References:

The **Foreign** function is used in the following formulas in AFL on-line library:

- 30 Week Hi Indicator - Display
- 4% Model - Determine Stock Market Direction
- 52 Week New High-New Low Index
- AccuTrack
- Alpha and Beta and R_Squared Indicator
- Auto-Optimization Framework
- Bad Tick Trim on 5 sec database
- Baseline Relative Performance Watchlist charts V2
- Basket Trading System T101
- BEANS-Summary of Holdings
- BMTRIX Intermediate Term Market Trend Indicator
- Bullish Percent Index 2004
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- COT REPORT
- Dave Landry PullBack Scan
- Detailed Equity Curve
- Elder Triple Screen Trading System
- Graphical sector analysis
- Graphical sector stock amalysis
- Heatmap V1
- How to add IB Option Symbols
- IB Backfiller

- IBD relative strength database ranker
- Improved NH-NH scan / indicator
- Index of 30 Wk Highs Vs Lows
- Indicator Explorer (ZigZag)
- Inter-market Yield Linear Regression Divergence
- Intraday Fibonacii Trend Break System
- Market Breadth Chart-In-Chart
- MOCS
- NASDAQ 100 Volume
- Peter Cooper
- qp2 industry charts as a panel in the stocks chart
- Ranking and sorting stocks
- Ranking Ticker WatchList
- Relative Strength Multichart of up to 10 tickers
- Rene Rijnaars
- RUTVOL timing signal with BB Scoring routine
- Sector Tracking
- SectorRSI
- Stress with SuperSmoother
- The D_oscillator
- Tracking Error
- Tracking Error
- Weighted Index
- WLBuildProcess

More information:

See updated/extended version on-line.

fputs
- write a string to a file

File Input/Output functions
(AmiBroker 4.50)

SYNTAX **fputs(** ***string***, ***filehandle*** **)**

RETURNS NOTHING

FUNCTION Writes (puts) the *string* to the file.

The *filehandle* must be a number returned by **fopen** function used to open the file. The file has to be open for writing or appending ("w" or "a") for this fputs to work.

EXAMPLE

```
//
// The following code exports quotes
// of current stock to quotes.csv
// comma separated file
//

fh = fopen( "quotes.csv", "w");
if( fh )
{
   fputs( "Date,Open,High,Low,Close,Volume\n", fh );

   y = Year();
   m = Month();
   d = Day();

   for( i = 0; i < BarCount; i++ )
   {
      ds = StrFormat("%02.0f-%02.0f-%02.0f,",
                     y[ i ], m[ i ], d[ i ] );
      fputs( ds, fh );

      qs = StrFormat("%.4f, %.4f, %.4f, %.4f, %.0f\n",
                     O[ i ], H[ i ], L[ i ], C[ i ], V[ i ] );
      fputs( qs, fh );
   }

   fclose( fh );
}
```

SEE ALSO fopen() function , fclose() function , fgets() function

References:

The **fputs** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- Advanced Search and Find
- AFL_Glossary_Converter
- Auto Trade Step by Step

- AutoTrade using an Exploration
- Backup Data of 1min Interval
- Continuous Contract Rollover
- elliott wave manual labelling
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Herman
- IBD relative strength database ranker
- TWS auto-export Executions-file parser

More information:

See updated/extended version on-line.

frac
- fractional part

Math functions

SYNTAX	**frac(NUMBER)**
	frac(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Eliminates the integer portion of NUMBER or ARRAY and returns the fractional part.
EXAMPLE	The formula frac(12.4) returns 0.4; the formula frac(-15.7) returns -0.7.
SEE ALSO	

References:

The **frac** function is used in the following formulas in AFL on-line library:

- Auto Trade Step by Step
- Bullish Percent Index 2 files combined
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- Continuous Contract Rollover
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Kagi Chart
- Lunar Phases - original
- LunarPhase
- Steve Woods' Cumulative Vol. Percentage Indicator
- Triangle exploration using P&F Chart
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

frmdir
- removes a directory

File Input/Output functions
(AmiBroker 4.70)

SYNTAX **frmdir("dirname")**

RETURNS NUMBER

FUNCTION This function removes a directory

"dirname" specifies path of the directory to be removed. Please note that this function removes only ONE directory at a time. So if you want to remove nested directory tree you have to call it multiple times, for example:

```
frmdir( "C:\\MyDirectory\\MySubDirectory" ); // delete nested subdir
first
frmdir( "C:\\MyDirectory" );
```

Note that directory must be empty before removing it otherwise it will not be possible to remove it.

Returns TRUE if directory successfully removed, FALSE otherwise

EXAMPLE

SEE ALSO

References:

The **frmdir** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

FullName - full name of the symbol

Information / Categories
(AmiBroker 3.10)

SYNTAX	**FullName()**
RETURNS	STRING
FUNCTION	The function returns stock full name which is definable by the user in **Symbol | Information** window.
EXAMPLE	printf(fullname());
SEE ALSO	

References:

The **FullName** function is used in the following formulas in AFL on-line library:

- Advanced Search and Find
- Advanced Trend Lines with S & R
- AFL Example
- AFL Example - Enhanced
- babaloo chapora
- colored CCI
- Commodity Channel Index
- Computing Cointegration and ADF Dashboard
- Darvas Amibroker
- Dave Landry PullBack Scan
- Dinapoli Guru Commentary
- Double Smoothed Stochastic from W.Bressert
- DPO with shading
- Elder Triple Screen Trading System
- FastStochK FullStochK-D
- Fibonacci Calculations & Speed Resistance
- Fre
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- hassan
- Improved NH-NH scan / indicator
- MACD commentary
- MACD Histogram - Change in Direction
- Market Facilitation Index VS Volume
- MS Darvas Box with Exploration
- Nadaraya-Watson Envelope
- Nick
- PF Chart - Close - April 2004
- Pivot End Of Day Trading System
- Pivot Finder
- Point & figure Chart India Securities
- prakash
- Relative Strength
- Scan New High and New Low

- shailu lunia
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Steve Woods' Float Channel Lines
- Stochastic Fast%K and Full
- TD Sequential
- Triangular Moving Average new
- Volume Divided Histogram
- VolumeDivAvgV3_Study_BrS
- William's Alligator System II
- Woodies CCI
- ZigZag Retracements

More information:

See updated/extended version on-line.

GapDown
- gap down

Basic price pattern detection

SYNTAX	**GapDown()**
RETURNS	ARRAY
FUNCTION	Gives a "1" or "true" on the day a security's prices gap down. Otherwise the result is "0". A gap down occurs if yesterday's low is greater than today's high.
EXAMPLE	
SEE ALSO	

References:

The **GapDown** function is used in the following formulas in AFL on-line library:

- Candle Identification
- Candle Pattern Function
- Candle Stick Analysis
- Candlestick Commentary
- Candlestick Commentary Modified
- Candlestick Commentary-modified

More information:

See updated/extended version on-line.

GapUp - gap up

Basic price pattern detection

SYNTAX	**GapUp()**
RETURNS	ARRAY
FUNCTION	Gives a "1" or "true" on the day a security's prices gap up. Otherwise the result is "0". A gap up occurs if yesterday's high is less than today's low.
EXAMPLE	
SEE ALSO	

References:

The **GapUp** function is used in the following formulas in AFL on-line library:

- Candle Identification
- Candle Pattern Function
- Candle Stick Analysis
- Candlestick Commentary
- Candlestick Commentary Modified
- Candlestick Commentary-modified
- swing chart

More information:

See updated/extended version on-line.

GetAsyncKeyState
- query the current state of keyboard keys

Miscellaneous functions
(AmiBroker 5.60)

SYNTAX **GetAsyncKeyState(vkey)**

RETURNS NUMBER

FUNCTION The function queries current state of keyboard keys (at the time of the call).

GetAsyncKeyState is 100% equivalent of Windows API function of the same name.

See: http://msdn.microsoft.com/en-us/library/windows/desktop/ms646293(v=vs.85).aspx

Arguments:
vkey is virtual key code to query (see table below).

The function returns 0 if key is NOT pressed and value < 0 (less than zero) when key is currently pressed.

Virtual Key codes:

```
vk_BackSpace = 8;
vk_Tab = 9;
vk_Return = 13;
vk_Shift = 16;
vk_Control = 17;
vk_Alt = 18;
vk_Pause = 19;
vk_CapsLock = 20;
vk_Escape = 27;
vk_Space = 32;
vk_PageUp = 33;
vk_PageDown = 34;
vk_End = 35;
vk_Home = 36;
vk_Left = 37;
vk_Up = 38;
vk_Right = 39;
vk_Down = 40;
vk_PrintScreen = 44;
vk_Insert = 45;
vk_Delete = 46;

/* NOTE: vk_0..vk_9 vk_A.. vk_Z match regular ASCII codes for digits and A-Z letters */

vk_0 = 48;
vk_1 = 49;
vk_2 = 50;
vk_3 = 51;
vk_4 = 52;
vk_5 = 53;
```

```
vk_6 = 54;
vk_7 = 55;
vk_8 = 56;
vk_9 = 57;
vk_A = 65;
vk_B = 66;
vk_C = 67;
vk_D = 68;
vk_E = 69;
vk_F = 70;
vk_G = 71;
vk_H = 72;
vk_I = 73;
vk_J = 74;
vk_K = 75;
vk_L = 76;
vk_M = 77;
vk_N = 78;
vk_O = 79;
vk_P = 80;
vk_Q = 81;
vk_R = 82;
vk_S = 83;
vk_T = 84;
vk_U = 85;
vk_V = 86;
vk_W = 87;
vk_X = 88;
vk_Y = 89;
vk_Z = 90;

vk_LWin = 91;
vk_RWin = 92;
vk_Apps = 93;
/* numerical key pad */
vk_NumPad0 = 96;
vk_NumPad1 = 97;
vk_NumPad2 = 98;
vk_NumPad3 = 99;
vk_NumPad4 = 100;
vk_NumPad5 = 101;
vk_NumPad6 = 102;
vk_NumPad7 = 103;
vk_NumPad8 = 104;
vk_NumPad9 = 105;
vk_Multiply = 106;
vk_Add = 107;
vk_Subtract = 109;
vk_Decimal = 110;
vk_Divide = 111;
/* function keys */
```

vk_F1 = 112;
vk_F2 = 113;
vk_F3 = 114;
vk_F4 = 115;
vk_F5 = 116;
vk_F6 = 117;
vk_F7 = 118;
vk_F8 = 119;
vk_F9 = 120;
vk_F10 = 121;
vk_F11 = 122;
vk_F12 = 123;
vk_F13 = 124;
vk_F14 = 125;
vk_F15 = 126;
vk_F16 = 127;

vk_NumLock = 144;
vk_ScrollLock = 145;
vk_LShift = 160;
vk_RShift = 161;
vk_LControl = 162;
vk_RControl = 163;
vk_LAlt = 164;
vk_RAlt = 165;
vk_SemiColon = 186;
vk_Equals = 187;
vk_Comma = 188;
vk_UnderScore = 189;
vk_Period = 190;
vk_Slash = 191;
vk_BackSlash = 220;
vk_RightBrace = 221;
vk_LeftBrace = 219;
vk_Apostrophe = 222;

EXAMPLE

```
Plot( C, "Close", colorRed );
vk_Shift = 16;

if( GetAsyncKeyState( vk_Shift ) < 0 ) Title = "Shift is pressed";
```

SEE ALSO

References:

The **GetAsyncKeyState** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling

More information:

See updated/extended version on-line.

GetBacktesterObject - get the access to backtester object

Trading system toolbox
(AmiBroker 4.60)

SYNTAX **GetBacktesterObject()**

RETURNS OBJECT

FUNCTION This funciton is used in custom backtester procedures to get the access to backtester object. Note that GetBacktester method should only be called when Status("action") returns actionPortfolio.

For more details please read Custom Backtester documentation

EXAMPLE

```
if( Status("action")== actionPortfolio )
{
    // retrieve the interface to portfolio backtester
     bo = GetBacktesterObject();

     //...here is your custom backtest formula.
}
```

SEE ALSO STATUS() function

References:

The **GetBacktesterObject** function is used in the following formulas in AFL on-line library:

- Customised Avg. Profit %, Avg. Loss % etc
- Detailed Equity Curve
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- MFE and MAE and plot trades as indicator
- Periodically ReBalance a BUY & HOLD Portfolio
- Rebalancing Backtest avoiding leverage

More information:

See updated/extended version on-line.

GetBaseIndex - retrieves symbol of relative strength base index

Referencing other symbol data
(AmiBroker 4.10)

SYNTAX	**GetBaseIndex( )**
RETURNS	STRING
FUNCTION	Retrieves base relative-strength index for given security as defined in Symbol->Categories.
EXAMPLE	AddTextColumn(GetBaseIndex(), "Base index");
SEE ALSO	

References:

The **GetBaseIndex** function is used in the following formulas in AFL on-line library:

- Dave Landry PullBack Scan
- Elder Triple Screen Trading System
- Indicator Explorer (ZigZag)

More information:

See updated/extended version on-line.

GetCategorySymbols - retrieves comma-separated list of symbols belonging to given category

Information / Categories
(AmiBroker 4.40)

SYNTAX	**GetCategorySymbols(category, index)**
RETURNS	STRING
FUNCTION	IMPORTANT: This function is now available under new name of **CategoryGetSymbols**. The old name is left only for backward compatibility. Please use new name in all new codes.
EXAMPLE	
SEE ALSO	CategoryGetSymbols() function

References:

The **GetCategorySymbols** function is used in the following formulas in AFL on-line library:

- Baseline Relative Performance Watchlist charts V2
- Count Tickers in Watchlist
- Ranking and sorting stocks
- Ranking Ticker WatchList
- WLBuildProcess

More information:

See updated/extended version on-line.

GetChartBkColor - get the RGB color value of chart background

Indicators
(AmiBroker 5.20)

SYNTAX **GetChartBkColor()**

RETURNS NUMBER

FUNCTION Returns color value of chart background. Color value in AmiBroker is either one of predefined color constants (colorWhite, colorBlack), or RGB value with offset of 56 (number of predefined colors). So to get actual RGB value you need to subtract 56 from the result of that function.

EXAMPLE

```
SetChartBkColor( ParamColor("Color", ColorRGB( 255, 255, 255 ) ) );

rgb = GetChartBkColor() - 56;

red = ( rgb & 255 );
green = floor( (rgb/256) & 255 );
blue = floor( rgb/(256*256) );

Title="R="+ red + " G=" + green + " B=" + blue;
```

SEE ALSO SetChartBkColor() function , SetChartBkGradientFill() function

References:

The **GetChartBkColor** function is used in the following formulas in AFL on-line library:

- B-Xtrender
- MACD BB Indicator
- Stress with SuperSmoother
- White Theme

More information:

See updated/extended version on-line.

GetChartID - get current chart ID

Exploration / Indicators
(AmiBroker 4.30)

SYNTAX	**GetChartID()**
RETURNS	NUMBER
FUNCTION	returns the chart ID of current indicator formula. Returns 0 if used in Automatic analysis.
EXAMPLE	Cross(graph0, Study("RE", GetChartID()));
SEE ALSO	

References:

The **GetChartID** function is used in the following formulas in AFL on-line library:

- AllinOneAlerts - Module
- channel indicator
- Continuous Contract Rollover
- For Auto Trading Setup
- Gfx Toolkit
- GFX ToolTip
- Heatmap V1
- Least Squares Channel Indicator
- MFE and MAE and plot trades as indicator
- Now Send Push Notifications From Amibroker
- Plot visual stop / target ratio.

More information:

See updated/extended version on-line.

GetCursorMouseButtons - get current state of mouse buttons

Indicators
(AmiBroker 4.80)

SYNTAX **GetCursorMouseButtons()**

RETURNS NUMBER

FUNCTION This function returns mouse button state at the time when chart formula is executed. The state is COMBINATION (bitwise OR) of the following flags:

- 0 - if no mouse button is down
- 1 - if left mouse button is down
- 2 - if right mouse button is down
- 4 - if middle mouse button is down
- 8 - if window just received mouse click (version 5.06 and higher only)

So for example the following combinations are possible:

- 3 - left + right down
- 3 - left + right down
- 5 - left + middle down
- 6 - right + middle down
- 7 - left + right + middle down
- 9 - left mouse button just clicked (v.5.06)
- 10 - right mouse button just clicked (v.5.06)
- 11 - left + right mouse button just clicked (v.5.06)

For example if you click the left mouse button and hold it down, the underlying chart will be refreshed and GetCursorMouseButtons will return 9 (8+1) during very first execution since click and 1 in all subsequent executions as long as left mouse button is kept pressed down. You need to use bitwise AND (&) to extract these flags as shown in the example.

EXAMPLE

```
if( GetCursorMouseButtons() & 1 )
{
printf("left mouse button is pressed down" );
}
```

Example 2. Low-level graphic + Interactive GUI control sample

```
/////////////////////////////////////////////////
// Low-level graphic + Interactive GUI control sample
// This example shows:
//  1. how to draw "buttons"
//  2. how to handle mouse clicks
//  3. how to implement event call-backs
///////////////////////////////////////////////////

Version( 5.04 ); // requires 5.04 or higher

////////////////////////////////////////////////////
// Part 1: DRAWING TABLE OF BUTTONS
```

```
//////////////////////////////////////////////////
GfxSetOverlayMode( 2 );
// formatted text output sample via low-level gfx functions

CellHeight = 20;
CellWidth = 100;
GfxSelectFont( "Tahoma", CellHeight/2 );

GfxSetBkMode( 1 );

function PrintInCell( string, row, Col )
{
GfxDrawText( string, Col * CellWidth, row * CellHeight, (Col + 1 ) *
CellWidth, (row + 1 ) * CellHeight, 0 );
}

GfxSelectPen( colorBlue );
for( i = 0; i < 10 && i < BarCount; i++ )
{
for( k = 0; k < 5; k++ )
{
   PrintInCell( "Button " + i + "," + k, i, k );
}
GfxMoveTo( 0, i * CellHeight );
GfxLineTo( 5 * CellWidth, i * CellHeight );
}
GfxMoveTo( 0, i * CellHeight );
GfxLineTo( 5 * CellWidth, i * CellHeight );

for( Col = 1; Col < 6; Col++ )
{
GfxMoveTo( Col * CellWidth, 0);
GfxLineTo( Col * CellWidth, 10 * CellHeight );
}


/////////////////////////////////////////////////////////
// Part 2: MOUSE BUTTON CALL BACKS
//////////////////////////////////////////////////////////
Title="";

function DrawButton( px, py, Clr1, Clr2, text )
{
     Col = floor( px / CellWidth );
     Row = floor( py / CellHeight );


   GfxGradientRect( Col * CellWidth, row * CellHeight, (Col + 1 ) *
CellWidth, (row + 1 ) * CellHeight,
                   Clr1, Clr2 );
     PrintInCell( text + " " + row + "," + Col, row, Col );

}

function OnLMouseButton(x, y, px, py)
{
   _TRACE("LButton x = " + DateTimeToStr( x ) + " y = " + y );

   DrawButton( px, py, ColorHSB( 50, 255, 255 ), ColorHSB( 90, 255,
255 ), "just clicked" );
}

function OnRMouseButton(x, y, px, py)
{
   _TRACE("RButton x = " + DateTimeToStr( x ) + " y = " + y );
}

function OnMMouseButton(x, y, px, py)
{
   _TRACE("MButton x = " + DateTimeToStr( x ) + " y = " + y );
}

function OnHoverMouse(x, y, px, py)
{
   _TRACE("LButton x = " + DateTimeToStr( x ) + " y = " + y );

     DrawButton( px, py, ColorRGB( 230, 230, 230 ), ColorRGB( 255,
255, 255 ), "mouse over" );
}

function OnLButtonIsDown(x, y, px, py)
{
   _TRACE("LButton x = " + DateTimeToStr( x ) + " y = " + y );

     DrawButton( px, py, ColorHSB( 190, 255, 255 ), ColorHSB( 210,
255, 255 ), "down" );
}

/////////////////////////////////////////////////////////
// Part 3: GENERAL PURPOSE EVENT HANDLER (reusable! - may be put
into "include" file)
////////////////////////////////////////////////////////

function EventHandler()
{
local b, x, y, px, py;
b = GetCursorMouseButtons();

// retrieve co-ordinates in date/value units
x = GetCursorXPosition(0);
y = GetCursorYPosition(0);
```

```
// retrieve co-ordinates in pixel units
px = GetCursorXPosition(1);
py = GetCursorYPosition(1);

if( b & 8 ) // flag = 8 is set when window just received mouse click
{
   // not-null means clicked in THIS (current) window
   if( b & 1 ) OnLMouseButton( x, y, px, py );
   if( b & 2 ) OnRMouseButton( x, y, px, py );
   if( b & 4 ) OnMMouseButton( x, y, px, py );
}
else
{
  if( b == 0 ) OnHoverMouse( x, y, px, py ); // no button pressed
  if( b == 1 ) OnLButtonIsDown( x, y, px, py ); // button pressed
}
}

EventHandler();
RequestTimedRefresh( 1 );
```

SEE ALSO GetCursorXPosition() function , GetCursorYPosition() function

References:

The **GetCursorMouseButtons** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- Gfx Toolkit
- GFX ToolTip
- Heatmap V1
- Visi-Trade

More information:

See updated/extended version on-line.

GetCursorXPosition - get current X position of mouse pointer

Indicators
(AmiBroker 4.80)

SYNTAX **GetCursorXPosition(mode = 0)**

RETURNS NUMBER (datetime)

FUNCTION Retrieves current mouse pointer X co-ordinate (i.e. datetime of bar under the mouse pointer). The functions now (v5.10 and up) by default return NULL (empty value) if mouse is OUTSIDE the current window

Parameters:

- mode = -1 - (old compatibility mode) - x - value gives X-coordinate in DateTime format. y - value gives PRICE. Values are reported no matter where is the mouse (i.e. may refer to window different than current if mouse is outside current window).
- mode = 0 - (default) x - value gives X-coordinate in DateTime format. y - value gives PRICE. Returns NULL if mouse is outside current window
- mode = 1 - x, y - are mouse coordinates expressed in screen PIXELS. Returns NULL if mouse is outside current window

Values returned are equal to those visible in the status bar, and these functions require status bar to be visible. Returned values represent cursor position at the formula execution time (or few milliseconds before it) and accurracy is subject to pixel resolution of the screen (first cursor position is read in screen pixels (integer) and then converted to actual value therefore for example when screen resolution is 1024x768 maximum obtainable resolution in X direction is 0.1% and in Y direction 0.13%), also X values are snap to datetime of nearest data bar.

It only makes sense to use these functions in indicator/interpretation code.

Using them in AA window may yield random values.GetCursorXPosition() function returns X position in DateTime format (the same as used by DateTime() function). You can convert it to string using DateTimeToStr() function.GetCursorYPosition() returns Y position (as displayed in Y axis of the chart).

EXAMPLE

```
ToolTip = "X=" + DateTimeToStr( GetCursorXPosition() ) +
          "nY=" + GetCursorYPosition();
```

SEE ALSO GetCursorYPosition() function , GetCursorMouseButtons() function

References:

The **GetCursorXPosition** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- Gfx Toolkit
- GFX ToolTip
- Heatmap V1
- Visi-Trade
- Visualization of stoploses and profit in chart

More information:

See updated/extended version on-line.

GetCursorYPosition - get current Y position of mouse pointer

Indicators
(AmiBroker 4.80)

SYNTAX **GetCursorYPosition()**

RETURNS NUMBER

FUNCTION Retrieves current mouse pointer Y co-ordinate (i.e. value in dollars or other Y-axis unit). The functions now (v5.10 and up) by default return NULL (empty value) if mouse is OUTSIDE the current window

Parameters:

- mode = -1 - (old compatibility mode) - x - value gives X-coordinate in DateTime format. y - value gives PRICE. Values are reported no matter where is the mouse (i.e. may refer to window different than current if mouse is outside current window).
- mode = 0 - (default) x - value gives X-coordinate in DateTime format. y - value gives PRICE. Returns NULL if mouse is outside current window
- mode = 1 - x, y - are mouse coordinates expressed in screen PIXELS. Returns NULL if mouse is outside current window

Values returned are equal to those visible in the status bar, and these functions require status bar to be visible. Returned values represent cursor position at the formula execution time (or few milliseconds before it) and accurracy is subject to pixel resolution of the screen (first cursor position is read in screen pixels (integer) and then converted to actual value therefore for example when screen resolution is 1024x768 maximum obtainable resolution in X direction is 0.1% and in Y direction 0.13%), also X values are snap to datetime of nearest data bar.

It only makes sense to use these functions in indicator/interpretation code.

Using them in AA window may yield random values.GetCursorXPosition() function returns X position in DateTime format (the same as used by DateTime() function). You can convert it to string using DateTimeToStr() function.GetCursorYPosition() returns Y position (as displayed in Y axis of the chart).

EXAMPLE
```
ToolTip = "X=" + DateTimeToStr( GetCursorXPosition() ) +
"nY=" + GetCursorYPosition();
```

SEE ALSO GetCursorXPosition() function , GetCursorMouseButtons() function

References:

The **GetCursorYPosition** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- Gfx Toolkit
- GFX ToolTip
- Heatmap V1
- Visi-Trade
- Visualization of stoploses and profit in chart

More information:

See updated/extended version on-line.

GetDatabaseName - retrieves folder name of current database

Information / Categories
(AmiBroker 4.30)

SYNTAX	**GetDatabaseName()**
RETURNS	STRING
FUNCTION	retrieves the name of the database - the last part (folder) of the database path
EXAMPLE	
SEE ALSO	

References:

The **GetDatabaseName** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GetExtraData
- get extra data from external data source

Miscellaneous functions
(AmiBroker 3.90)

SYNTAX **GetExtraData("fieldname")**

RETURNS NUMBER or ARRAY or STRING

FUNCTION Retrieves data-source specific data.
Currently only Quotes Plus and TC2000 plug-isn support this function.
The list of fields available via QP2 plug-in:

- "AnnDividend"
- "Shares"
- "SharesFloat"
- "IssueType" (string)
- "SharesOut"
- "SharesShort"
- "TTMsales"
- "Beta"
- "TTMEps"
- "HiPERange"
- "LoPERange"
- "PEG"
- "InstHolds"
- "LTDebtToEq"
- "CashFlowPerShare"
- "ROE"
- "TTMSales"
- "Yr1EPSGrowth"
- "Yr5EPSGrowth"
- "Yr1ProjEPSGrowth"
- "Yr2ProjEPSGrowth"
- "Yr3to5ProjEPSGrowth"
- "BookValuePerShare"
- "Briefing" (string)
- "QRS" (array)
- "HasOptions"
- "EPSRank" (array) - requires QP plugin 1.4.3
- "Sales" (array) - requires QP plugin 1.5.0
- "EPS" (array) - requires QP plugin 1.5.0
- "LastMainDate" - (number) date of last update of given symbol in YYYYMMDD format - requires QP plugin 1.5.1
- "Exchange" - (string) - exchange code - requires QP plugin 1.5.1
- "ExchangeSub" - (string) exchange sub-code - requires QP plugin 1.5.1
- "Flags" - (string) - requires QP plugin 1.5.1
- "MarginFlag" - (string)- requires QP plugin 1.5.1
- "CUSIP" - (string) - requires QP plugin 1.5.1
- "SIC" - (string) - requires QP plugin 1.5.1
- "IssueStatus" - (string) - with default settings all symbols should have issue status = 0 other possible values: 0 = actively trading; 1, P = trading on a when issued basis, 5, 6, 7, A, B, C, D, E, M = not trading 4, N = new symbol, G, K, X, R, Z = changes to

symbol, cusip, name, etc. - requires QP plugin 1.5.1

The list of fields available via TC2000 plug-in:

- "BOP" - balance of power indicator
- "MoneyStream" - money stream indicator

EXAMPLE

```
GetExtraData("briefing"); /* gives briefing text (STRING) */
graph0 = GetExtraData("QRS"); /*gives Quotes Plus relative strength (ARRAY) */
```

SEE ALSO

References:

The **GetExtraData** function is used in the following formulas in AFL on-line library:

- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Steve Woods' Float Channel Lines

More information:

See updated/extended version on-line.

GetExtraDataForeign - get extra data from external data source for specified symbol

Miscellaneous functions
(AmiBroker 6.20)

SYNTAX **GetExtraData("fieldname", "symbol")**

RETURNS

FUNCTION Retrieves data-source specific data for specified (non-current) symbol
Currently only Quotes Plus and TC2000 plug-isn support this function.
The list of fields available via QP2 plug-in:

- "AnnDividend"
- "Shares"
- "SharesFloat"
- "IssueType" (string)
- "SharesOut"
- "SharesShort"
- "TTMsales"
- "Beta"
- "TTMEps"
- "HiPERange"
- "LoPERange"
- "PEG"
- "InstHolds"
- "LTDebtToEq"
- "CashFlowPerShare"
- "ROE"
- "TTMSales"
- "Yr1EPSGrowth"
- "Yr5EPSGrowth"
- "Yr1ProjEPSGrowth"
- "Yr2ProjEPSGrowth"
- "Yr3to5ProjEPSGrowth"
- "BookValuePerShare"
- "Briefing" (string)
- "QRS" (array)
- "HasOptions"
- "EPSRank" (array) - requires QP plugin 1.4.3
- "Sales" (array) - requires QP plugin 1.5.0
- "EPS" (array) - requires QP plugin 1.5.0
- "LastMainDate" - (number) date of last update of given symbol in YYYYMMDD format - requires QP plugin 1.5.1
- "Exchange" - (string) - exchange code - requires QP plugin 1.5.1
- "ExchangeSub" - (string) exchange sub-code - requires QP plugin 1.5.1
- "Flags" - (string) - requires QP plugin 1.5.1
- "MarginFlag" - (string)- requires QP plugin 1.5.1
- "CUSIP" - (string) - requires QP plugin 1.5.1
- "SIC" - (string) - requires QP plugin 1.5.1
- "IssueStatus" - (string) - with default settings all symbols should have issue status = 0 other possible values: 0 = actively trading; 1, P = trading on a when issued basis, 5, 6, 7, A, B, C, D, E, M = not trading 4, N = new symbol, G, K, X, R, Z = changes to

symbol, cusip, name, etc. - requires QP plugin 1.5.1

The list of fields available via TC2000 plug-in:

- "BOP" - balance of power indicator
- "MoneyStream" - money stream indicator

EXAMPLE GetExtraDataForeign("briefing", "MSFT"); /* gives briefing text (STRING) */
graph0 = GetExtraDataForeign("QRS", "MSFT"); /*gives Quotes Plus relative strength (ARRAY) */

SEE ALSO GetExtraData() function

References:

The **GetExtraDataForeign** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GetFnData - get fundamental data

Information / Categories
(AmiBroker 4.90)

SYNTAX	**GetFnData("field")**
RETURNS	NUMBER or STRING
FUNCTION	GetFnData allows accessing fundamental data from Information window (Window->Symbol Information) "field" parameter can be one of the following:

- "EPS"
- "EPSEstCurrentYear"
- "EPSEstNextYear"
- "EPSEstNextQuarter"
- "PEGRatio"
- "SharesFloat"
- "SharesOut"
- "DividendPayDate"
- "ExDividendDate"
- "BookValuePerShare"
- "DividendPerShare"
- "ProfitMargin"
- "OperatingMargin"
- "OneYearTargetPrice"
- "ReturnOnAssets"
- "ReturnOnEquity"
- "QtrlyRevenueGrowth"
- "GrossProfitPerShare"
- "SalesPerShare"
- "EBITDAPerShare"
- "QtrlyEarningsGrowth"
- "InsiderHoldPercent"
- "InstitutionHoldPercent"
- "SharesShort"
- "SharesShortPrevMonth"
- "ForwardDividendPerShare"
- "ForwardEPS"
- "OperatingCashFlow"
- "LeveredFreeCashFlow"
- "Beta"
- "LastSplitRatio"
- "LastSplitDate"
- "Alias" (returns symbol alias - string) - 5.50 and above
- "Address" (returns symbol address - string) - 5.50 and above
- "Country" (returns symbol country - string) - 5.60 and above
- "DelistingDate" returns symbol delisting date (as datetime) - 5.70 and above
- "PointValue" - returns symbols point value
- "FullName" - returns full name of the symbol
- "Currency" - returns currency field - 5.70 and above
- "WebID" - returns web ID field - 5.70 and above

EXAMPLE

```
AddColumn( Close / GetFnData( "EPS" ) , "Current P/E ratio" );
AddColumn( Close / GetFnData( "EPSEstNextYear" ) , "Est. Next Year
P/E ratio" );
Filter = Status("lastbarinrange");
```

SEE ALSO GetRTData() function , GetRTDataForeign() function , GetFnDataForeign() function

References:

The **GetFnData** function is used in the following formulas in AFL on-line library:

- ICHIMOKU SIGNAL TRADER
- Price Chart - Fundamental
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

GetFnDataForeign - get fundamental data for specified symbol

Information / Categories
(AmiBroker 6.20)

SYNTAX **GetFnDataForeign("field", "symbol')**

RETURNS NUMBER or STRING

FUNCTION GetFnDataForeign allows accessing fundamental data from Information window (Window->Symbol Information) for specified (non-current) symbol "field" parameter can be one of the following:

- "EPS"
- "EPSEstCurrentYear"
- "EPSEstNextYear"
- "EPSEstNextQuarter"
- "PEGRatio"
- "SharesFloat"
- "SharesOut"
- "DividendPayDate"
- "ExDividendDate"
- "BookValuePerShare"
- "DividendPerShare"
- "ProfitMargin"
- "OperatingMargin"
- "OneYearTargetPrice"
- "ReturnOnAssets"
- "ReturnOnEquity"
- "QtrlyRevenueGrowth"
- "GrossProfitPerShare"
- "SalesPerShare"
- "EBITDAPerShare"
- "QtrlyEarningsGrowth"
- "InsiderHoldPercent"
- "InstitutionHoldPercent"
- "SharesShort"
- "SharesShortPrevMonth"
- "ForwardDividendPerShare"
- "ForwardEPS"
- "OperatingCashFlow"
- "LeveredFreeCashFlow"
- "Beta"
- "LastSplitRatio"
- "LastSplitDate"
- "Alias" (returns symbol alias - string) - 5.50 and above
- "Address" (returns symbol address - string) - 5.50 and above
- "Country" (returns symbol country - string) - 5.60 and above
- "DelistingDate" returns symbol delisting date (as datetime) - 5.70 and above
- "PointValue" - returns symbols point value
- "FullName" - returns full name of the symbol

EXAMPLE

```
AddColumn( Foreign(  "MSFT",  "C") / GetFnDataForeign( "EPS", "MSFT"
) , "MSFT Current P/E ratio" );
AddColumn( Foreign(  "MSFT",  "C") / GetFnDataForeign(
"EPSEstNextYear", "MSFT" ) ,  "MSFT Est. Next Year P/E ratio" );
Filter = Status("lastbarinrange");
```

SEE ALSO GetFnData() function

References:

The **GetFnDataForeign** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GetFormulaPath - get file path of current formula

Miscellaneous functions
(AmiBroker 5.90)

SYNTAX	**GetFormulaPath()**
RETURNS	STRING
FUNCTION	The function returns full file path of current formula
EXAMPLE	
SEE ALSO	

References:

The **GetFormulaPath** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- White Theme

More information:

See updated/extended version on-line.

GetLastOSError - get text of last operating system (Windows) error

Miscellaneous functions
(AmiBroker 6.30)

SYNTAX **GetLastOSError()**

RETURNS STRING

FUNCTION The function retrieves text of last OS (Windows) error message. It is useful for providing meaningful error reporting in case when you interact with OS (for example opening files or remote Internet sites). See examples.

EXAMPLE

```
// Example 1:

fh = fopen("non_existing_file.txt", "r" );

if( ! fh )
{
   printf("File can not be open because: %s", GetLastOSError() );
}

// Example 2:

ih = InternetOpenUrl("http://non_existing_host.com" );

if( ! ih )
{
   printf("Internet connection can not be open because: %s",
GetLastOSError() );
}
```

SEE ALSO fopen() function , fmkdir() function , frmdir() function , fdelete() function , InternetOpenURL() function

References:

The **GetLastOSError** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GetObject - get handle to already running OLE object

Miscellaneous functions
(AmiBroker 6.40)

SYNTAX **GetObject(path, class)**

RETURNS OBJECT

FUNCTION The function obtains a handle to already running OLE object, providing functionality equivalent to JScript GetObject and VBScript GetObject

EXAMPLE

```
Version(6.38); // minimum required version
Workbook = GetObject("e:\\Full_Path\\To_File\\Sheet.xls", "");
Excel = Workbook.Application;
Excel.Visible = True;
Worksheet = Excel.ActiveSheet;
Range = Worksheet.Range("A1", "A1");
Range.Value = 123;
```

SEE ALSO CreateObject() function , CreateStaticObject() function

References:

The **GetObject** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GetOption - gets the value of option in automatic analysis settings

Trading system toolbox
(AmiBroker 4.60)

SYNTAX **GetOption("fieldname")**

RETURNS NUMBER

FUNCTION Gets the value of various options in automatic analysis settings.

field - is a string that defines the option to read. There are following options available:

- "NoDefaultColumns" - if set to True - exploration does not have default Ticker and Date/Time columns
- "InitialEquity"
- "AllowSameBarExit"
- "ActivateStopsImmediately"
- "AllowPositionShrinking"
- "FuturesMode"
- "InterestRate"
- "ApplyTo" - returns Analysis "Apply To" setting 0 - all symbols, 1 - current symbol, 2 - filter
- "FilterIncludeIndex", "FilterIncludeFavorite", "FilterIncludeMarket", "FilterIncludeGroup", "FilterIncludeSector", "FilterIncludeIndustry", "FilterIncludeWatchlist" - return "Include" filter settings -1 - means NOT selected (not included), >= 0 index of included category
- "FilterExcludeIndex", "FilterExcludeFavorite", "FilterExcludeMarket", "FilterExcludeGroup", "FilterExcludeSector", "FilterExcludeIndustry", "FilterExcludeWatchlist" - return "Exclude" filter settings
- -1 - means NOT selected (not excluded), >= 0 index of excluded category "BuyDelay", "SellDelay", "ShortDelay", "CoverDelay" (new in 5.90) - retrieves trade delays
- "MaxOpenPositions" - maximum number of simlutaneously open positions (trades) in portfolio backtest/optimization
- "WorstRankHeld" - the worst rank of symbol to be held in rotational trading mode (see **EnableRotationalTrading** for more details)
- "MinShares" - the minimum number of shares required to open the position in the backtester/optimizer. If you don't have enough funds to purchase that many, trade will NOT be entered
- "MinPosValue" - (4.70.3 and above) the minimum dollar amount required to open the position in the backtester/optimizer. If you don't have enough funds trade will NOT be entered
- "PriceBoundChecking" - if set to False - disables checking and adjusting buyprice/sellprice/coverprice/shortprice arrays to current symbol High-Low range.
- CommissionMode -
  0 - use portfolio manager commission table
  1 - percent of trade
  2 - $ per trade
  3 - $ per share/contract
- CommissionAmount - amount of commission in modes 1..3
- AccountMargin (in old versios it was 'MarginRequirement') - account margin requirement (as in settings), 100 = no margin

- ReverseSignalForcesExit - reverse entry signal forces exit of existing trade (default = True)
- UsePrevBarEquityForPosSizing - Affects how percent of current equity position sizing is performed.
  False (default value) means: use current (intraday) equity to perform position sizing,
  True means: use previous bar closing equity to perform position sizing
- PortfolioReportMode - sets backtester report mode:
  0 - trade list
  1 - detailed log
  2 - summary
  3 - no output (custom only)
- UseCustomBacktestProc - True/False - allows to turn on/off custom backtest procedure
- EveryBarNullCheck - allows to turn on checking for Nulls in arithmetic operations on every bar in the array(by default it is OFF - i.e. AmiBroker checks for nulls that appear in the beginning of the arrayand in the end of the array and once non-null value is detected it assumes no further holes (nulls) in the middle). Turning "EveryBarNullCheck" to True allows to extend these checks to each and every barwhich is the way 4.74.x and earlier versions worked.
  Note however that turning it on gives huge performance penalty (arithmetic operations are performed even 4x slower when this option is ON, so don't use it unless you really have to).
- HoldMinBars - Number - if set to value > 0 - it disables exit during user-specified number of bars even if signals/stops are generated during that period
- EarlyExitBars - Number if set to value > 0 - causes that special early exit (redemption) fee is charged if trade is exited during this period
- EarlyExitFee - defines the % (percent) value of early exit fee
- HoldMinDays - Number - if set to value > 0 - it disables exit during user-specified number of CALENDAR DAYS (not bars) even if signals/stops are generated during that period
- EarlyExitDays - Number if set to value > 0 - causes that special early exit (redemption) fee is charged if trade is exited during the period specified in calendar days (not bars).
- DisableRuinStop - it set to TRUE built-in ruin stop is disabled
- Generate report - allows to suppress/force generation of backtest report. Allowable values: 0, 1, or 2
  By default backtest reports are generated ONLY for portfolio backtests and for individual backtests if individual reporting is turned on in the settings. Reports are disabled for optimization.
  Now with the SetOption() function you can either supress report generation for backtests or enable report generation during certain optimization steps, all from code level.
  SetOption("GenerateReport", 0); // suppress generation of report
  SetOption("GenerateReport", 1); // force generation of full report
  SetOption("GenerateReport", 2); // only one-line report is generated (in results.rlst file) viewable as single line in Report Explorer
- SeparateLongShortRank - True/False
  When separate long/short ranking is enabled, the backtester maintains TWO separate "top-ranked" signal lists, one for long signals and one for short signals. This ensures that long and short candidates are independently even if position score is not symetrical (for example when long candidates have very high positive scores

while short candidates have only fractional negative scores). That contrasts with the default mode where only absolute value of position score matters, therefore one side (long/short) may completely dominate ranking if score values are asymetrical.
When SeparateLongShortRank is enabled, in the second phase of backtest, two separate ranking lists are interleaved to form final signal list by first taking top ranked long, then top ranked short, then 2nd top ranked long, then 2nd top ranked short, then 3rd top ranked long and 3rd top ranked short, and so on... (as long as signals exist in BOTH long/short lists, if there is no more signals of given kind, then remaining signals from either long or short lists are appended)
For example: Entry signals(score):ESRX=Buy(60.93), GILD=Short(-47.56), CELG=Buy(57.68), MRVL=Short(-10.75), ADBE=Buy(34.75), VRTX=Buy(15.55), SIRI=Buy(2.79),
As you can see Short signals get interleaved between Long signals even though their absolute values of scores are smaller than corresponding scores of long signals. Also there were only 2 short signals for that particular bar so, the rest of the list shows long signals in order of position score
Although this feature can be used independently, it is intended to be used in combination with MaxOpenLong and MaxOpenShort options.

- MaxOpenLong - limits the number of LONG positions that can be open simultaneously
- MaxOpenShort - limits the number of SHORT positions that can be open simultaneously
  The value of ZERO (default) means NO LIMIT. If both MaxOpenLong and MaxOpenShort are set to zero (or not defined at all) the backtester works old way - there is only global limit active (MaxOpenPositions) regardless of type of trade.
  Note that these limits are independent from global limit (MaxOpenPositions). This means that MaxOpenLong + MaxOpenShort may or may not be equal to MaxOpenPositions.
  If MaxOpenLong + MaxOpenShort is greater than MaxOpenPositions then total number of positions allowed will not exceed MaxOpenPositions, and individual long/short limits will apply too. For example if your system MaxOpenLong is set to 7 and maxOpenShort is set to 7 and MaxOpenPositions is set to 10 and your system generated 20 signals: 9 long (highest ranked) and 11 short, it will open 7 long and 3 shorts.
  If MaxOpenLong + MaxOpenShort is smaller than MaxOpenPositions (but greater than zero), the system won't be able to open more than (MaxOpenLong+MaxOpenShort).
  Please also note that MaxOpenLong and MaxOpenShort only cap the number of open positions of given type (long/short). They do NOT affect the way ranking is made. I.e. by default ranking is performed using ABSOLUTE value of positionscore.
  If your position score is NOT symetrical, this may mean that you are not getting desired top-ranked signals from one side. Therefore, to fully utilise MaxOpenLong and MaxOpenShort in rotational balanced ("market neutral") long/short systems it is desired to perform SEPARATE ranking for long signals and short signals.
  To enable separate long/short ranking use:
  SetOption("SeparateLongShortRank", True);
- RefreshWhenCompleted - when set to TRUE, it will perform View->Refresh All after Automatic-Analysis operation (scan/exploration/backtest/optimize) is completed.
- RequireDeclarations - when set to TRUE the AFL engine will always require variable declarations (using local/global) on formula-by-formula basis

- ExtraColumnsLocation - allows the user to change the location of custom columns added during backtest/optimization.
  "extra" columns mean:
  a) any custom metrics added using custom backtester
  b) any optimization parameters defined using Optimize() function

  If both custom metrics and optimization parameters are present then custom metrics appear first then optimization parameters

  This function is provided to allow the user to change the default "at the end" location of custom columns/optimization parameters.
  For example:

  SetOption("ExtraColumnsLocation", 1);

  will cause that custom metrics and opt params will be subsequently added starting from column 1 (as opposed to last column default)

  Note that this setting changes "visual" order of columns, not really in-memory order or export order, so exported data files or copy/paste format do not change.
- SettlementDelay - this option describes the number of days (not bars) it takes for sale proceeds to settle and be available for opening new positions.

  SetOption("SettlementDelay", 3); // this will cause that proceeds from sale are only available for trading on 3rd day after sale

  For detailed tracking " Detailed log" report option now shows available and unsettled funds for T+1, T+2 and so on

  Note: when using this option it is recommended to use backtestRegularRaw instead of backtestRegular, otherwise some trades may not be entered because funds are not settled immediately and you need to be able to enter not on first but subsequent buy signals and that is exactly what backtestRegularRaw offers.

  Note2: old backtester (Equity() function) ignores settlement delay
- StaticVarAutoSave - allow periodical auto-saving of persistent static variables (in addition to saving on exit, which is always done).

  The interval is given in seconds.
  For example:
  SetOption("StaticVarAutoSave", 60); // auto-save persistent variables every 60 seconds (1-minute)
  It is important to understand that persistent variables are saved ON EXIT automatically, without any user intervention so it should be enough for most cases. If you for some reason want auto-saves when AmiBroker is running, then you can use this function. Please note that writing many static variables into physical disk file takes time and it blocks all static variable access so you should AVOID specifying too small auto-save intervals. Saving every second is bad idea - it will cause overload. Saving every 60 seconds should be fine. Calling function with interval set to zero disables auto-save.
  SetOption("StaticVarAutoSave", 0);

- MCEnable - controls Monte Carlo simulation: 0 - disabled, 1 - enabled in backtests, 2 - enabled in backtests and optimizations
- MCRuns - number of Monte Carlo simulation runs (realizations) default 1000
- MCPosSizeMethod - Monte Carlo position size method: 0 - don't change, 1 - fixed size, 2 - constant amount, 3 - percent of equity
- MCPosSizeShares - number of shares per trade in MC simulation
- MCPosSizeValue - dollar value per trade in MC simulation
- MCPosSizePctEquity - percent of current equity per trade in MC simulation
- MCChartEquityCurves - true/false (1/0) - enables Monte Carlo equity chart
- MCStrawBroomLines - defines number of equity lines drawn in Monte Carlo straw broom chart
- BHSymbol - (new in 6.90) - get Portfolio B&H symbol. Note that this is read-only (can't set it via SetOption)

EXAMPLE

```
InitialEquity = GetOption("InitialEquity");
```

SEE ALSO SetOption() function

References:

The **GetOption** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GetPerformanceCounter - retrieves the current value of the high-resolution performance counter

Miscellaneous functions
(AmiBroker 4.90)

SYNTAX **GetPerformanceCounter(bReset = False)**

RETURNS NUMBER

FUNCTION GetPerformanceCounter retrieves the current value of the high-resolution performance counter. Returned value is in milliseconds. Resolution is upto 0.001 ms (1 microsecond). The value of high-resolution counter represents number of milliseconds from either system start (boot) or from last counter reset. To reset the counter you need to call GetPerformanceCounter function with bReset parameter set to True.

Note that reseting counters inside one formula does not affect counters in other formulas. Since returned values are very large (time in milliseconds since system start is usually quite large), for precise measurements of single function or small function block execution times it is strongly recommended to reset counter at the beginning of the block so floating point resolution (7 digits) does not affect the precision of measurement.

GetPerformanceCounter function can be also used in trading system automation to measure time in milliseconds between various events (just subtract values returned by GetPerformanceCounter() during two different events)

Caveat: this function relies on Windows API QueryPerformanceCounter function and CPU RTDSC instruction and it may yield to inaccurrate results if you have multiple-core processor and AMD's "Cool and Quiet" enabled in BIOS or other CPU clock stepping technologies enabled. If this applies to you, you may check Microsoft hotfix to this problem at: http://support.microsoft.com/?id=896256

EXAMPLE

```
////////////////////////////////
// EXAMPLE 1
// The code shows that 1000 iterations of sin() calculation
// takes about 1.7 milliseconds.
// Note that call to the GetPerformanceCounter()
// has overhead of about 0.015 ms (15 microseconds)

GetPerformanceCounter(True); // reset counter to zero
for( i = 0; i < 1000; i++ )
{
   k = sin( i );
}

elapsed=GetPerformanceCounter();

"Time [ms] = "+elapsed;




////////////////////////////////
```

```
// EXAMPLE 2
// GetPerformanceCounter function
// may also be used to report time since system start.

elapsed=GetPerformanceCounter();

StrFormat("Time since system start %.0f days, %.0f hours, %.0f
minutes, %.0f seconds, %.0f milliseconds ",
floor(elapsed/(24*60*60*1000)),
floor( elapsed/(60*60*1000) ) % 24,
floor( elapsed/(60*1000) ) % 60,
floor( elapsed/1000 ) % 60,
elapsed % 1000 );
```

SEE ALSO

References:

The **GetPerformanceCounter** function is used in the following formulas in AFL on-line library:

- Animated BackGround
- Animated BackGround 1.1
- AutoTrade using an Exploration
- Heatmap V1

More information:

See updated/extended version on-line.

GetPlaybackDateTime - get bar replay position date/time

Date/Time
(AmiBroker 50)

SYNTAX **GetPlaybackDateTime()**

RETURNS NUMBER

FUNCTION The function returns bar replay playback position in datetime format, or zero if bar replay is NOT active

EXAMPLE

```
pt = GetPlaybackDateTime(); // new function to retrieve playback
position date/time,
//returns zero if bar replay is NOT active

if( pt )
{
  Title = "Playback time: " + DateTimeToStr( pt );
}
else
{
  Title = "Bar Replay not active";
}
```

SEE ALSO DateTime() function , DateTimeConvert() function , DateTimeToStr() function

References:

The **GetPlaybackDateTime** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GetPriceStyle - get current price chart style

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **GetPriceStyle**

RETURNS NUMBER

FUNCTION Returns price chart style value to be used in Plot statement Returned value depends on selection in View->Price chart style menu

EXAMPLE

```
Plot( C, "Close", ParamColor("Color", colorBlack ), styleNoTitle |
ParamStyle("Style") | GetPriceStyle() );
```

SEE ALSO

References:

The **GetPriceStyle** function is used in the following formulas in AFL on-line library:

- Advisory NRx price chart display.
- Alphatrend
- Average Price Crossover
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- changing period moving avarage
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- Coppock Trade Signal on Price Chart
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Dave Landry PullBack Scan
- DEBAJ
- DiNapolis 3x Displaced Moving Averages
- Elder Triple Screen Trading System
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fibonacci Moving averages
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Gann level plotter
- Guppy Cloud
- Halftrend
- HH-LL-PriceBar
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- interactively test discretionary trading
- Intraday Fibonacii Trend Break System
- Intraday Strength
- Intraday Trend Break System
- Last Five Trades Result Dashboard – AFL code
- LunarPhase
- MAVG
- MS Darvas Box with Exploration

- nifty
- Open Range Breakout Trading System
- pattenz
- Perceptron
- Pivot End Of Day Trading System
- plot tomorrows pivots on an intraday database
- prakash
- Price Chart - Fundamental
- Prior Daily OHLC
- PVT Trend Decider
- Range Filter - Trading Strategy
- Rebalancing Backtest avoiding leverage
- SAR-ForNextBarStop
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Super Trend Indicator
- suresh
- TrendingRibbonArrowsADX
- Vikram's Floor Pivot Intraday System
- Visible Min and Max Value Demo
- visual turtle trading system
- William's Alligator System II
- WILSON RELATIVE PRICE CHANNEL
- Zig-Hi Zap-Lo

More information:

See updated/extended version on-line.

GetRTData
- retrieves the real-time data fields

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX **GetRTData("fieldname")**

RETURNS NUMBER

FUNCTION Retrieves the LAST (the most recent) value of the following fields reported by streaming real time data source:

- "Ask" - current best ask price
- "AskSize " - current ask size
- "Bid" - current best bid price
- "BidSize " - current bid size
- "52WeekHigh" - 52 week high value
- "52WeekHighDate" - 52 week high date (in datenum format)
- "52WeekLow" - 52 week low value
- "52WeekLowDate" - 52 week low date (in datenum format)
- "Change" - change since yesterdays close
- "Dividend" - last dividend value
- "DivYield" - dividend yield
- "EPS" - earnings per share
- "High" - current day's high price
- "Low" - current day's low price
- "Open" - current day's open price
- "Last" - last trade price
- "OpenInt" - current open interest
- "Prev" - previous day close
- "TotalVolume" - total today's volume
- "TradeVolume" - last trade volume
- "ChangeDate" - datenum (YYYMMDD) of last data change
- "ChangeTime" - timenum (HHMMSS) of last data change
- "UpdateDate" - datenum (YYYMMDD) of last data update
- "UpdateTime" - timenum (HHMMSS) of last data update
- "Shares" - total number of shares

Note 1: this function is available ONLY in PROFESSIONAL edition, calling it using Standard edition will give you NULL values for all fields

Note 2: works only if data source uses real time data source (plugin)

Note 3: availablity of data depends on underlying data source - check the real-time quote window to see if given field is available

Note 4: function result represents the current value at the time of the call /formula execution/, and they will be refreshed depending on chart or commentary refresh interval /settable in preferences/. Built-in real time quote window is refreshed way more often (at least 10 times per second)

EXAMPLE

```
"Bid = "+GetRTData("Bid");
"Ask = "+GetRTData("Ask");
```

```
            "Last = "+GetRTData("Last");
            "Vol = "+GetRTData("TradeVolume");

            "EPS = "+GetRTData("EPS");
            "52week high = "+GetRTData("52weekhigh");
```

SEE ALSO GetRTDataForeign() function

References:

The **GetRTData** function is used in the following formulas in AFL on-line library:

- AFL Stock Price
- Bid Vs Ask Dashboard

More information:

See updated/extended version on-line.

GetRTDataForeign - retrieves the real-time data fields (for specified symbol)

Miscellaneous functions
(AmiBroker 4.80)

SYNTAX **GetRTDataForeign("fieldname" , "symbol")**

RETURNS NUMBER

FUNCTION This function is similar to GetRTData but allows to specify symbol OTHER than currently selected and it is much faster than SetForeign/GetRTData combo.

The function retrieves the LAST (the most recent) value of the following fields reported by streaming real time data source for specified symbol:

- "Ask" - current best ask price
- "AskSize " - current ask size
- "Bid" - current best bid price
- "BidSize " - current bid size
- "52WeekHigh" - 52 week high value
- "52WeekHighDate" - 52 week high date (in datenum format)
- "52WeekLow" - 52 week low value
- "52WeekLowDate" - 52 week low date (in datenum format)
- "Change" - change since yesterdays close
- "Dividend" - last dividend value
- "DivYield" - dividend yield
- "EPS" - earnings per share
- "High" - current day's high price
- "Low" - current day's low price
- "Open" - current day's open price
- "Last" - last trade price
- "OpenInt" - current open interest
- "Prev" - previous day close
- "TotalVolume" - total today's volume
- "TradeVolume" - last trade volume
- "ChangeDate" - datenum (YYYMMDD) of last data change
- "ChangeTime" - timenum (HHMMSS) of last data change
- "UpdateDate" - datenum (YYYMMDD) of last data update
- "UpdateTime" - timenum (HHMMSS) of last data update
- "Shares" - total number of shares

Note 1: this function is available ONLY in PROFESSIONAL edition, calling it using Standard edition will give you NULL values for all fields

Note 2: works only if data source uses real time data source (plugin)

Note 3: availablity of data depends on underlying data source - check the real-time quote window to see if given field is available

Note 4: function result represents the current value at the time of the call /formula execution/, and they will be refreshed depending on chart or commentary refresh interval /settable in preferences/. Built-in real time quote window is refreshed way more often (at least 10 times per second)

EXAMPLE

```
"Bid = "+GetRTDataForeign("Bid");
"Ask = "+GetRTData("Ask");
"Last = "+GetRTData("Last");
"Vol = "+GetRTData("TradeVolume");

"EPS = "+GetRTDataForeign("EPS", "AAPL");
"52week high = "+GetRTDataForeign("52weekhigh", "MSFT");
```

SEE ALSO GetRTData() function , SetForeign() function

References:

The **GetRTDataForeign** function is used in the following formulas in AFL on-line library:

- Heatmap V1

More information:

See updated/extended version on-line.

GetScriptObject
- get access to script COM object

Miscellaneous functions
(AmiBroker 3.80)

SYNTAX	**getscriptobject( )**
RETURNS	OBJECT
FUNCTION	Retrieves AFL host's script object. This allows to call functions defined in JScript/VBScript directly from AFL.

EXAMPLE

```
EnableScript("jscript")
function MyAdd(x, y)
{
return x+y;
}
%>
script = GetScriptObject();
WriteVal( script.MyAdd( 7, 9 ) ); // call the function defined in JScript
```

SEE ALSO

References:

The **GetScriptObject** function is used in the following formulas in AFL on-line library:

- accum/dist mov avg crossover SAR
- AllinOneAlerts - Module
- AR_Prediction.afl
- Heatmap V1
- Now Send Push Notifications From Amibroker
- nth (1 - 8) Order Polynomial Fit
- Polarized Fractal Efficiency
- Standard Error Bands
- tomy_frenchy
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

GetTradingInterface - retrieves OLE automation object to automatic trading interfac

Trading system toolbox
(AmiBroker 4.70)

SYNTAX **GetTradingInterface(**

RETURNS OBJECT

FUNCTION Retrieves OLE automation object to automatic trading interface. *"Name"* is the interface name. You have to have trading interface installed separately to make it work otherwise you will get the error message attempting to use this function. Trading interface for Interactive Brokers is available from download section: http://www.amibroker.com/download.html

EXAMPLE

SEE ALSO

References:

The **GetTradingInterface** function is used in the following formulas in AFL on-line library:

- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- Manual Bracket Order Trader
- Moving Averages NoX
- Reconnect TWS
- Visi-Trade

More information:

See updated/extended version on-line.

GfxArc
- draw an arc

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxArc(x1, y1, x2, y2, x3, y3, x4, y4)**

RETURNS NOTHING

FUNCTION Draws an elliptical arc. The arc drawn by using the function is a segment of the ellipse defined by the specified bounding rectangle.

The actual starting point of the arc is the point at which a ray drawn from the center of the bounding rectangle through the specified starting point intersects the ellipse. The actual ending point of the arc is the point at which a ray drawn from the center of the bounding rectangle through the specified ending point intersects the ellipse. The arc is drawn in a counterclockwise direction.

Parameters

- x1 - x-coordinate of the upper left corner of the bounding rectangle
- y1 - y-coordinate of the upper left corner of the bounding rectangle
- x2 - x-coordinate of the lower right corner of the bounding rectangle
- y2 - y-coordinate of the lower right corner of the bounding rectangle
- x3 - x-coordinate of the arc's starting point.
- y3 - y-coordinate of the arc's starting point.
- x4 - x-coordinate of the arc's ending point.
- y4 - y-coordinate of the arc's ending point.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxSelectPen( colorRed );
 GfxArc(100,0,200,100,150,0,200,50);
```

SEE ALSO GfxChord() function , GfxPie() function , GfxSelectPen() function

References:

The **GfxArc** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GfxChord
- draw a chord

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxChord(x1, y1, x2, y2, x3, y3, x4, y4)**

RETURNS NOTHING

FUNCTION Draws a chord (a closed figure bounded by the intersection of an ellipse and a line segment). The (x1, y1) and (x2, y2) parameters specify the upper-left and lower-right corners, respectively, of a rectangle bounding the ellipse that is part of the chord. The (x3, y3) and (x4, y4) parameters specify the endpoints of a line that intersects the ellipse. The chord is drawn by using the selected pen and filled by using the selected brush.

Parameters

- x1 - x-coordinate of the upper left corner of the bounding rectangle
- y1 - y-coordinate of the upper left corner of the bounding rectangle
- x2 - x-coordinate of the lower right corner of the bounding rectangle
- y2 - y-coordinate of the lower right corner of the bounding rectangle
- x3 - x-coordinate of the chord's starting point.
- y3 - y-coordinate of the chord's starting point.
- x4 - x-coordinate of the chord's ending point.
- y4 - y-coordinate of the chord's ending point.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE
```
GfxSelectPen( colorRed );
GfxSelectSolidBrush( colorBlue );
GfxChord(100,0,200,100,150,0,200,50);
```

SEE ALSO GfxSelectPen() function , GfxSelectSolidBrush() function , GfxPie() function

References:

The **GfxChord** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GfxCircle
- draw a circle

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxCircle(x, y, radius)**

RETURNS NOTHING

FUNCTION Draws a circle. The center of the circle is given by x and y parameters. The circle is drawn with the current pen, and its interior is filled with the current brush.

Parameters

- x - x-coordinate of the center of the circle
- y - y-coordinate of the the center of the circle
- radius - radius of the circle

This function is essentially the same as GfxEllipse(x - radius, y - radius, x + radius, y + radius);

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE GfxCircle(100, 100, 50);

SEE ALSO GfxEllipse() function , GfxRoundRect() function , GfxSelectPen() function , GfxSelectSolidBrush() function

References:

The **GfxCircle** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling

More information:

See updated/extended version on-line.

GfxDrawImage - draw bitmap image

Low-level graphics
(AmiBroker 6.20)

SYNTAX **GfxDrawImage("filename", x, y)**

RETURNS NOTHING

FUNCTION The function draws bitmap image given in file. Top left corner of the image is specified by x, y. Currently GfxDrawImage supports BMP and PNG files.

Per-pixel transparency (alpha channel) is supported ONLY for 32-bit PNG files. PNG files with 32-bit depth are rendered with AlphaBlend which is not be available on oldest systems (Win95). BMP files PNG files of lower depths are rendered without transparency using BitBlt (fast and always available)

Loading and drawing BMP files is much faster than PNG (as much as 10 times faster), approx load and rendering time is 0.3 ms for BMP, 3 ms for PNG (alphablended). Timings that you get from Code check and profile are misleading for Gfx functions because they don't include actual on-screen rendering.

EXAMPLE

```
GfxDrawImage( "wizard.bmp", 100, 0 ); // bitmap - fast
GfxDrawImage( "logo.png", 30, 30 ); // png - supports per-pixel alpha channel / transparency
```

SEE ALSO

References:

The **GfxDrawImage** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GfxDrawText - draw a text (clipped to rectangle)

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxDrawText("text", left, top, right, bottom, format = 0)**

RETURNS NOTHING

FUNCTION Formats and draws text in the given rectangle. It formats text by expanding tabs into appropriate spaces, aligning text to the left, right, or center of the given rectangle, and breaking text into lines that fit within the given rectangle. The type of formatting is specified by format argument. When format is not specified the text is aligned to the top/left corner.

Parameters:

- **"text"** - string to be drawn
- **left** - x-coordinate of upper left corner of the clipping rectangle
- **top** - y-coordinate of upper left corner of the clipping rectangle
- **right** - x-coordinate of lower right corner of the clipping rectangle
- **bottom** - y-coordinate of lower right corner of the clipping rectangle
- **format** - specifies the method of formatting the text. It can be any combination of the following values (combine using the bitwise OR operator):
  - ♦ DT_BOTTOM = 8 - Specifies bottom-justified text. This value must be combined with DT_SINGLELINE.
  - ♦ DT_CENTER = 1 - Centers text horizontally.
  - ♦ DT_END_ELLIPSIS = 32768 or DT_PATH_ELLIPSIS = 16384 - Replaces part of the given string with ellipses, if necessary, so that the result fits in the specified rectangle. You can specify DT_END_ELLIPSIS to replace characters at the end of the string, or DT_PATH_ELLIPSIS to replace characters in the middle of the string. If the string contains backslash (\) characters, DT_PATH_ELLIPSIS preserves as much as possible of the text after the last backslash.
  - ♦ DT_EXPANDTABS = 64 - Expands tab characters. The default number of characters per tab is eight.
  - ♦ DT_LEFT = 0 - Aligns text flush-left.
  - ♦ DT_NOCLIP = 256 - Draws without clipping. DrawText is somewhat faster when DT_NOCLIP is used.
  - ♦ DT_NOPREFIX = 2048 - Turns off processing of prefix characters. Normally, DrawText interprets the ampersand (&) mnemonic-prefix character as a directive to underscore the character that follows, and the two-ampersand (&&) mnemonic-prefix characters as a directive to print a single ampersand. By specifying DT_NOPREFIX, this processing is turned off.
  - ♦ DT_RIGHT = 2 - Aligns text flush-right.
  - ♦ DT_SINGLELINE = 32 - Specifies single line only. Carriage returns and linefeeds do not break the line.
  - ♦ DT_TOP = 0 - Specifies top-justified text (single line only).
  - ♦ DT_VCENTER = 4 - Specifies vertically centered text (single line only).
  - ♦ DT_WORDBREAK = 16 - Specifies word-breaking. Lines are automatically broken between words if a word would extend past the edge of the rectangle specified by lpRect. A carriage return linefeed sequence will also break the line.

Note: DT_ constants come from Windows API and are provided here for reference only. They are not defined in AmiBroker therefore you should use numerical values instead.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
// formatted text output sample via low-level gfx functions


CellHeight = 20;
CellWidth = 100;
GfxSelectFont( "Tahoma", CellHeight/2 );

function PrintInCell( string, row, Col )
{
GfxDrawText( string, Col * CellWidth, row * CellHeight, (Col + 1 ) *
CellWidth, (row + 1 ) * CellHeight, 0 );
}

PrintInCell( "Open", 0, 0 );
PrintInCell( "High", 0, 1 );
PrintInCell( "Low", 0, 2 );
PrintInCell( "Close", 0, 3 );
PrintInCell( "Volume", 0, 4 );

GfxSelectPen( colorBlue );
for( i = 1; i < 10 && i < BarCount; i++ )
{
PrintInCell( StrFormat("%g", O[ i ] ), i, 0 );
PrintInCell( StrFormat("%g", H[ i ] ), i, 1 );
PrintInCell( StrFormat("%g", L[ i ] ), i, 2 );
PrintInCell( StrFormat("%g", C[ i ] ), i, 3 );
PrintInCell( StrFormat("%g", V[ i ] ), i, 4 );
GfxMoveTo( 0, i * CellHeight );
GfxLineTo( 5 * CellWidth, i * CellHeight );
}
GfxMoveTo( 0, i * CellHeight );
GfxLineTo( 5 * CellWidth, i * CellHeight );

for( Col = 1; Col < 6; Col++ )
{
GfxMoveTo( Col * CellWidth, 0);
GfxLineTo( Col * CellWidth, 10 * CellHeight );
}
```

SEE ALSO GfxSetTextColor() function , GfxTextOut() function , GfxSetBkColor() function , GfxSetBkMode() function

References:

The **GfxDrawText** function is used in the following formulas in AFL on-line library:

- Basket Trading System T101

- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Continuous Contract Rollover
- elliott wave manual labelling
- Gfx Toolkit
- GFX ToolTip
- Halftrend
- Heatmap V1
- Market Breadth Chart-In-Chart
- Multiple Ribbon Demo
- Range Filter - Trading Strategy
- TAPE READING
- Visi-Trade

More information:

See updated/extended version on-line.

GfxEllipse
- draw an ellipse

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxEllipse(x1, y1, x2, y2)**

RETURNS NOTHING

FUNCTION Draws an ellipse. The center of the ellipse is the center of the bounding rectangle specified by x1, y1, x2, and y2. The ellipse is drawn with the current pen, and its interior is filled with the current brush.

Parameters

- x1 - x-coordinate of the upper left corner of the bounding rectangle
- y1 - y-coordinate of the upper left corner of the bounding rectangle
- x2 - x-coordinate of the lower right corner of the bounding rectangle
- y2 - y-coordinate of the lower right corner of the bounding rectangle

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE GfxEllipse(10, 10, 200, 100);

SEE ALSO GfxSelectPen() function , GfxSelectSolidBrush() function

References:

The **GfxEllipse** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GfxFillSolidRect - fill rectangle with solid color

Low-level graphics
(AmiBroker 6.10)

SYNTAX **GfxFillSolidRect(x1, y1, x2, y2, color)**

RETURNS NOTHING

FUNCTION GfxFillSolidRect is the fastest method to fill solid rectangle with single color (faster than GfxRectangle).

- x1, y1 define upper left corner,
- x2, y2 define lower right corner of the rectangle.

x1 should be less or equal x2. y1 should be less or equal y2.

The function does not require and does not use brush (so you do NOT need to call GfxSelectSolidBrush before it).

EXAMPLE

```
pw = Status("pxwidth");
ph = Status("pxheight");

GfxFillSolidRect( 0, 0, pw, ph, colorBlack );
```

SEE ALSO GfxRoundRect() function , GfxRectangle() function , GfxSelectSolidBrush() function

References:

The **GfxFillSolidRect** function is used in the following formulas in AFL on-line library:

- Multi-color Volume At Price (VAP)

More information:

See updated/extended version on-line.

GfxGetTextWidth - get pixel width of text

Low-level graphics
(AmiBroker 4.80)

SYNTAX **GfxGetTextWidth("text")**

RETURNS NUMBER

FUNCTION The function returns pixel width of specified string. NOTE: it is slow because it has to create temporary DC and font to measure the text. It takes 40us (microseconds), that is about 40 times more than other Gfx functions.

EXAMPLE

```
GfxSetZOrder(5 );
GfxSelectFont("Tahoma", 30 );
GfxSetTextColor( colorWhite );
text = "This is a test";
GfxTextOut(text, 0, 50);
GfxSetTextColor( colorRed );
GfxTextOut("second part in red", GfxGetTextWidth(text), 50 );
```

SEE ALSO GfxTextOut() function

References:

The **GfxGetTextWidth** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit
- TAPE READING

More information:

See updated/extended version on-line.

GfxGradientRect - draw a rectangle with gradient fill

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxGradientRect(x1, y1, x2, y2, fromcolor, tocolor)**

RETURNS NOTHING

FUNCTION Draws a rectangle. The interior of the rectangle is filled using gradient color.

Parameters

- x1 - x-coordinate of the upper left corner of the rectangle
- y1 - y-coordinate of the upper left corner of the rectangle
- x2 - x-coordinate of the lower right corner of the rectangle
- y2 - y-coordinate of the lower right corner of the rectangle
- fromcolor - the 'upper' color of the gradient fill
- tocolor - the 'lower' color of the gradient fill

The rectangle extends up to, but does not include, the right and bottom coordinates. This means that the height of the rectangle is y2 - y1 and the width of the rectangle is x2 - x1. Both the width and the height of a rectangle must be greater than 2 and less than 32767.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE GfxGradientRect(10, 10, 100, 100, colorWhite, colorGrey50);

SEE ALSO GfxRoundRect() function , GfxRectangle() function

References:

The **GfxGradientRect** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- Gfx Toolkit
- Halftrend
- Open Range Breakout Trading System
- Probability Density & Gaussian Distribution
- Range Filter - Trading Strategy
- Visi-Trade

More information:

See updated/extended version on-line.

GfxLineTo
- draw a line to specified point

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxLineTo(x, y)**

RETURNS NOTHING

FUNCTION Draws a line from the current position up to, but not including, the point specified by x and y. The line is drawn with the selected pen. The current position is set to x,y. Parameters

- x - Specifies the x-coordinate of the end point of the line.
- y - Specifies the y-coordinate of the end point of the line.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxMoveTo( 0, 0 );
GfxLineTo( 100, 100 );
```

SEE ALSO GfxMoveTo() function

References:

The **GfxLineTo** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Continuous Contract Rollover
- elliott wave manual labelling
- Gfx Toolkit
- Harmonic Pattern Detection
- High Low Detection code
- Market Breadth Chart-In-Chart
- Probability Density & Gaussian Distribution
- Visi-Trade
- Volume Charts

More information:

See updated/extended version on-line.

GfxMoveTo
- move graphic cursor to new position

Low-level graphics
(AmiBroker 50)

SYNTAX	**GfxMoveTo(x, y)**
RETURNS	NOTHING
FUNCTION	Moves the current position to the point specified by x and y. Parameters • x - Specifies the x-coordinate of the new position. • y - Specifies the y-coordinate of the new position. NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.
EXAMPLE	GfxMoveTo(10, 20);
SEE ALSO	

References:

The **GfxMoveTo** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Continuous Contract Rollover
- elliott wave manual labelling
- Gfx Toolkit
- Harmonic Pattern Detection
- High Low Detection code
- Market Breadth Chart-In-Chart
- Probability Density & Gaussian Distribution
- Visi-Trade

More information:

See updated/extended version on-line.

GfxPie - draw a pie

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxPie(x1, y1, x2, y2, x3, y3, x4, y4)**

RETURNS NOTHING

FUNCTION Draws a pie-shaped wedge by drawing an elliptical arc whose center and two endpoints are joined by lines. The center of the arc is the center of the bounding rectangle specified by x1, y1, x2, and y2. The starting and ending points of the arc are specified by x3, y3, x4, and y4.

The arc is drawn with the selected pen, moving in a counterclockwise direction. Two additional lines are drawn from each endpoint to the arc's center. The pie-shaped area is filled with the current brush. If x3 equals x4 and y3 equals y4, the result is an ellipse with a single line from the center of the ellipse to the point (x3, y3) or (x4, y4).

Parameters

- x1 - x-coordinate of the upper left corner of the bounding rectangle
- y1 - y-coordinate of the upper left corner of the bounding rectangle
- x2 - x-coordinate of the lower right corner of the bounding rectangle
- y2 - y-coordinate of the lower right corner of the bounding rectangle
- x3 - x-coordinate of the arc's starting point. This point does not have to lie exactly on the arc.
- y3 - y-coordinate of the arc's starting point. This point does not have to lie exactly on the arc.
- x4 - x-coordinate of the arc's ending point. This point does not have to lie exactly on the arc.
- y4 - y-coordinate of the arc's ending point. This point does not have to lie exactly on the arc.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxSelectPen( colorRed );
GfxSelectSolidBrush( colorBlue );
 GfxPie(100,0,200,100,150,0,200,50);
```

SEE ALSO GfxSelectPen() function , GfxSelectSolidBrush() function

References:

The **GfxPie** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GfxPolygon
- draw a polygon

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxPolygon(x1, y1, x2, y2, ...)**

RETURNS NOTHING

FUNCTION Draws a polygon consisting of two or more points (vertices) connected by lines, using the current pen. The system closes the polygon automatically, if necessary, by drawing a line from the last vertex to the first.

This function takes variable number of arguments and accepts up to 12 points (24 arguments = 12 co-ordinate pairs). The number of arguments must be even as each pair represents (x,y) co-ordinates of the vertex.

The polygon is filled with current brush and outline is painted with current pen.

Parameters:

- x1 - x co-ordinate of first point
- y1 - y co-ordinate of first point
- x2 - x co-ordinate of second point
- y2 - y co-ordinate of second point
- ...
- x12 - x co-ordinate of 12th point
- y12 - y co-ordinate of 12th point

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxSelectPen( colorGreen, 2 );
GfxSelectSolidBrush( colorYellow );
GfxPolygon(250,200,200,200,250,0,200,50);
```

SEE ALSO GfxPolyline() function , GfxSelectPen() function , GfxSelectSolidBrush() function

References:

The **GfxPolygon** function is used in the following formulas in AFL on-line library:

- Harmonic Pattern Detection

More information:

See updated/extended version on-line.

GfxPolyline - draw a polyline

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxPolyline(x1, y1, x2, y2, ...)**

RETURNS NOTHING

FUNCTION Draws a set of line segments connecting the points specified by arguments (x1,y1), (x2,y2), ... The lines are drawn from the first point through subsequent points using the current pen. Unlike the GfxLineTo function, the GfxPolyline function neither uses nor updates the current position.

This function takes variable number of arguments and accepts up to 12 points (24 arguments = 12 co-ordinate pairs). The number of arguments must be even as each pair represents (x,y) co-ordinates of the point.

Parameters:

- x1 - x co-ordinate of first point
- y1 - y co-ordinate of first point
- x2 - x co-ordinate of second point
- y2 - y co-ordinate of second point
- ...
- x12 - x co-ordinate of 12th point
- y12 - y co-ordinate of 12th point

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxSelectPen( colorGreen, 2 );
 GfxPolyline(250,200,200,200,250,0,200,50);
```

SEE ALSO GfxPolygon() function , GfxSelectPen() function

References:

The **GfxPolyline** function is used in the following formulas in AFL on-line library:

- Market Profile

More information:

See updated/extended version on-line.

GfxRectangle - draw a rectangle

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxRectangle(x1, y1, x2, y2)**

RETURNS NOTHING

FUNCTION Draws a rectangle using the current pen. The interior of the rectangle is filled using the current brush.

Parameters

- x1 - x-coordinate of the upper left corner of the rectangle
- y1 - y-coordinate of the upper left corner of the rectangle
- x2 - x-coordinate of the lower right corner of the rectangle
- y2 - y-coordinate of the lower right corner of the rectangle

The rectangle extends up to, but does not include, the right and bottom coordinates. This means that the height of the rectangle is y2 - y1 and the width of the rectangle is x2 - x1. Both the width and the height of a rectangle must be greater than 2 and less than 32767.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE GfxRectangle(10, 10, 30, 30)

SEE ALSO GfxSelectPen() function , GfxSelectSolidBrush() function

References:

The **GfxRectangle** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- elliott wave manual labelling
- Gfx Toolkit
- GFX ToolTip
- Market Breadth Chart-In-Chart
- Market Profile
- Support and resistance
- Volume Charts

More information:

See updated/extended version on-line.

GfxRoundRect
- draw a rectangle with rounded corners

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxRoundRect(x1, y1, x2, y2, x3, y3)**

RETURNS NOTHING

FUNCTION Draws a rectangle with rounded corners using the current pen. The interior of the rectangle is filled using the current brush.

Parameters

- x1 - x-coordinate of the upper left corner of the rectangle
- y1 - y-coordinate of the upper left corner of the rectangle
- x2 - x-coordinate of the lower right corner of the rectangle
- y2 - y-coordinate of the lower right corner of the rectangle
- x3 - the width of the ellipse used to draw the rounded corners
- y3 - the height of the ellipse used to draw the rounded corners

The rectangle extends up to, but does not include, the right and bottom coordinates. This means that the height of the rectangle is y2 - y1 and the width of the rectangle is x2 - x1. Both the width and the height of a rectangle must be greater than 2 and less than 32767.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE GfxRoundRect(10, 10, 100, 100, 15, 15);

SEE ALSO GfxRectangle() function , GfxSelectPen() function , GfxSelectSolidBrush() function

References:

The **GfxRoundRect** function is used in the following formulas in AFL on-line library:

- Alphatrend
- Bid Vs Ask Dashboard
- Gfx Toolkit
- GFX ToolTip
- Halftrend
- Heatmap V1
- Open Range Breakout Trading System
- Range Filter - Trading Strategy
- White Theme
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

GfxSelectFont - create / select graphic font

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSelectFont("facename", pointsize, weight = fontNormal, italic = False, underline = False, orientation = 0)**

RETURNS NOTHING

FUNCTION Initializes a font with the specified characteristics. Then selects the the as current for subsequent drawing operations.

Parameters:

- **"facename"** - specifies the typeface name of the font
- **pointsize** - specifies point size of the font (fractional numbers are allowed), for example 11.5 gives 11.5 point font.
- **weight** - specifies the font weight (in inked pixels per 1000). Typical values are: 300 - light, 400 - normal, 700 - bold, 800 - ultrabold
- **italic** - specifies whether the font is italic
- **underline** - specifies whether the font is underlined
- **orientation** - specifies the angle (in 0.1-degree units) between the baseline of a character and the x-axis. The angle is measured counterclockwise from the x-axis.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxSelectFont("Tahoma", 20, 700 );
GfxSetBkMode(1);
GfxSetTextColor(colorBrown);
GfxTextOut("Testing graphic capabilites", 20, 28 );
```

SEE ALSO GfxLineTo() function , GfxMoveTo() function , GfxSelectPen() function , GfxSelectSolidBrush() function , GfxSetPixel() function , GfxTextOut() function

References:

The **GfxSelectFont** function is used in the following formulas in AFL on-line library:

- Alphatrend
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bid Vs Ask Dashboard
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- elliott wave manual labelling
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Get Moneycontrol News Snippets into Amiboker
- Gfx Toolkit
- GFX ToolTip
- Halftrend
- Harmonic Pattern Detection
- Heatmap V1
- Market Breadth Chart-In-Chart

- Market Profile
- Multiple Ribbon Demo
- Nadaraya-Watson Envelope
- Open Range Breakout Trading System
- pattenz
- Perceptron
- Peter Cooper
- Point & figure Chart India Securities
- Probability Density & Gaussian Distribution
- Range Filter - Trading Strategy
- Stan Weinstein strategy
- TAPE READING
- Visi-Trade
- White Theme
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

GfxSelectHatchBrush - select hatch style brush

Low-level graphics
(AmiBroker 60)

SYNTAX **GfxSelectHatchBrush(color, style)**

RETURNS NOTHING

FUNCTION The function select hatch brush style for filled areas.

Supported hatch styles:

HS_HORIZONTAL 0 /* ----- */
HS_VERTICAL 1 /* ||||| */
HS_FDIAGONAL 2 /* \\ */
HS_BDIAGONAL 3 /* ///// */
HS_CROSS 4 /* +++++ */
HS_DIAGCROSS 5 /* xxxxx */

Hatch color is specified by color parameter, hatch background is specified by current background color see: GfxSetBkColor()

EXAMPLE

```
GfxSelectPen( colorOrange, 4 );
GfxSetBkColor( colorLightGrey );
GfxSelectHatchBrush( colorBlue, Param("Hatch pattern", 5, 0, 5 ) );
GfxCircle(100, 100, 20 );
```

SEE ALSO GfxSetBkColor() function

References:

The **GfxSelectHatchBrush** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit

More information:

See updated/extended version on-line.

GfxSelectPen - create / select graphic pen

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSelectPen(color, width = 1, penstyle = penSolid)**

RETURNS NOTHING

FUNCTION GfxSelectPen initializes (if not already initialized) a pen with the specified style, width, and color. Then selects the pen as current for subsequent drawing operations.

Parameters:

- **color** - specifies color for the pen
- **penstyle** - specifies the style for the pen. Solid=0, Dash=1, Dot=2, Null=5 (invisible pen). Lines of width > 1 can only use solid style. For a list of other possible values, see the Microsoft docs on CreatePen Windows API function.
- **width** - specifies the width of the pen. If this value is 0, the width in device units is always 1 pixel, regardless of the mapping mode (this is useful for drawing hairline lines on printer outputs).

More info on pens in Windows GDI: http://msdn2.microsoft.com/en-us/library/ms535467

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxSelectPen( colorGreen, 2 );
GfxSelectSolidBrush( colorYellow );
GfxPolygon(250,200,200,200,250,0,200,50);
```

SEE ALSO GfxLineTo() function , GfxMoveTo() function , GfxSetPixel() function , GfxTextOut() function

References:

The **GfxSelectPen** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Alphatrend
- Bid Vs Ask Dashboard
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- elliott wave manual labelling
- Gfx Toolkit
- GFX ToolTip
- Halftrend
- Harmonic Pattern Detection
- Heatmap V1
- High Low Detection code
- Market Breadth Chart-In-Chart
- Market Profile
- Open Range Breakout Trading System
- Probability Density & Gaussian Distribution
- Range Filter - Trading Strategy

- Support and resistance
- Visi-Trade
- Volume Charts
- White Theme
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

GfxSelectSolidBrush - create / select graphic brush

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSelectSolidBrush(color)**

RETURNS NOTHING

FUNCTION GfxSelectSolidBrush initializes a brush with a specified solid color. Then selects the brush as current for subsequent drawing operations.

Parameters:

- **color** - specifies color for the brush

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxSelectPen( colorGreen, 2 );
GfxSelectSolidBrush( colorYellow );
GfxPolygon(250,200,200,200,250,0,200,50);
```

SEE ALSO GfxLineTo() function , GfxMoveTo() function , GfxSelectPen() function , GfxSetPixel() function , GfxTextOut() function

References:

The **GfxSelectSolidBrush** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Alphatrend
- Bid Vs Ask Dashboard
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- elliott wave manual labelling
- Gfx Toolkit
- GFX ToolTip
- Halftrend
- Harmonic Pattern Detection
- Heatmap V1
- High Low Detection code
- Market Breadth Chart-In-Chart
- Market Profile
- Open Range Breakout Trading System
- Range Filter - Trading Strategy
- Support and resistance
- Volume Charts
- White Theme
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

GfxSelectStockObject - select built-in graphic object

Low-level graphics
(AmiBroker 60)

SYNTAX **GfxSelectStockObject(id)**

RETURNS NOTHING

FUNCTION GfxSelectStockObject selects Windows built-in (stock) GDI object. The kind of object is defined by *id* argument. Possible values are as follows:

0. WHITE_BRUSH = 0
1. LTGRAY_BRUSH = 1
2. GRAY_BRUSH = 2
3. DKGRAY_BRUSH = 3
4. BLACK_BRUSH = 4
5. NULL_BRUSH = 5 (the same as hollow brush), HOLLOW_BRUSH = 5
6. WHITE_PEN = 6
7. BLACK_PEN = 7
8. NULL_PEN = 8
9. -
10. OEM_FIXED_FONT = 10
11. ANSI_FIXED_FONT = 11
12. ANSI_VAR_FONT = 12
13. SYSTEM_FONT = 13
14. DEVICE_DEFAULT_FONT = 14
15. -
16. SYSTEM_FIXED_FONT = 16
17. DEFAULT_GUI_FONT = 17

EXAMPLE

```
GfxSelectPen( colorOrange, 4 );
GfxSelectStockObject( 5 ); // hollow brush
GfxCircle(100, 100, 20 );
```

SEE ALSO GfxSelectFont() function , GfxSelectHatchBrush() function , GfxSelectPen() function , GfxSelectSolidBrush() function

References:

The **GfxSelectStockObject** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GfxSetBkColor - set graphic background color

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSetBkColor(color)**

RETURNS NOTHING

FUNCTION Sets the current background color to the specified color. If the background mode is OPAQUE (see GfxSetBkMode), the system uses the background color to fill the gaps in styled lines, the gaps between hatched lines in brushes, and the background in character cells.

Parameters:

- color - specifies the new background color

  NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE GfxSetBkColor(ColorRGB(10, 20, 30));

SEE ALSO GfxSelectFont() function , GfxSetTextColor() function

References:

The **GfxSetBkColor** function is used in the following formulas in AFL on-line library:

- Basket Trading System T101
- elliott wave manual labelling
- Gfx Toolkit
- GFX ToolTip
- Point & figure Chart India Securities
- TAPE READING

More information:

See updated/extended version on-line.

GfxSetBkMode - set graphic background mode

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSetBkMode(bkmode)**

RETURNS NOTHING

FUNCTION Sets the background mode. The background mode defines whether the system removes existing background colors on the drawing surface before drawing text, hatched brushes, or any pen style that is not a solid line.

Parameters:

- bkmode - Specifies the mode to be set. This parameter can be either of the following values:
  OPAQUE = 2 - Background is filled with the current background color before the text, hatched brush, or pen is drawn. This is the default background mode.
  TRANSPARENT = 1 - Background is not changed before drawing

  NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE GfxSetBkMode(1); // set transparent mode

SEE ALSO GfxSetTextAlign() function , GfxSetTextColor() function , GfxTextOut() function , GfxSelectPen() function

References:

The **GfxSetBkMode** function is used in the following formulas in AFL on-line library:

- Alphatrend
- BEANS-Summary of Holdings
- Bid Vs Ask Dashboard
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- elliott wave manual labelling
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Get Moneycontrol News Snippets into Amiboker
- Gfx Toolkit
- GFX ToolTip
- Halftrend
- Heatmap V1
- Market Profile
- Nadaraya-Watson Envelope
- Open Range Breakout Trading System
- pattenz
- Perceptron
- Peter Cooper
- Probability Density & Gaussian Distribution
- Range Filter - Trading Strategy
- Stan Weinstein strategy
- Visi-Trade

- White Theme
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

GfxSetCoordsMode - set low-level graphics co-ordinate mode

Low-level graphics
(AmiBroker 4.80)

SYNTAX **GfxSetCoordsMode(mode)**

RETURNS NOTHING

FUNCTION The function allows to switch co-ordinate system for low-level gfx functions:

- mode = 0 - the default screen pixel mode, where both X and Y are expressed in pixels
- mode = 1 - bar / price mode where X is expressed in bar index and Y is expressed in price. This new mode allows way easier overlays on top of existing charts without need to do conversion between bars/price pixels and without any extra refresh normally required in old versions when Y scale changed.
- mode = 2 - X coordinate is pixel, Y coordinate is price (new in 6.20)
- mode = 3 - X coordinate is bar index, Y is pixel (new in 6.20)

The function can be called to switch back and forth from pixel -> bar/price mode and vice versa a number of times allowing to mix different modes in the same chart. When co-ordinate mode 1 is selected (bar/price), co-ordinates can be fractional. For example if x is 2.5 it means half way between bar 2 and 3.

EXAMPLE

```
// The sample shows how using GfxSetCoordsMode( 1 )
// results in
// a) easier coding of overlay charts that plot on top of built-in
charts (no need to convert from bar/price to pixels)
// b) perfect matching between built-in Plot() and Gfx positioning
Plot( C, "Price", colorDefault, styleLine );
GfxSetOverlayMode( 1 );
GfxSetCoordsMode( 1 ); // bar/price mode (instead of pixel)

GfxSelectSolidBrush( colorRed );
GfxSelectPen( colorRed );

boxheight = 0.01 * ( HighestVisibleValue( C ) - LowestVisibleValue(
C ) );

bi = BarIndex();

start = FirstVisibleValue( bi );
end = LastVisibleValue( bi );

for ( i = start; i <= end; i++ )
{
     Cl = Close[ i ];
     Op = Open[ i ];

     Color = IIf( Cl > Op, colorGreen, colorRed );
     GfxSelectPen( Color );
     GfxSelectSolidBrush( Color );
```

```
                bodyup = Max( Op, Cl );
                bodydn = Min( Op, Cl );

                GfxEllipse( i - 0.4, bodyup, i + 0.4, bodydn );

                GfxMoveTo( i, H[ i ] );
                GfxLineTo( i, bodyup );
                GfxMoveTo( i, bodydn );
                GfxLineTo( i, L[ i ] );
           }
```

SEE ALSO

References:

The **GfxSetCoordsMode** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- Harmonic Pattern Detection
- High Low Detection code
- Multi-color Volume At Price (VAP)
- Support and resistance
- TAPE READING
- White Theme

More information:

See updated/extended version on-line.

GfxSetOverlayMode - set low-level graphic overlay mode

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSetOverlayMode(mode = 0)**

RETURNS NOTHING

FUNCTION Sets overlay mode for low-level graphics.

Parameters:

- **mode** - desired overlay mode. Possible values are:
  0 - (default) low-level graphic is overlaid on top of charts
  1 - charts are overlaid on top of low-level graphics
  2 - only low-level graphics is displayed (no charts, no grids, etc)

To learn more about low level graphics please read Tutorial: Using low-level graphics

EXAMPLE GfxSetOverlayMode(2); // don't display charts nor grids

SEE ALSO GfxArc() function , GfxChord() function , GfxCircle() function , GfxDrawText() function , GfxEllipse() function , GfxGradientRect() function , GfxLineTo() function , GfxMoveTo() function , GfxPie() function , GfxPolygon() function , GfxPolyline() function , GfxRectangle() function , GfxRoundRect() function [[331:Gfx

References:

The **GfxSetOverlayMode** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Color Combination
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- GFX ToolTip
- Halftrend
- Market Breadth Chart-In-Chart
- Market Profile
- Multi-color Volume At Price (VAP)
- Nadaraya-Watson Envelope
- pattenz
- Peter Cooper
- Range Filter - Trading Strategy
- Stan Weinstein strategy
- TAPE READING
- Visi-Trade
- White Theme
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

GfxSetPixel
- set pixel at specified position to specified color

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSetPixel(x, y, color)**

RETURNS NOTHING

FUNCTION Sets the pixel at the specified x, y coordinates to the specified color. The point must be in the visible part drawing surface (otherwise it won't be painted). Parameters

- x - Specifies the x-coordinate of the point.
- y - Specifies the y-coordinate of the point.
- color - specifies the color to be used to paint the point

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE
```
GfxSetPixel( 20, 20 );
```

SEE ALSO GfxLineTo() function , GfxMoveTo() function

References:

The **GfxSetPixel** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GfxSetTextAlign - set text alignment

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSetTextAlign(align)**

RETURNS NOTHING

FUNCTION Sets the text-alignment flags.

The GfxTextOut function uses these flags when positioning a string of text on a display or device. The flags specify the relationship between a specific point and a rectangle that bounds the text. The coordinates of this point are passed as parameters to the TextOut member function. The rectangle that bounds the text is formed by the adjacent character cells in the text string.

Parameters:

- **align** - combination (binary-OR) of one or more the following flags:
  X-direction alignment:
  - TA_CENTER = 6 - Aligns the point with the horizontal center of the bounding rectangle.
  - TA_LEFT = 0 - Aligns the point with the left side of the bounding rectangle. This is the default setting.
  - TA_RIGHT = 2 - Aligns the point with the right side of the bounding rectangle.

  Y-direction alignment
  - TA_BASELINE = 24 - Aligns the point with the base line of the chosen font.
  - TA_BOTTOM = 8 - Aligns the point with the bottom of the bounding rectangle.
  - TA_TOP = 0 - Aligns the point with the top of the bounding rectangle. This is the default setting.

  flags that determine whether the current position is updated when text is written:
  - TA_NOUPDATECP = 0 - Does not update the current position after each call to a text-output function. This is the default setting.
  - TA_UPDATECP = 1 - Updates the current x-position after each call to a text-output function. The new position is at the right side of the bounding rectangle for the text. When this flag is set, the coordinates specified in calls to the GfxTextOut member function are ignored

Note: TA_ constants come from Windows API, they are given for reference only they are not predefined in AmiBroker, so you need to use numerical values.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE GfxSetTextAlign(6 | 24); // center and baseline alignment

SEE ALSO GfxSetTextColor() function , GfxTextOut() function , GfxSetBkColor() function , GfxSetBkMode() function , GfxSelectFont() function

References:

The **GfxSetTextAlign** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- GFX ToolTip
- Harmonic Pattern Detection
- Market Profile
- Nadaraya-Watson Envelope
- pattenz
- Peter Cooper
- Probability Density & Gaussian Distribution
- Stan Weinstein strategy
- White Theme
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

GfxSetTextColor - set graphic text color

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxSetTextColor(color)**

RETURNS NOTHING

FUNCTION Sets the text color to the specified color. AmiBroker will use this text color when writing text to this window using GfxTextOut or GfxDrawText.

The background color for a character is specified by the GfxSetBkColor and GfxSetBkMode member functions.

Parameters:

- color - Specifies the color of the text

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE
```
GfxSetTextColor( colorRed );
GfxSetTextColor( ColorRGB( 100, 200, 100 ) );
```

SEE ALSO GfxTextOut() function , GfxDrawText() function , GfxSelectFont() function , GfxSetBkColor() function , GfxSetBkMode() function

References:

The **GfxSetTextColor** function is used in the following formulas in AFL on-line library:

- Alphatrend
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bid Vs Ask Dashboard
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- elliott wave manual labelling
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Get Moneycontrol News Snippets into Amiboker
- Gfx Toolkit
- GFX ToolTip
- Halftrend
- Harmonic Pattern Detection
- Heatmap V1
- Market Breadth Chart-In-Chart
- Market Profile
- Multiple Ribbon Demo
- Nadaraya-Watson Envelope
- Open Range Breakout Trading System
- pattenz
- Peter Cooper
- Point & figure Chart India Securities

- Probability Density & Gaussian Distribution
- Range Filter - Trading Strategy
- Stan Weinstein strategy
- TAPE READING
- Visi-Trade
- White Theme
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

GfxSetZOrder - set current low-level graphic Z-order layer

Low-level graphics
(AmiBroker 4.80)

SYNTAX **GfxSetZOrder(layer)**

RETURNS NOTHING

FUNCTION Sets current z-order layer to be used for subsequent low level Gfx* calls

EXAMPLE

```
Plot( C, "Price", colorDefault );
GraphGridZOrder = 1;

GfxSetZOrder(0);
GfxSelectSolidBrush( colorGreen );
GfxCircle( 100, 100, 100 );

GfxSetZOrder(-1);
GfxSelectSolidBrush( colorRed );
GfxCircle( 150, 150, 100 );

GfxSetZOrder(-2);
GfxSelectSolidBrush( colorBlue );
GfxCircle( 180, 180, 100 );
```

SEE ALSO

References:

The **GfxSetZOrder** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- Harmonic Pattern Detection
- Support and resistance

More information:

See updated/extended version on-line.

GfxTextOut
- writes text at the specified location

Low-level graphics
(AmiBroker 50)

SYNTAX **GfxTextOut("text", x, y)**

RETURNS NOTHING

FUNCTION Writes a character string at the specified location using the currently selected font.

Parameters:

- "text" - Specifies the character string to be drawn
- x - Specifies the x-coordinate of the starting point of the text
- y - Specifies the y-coordinate of the starting point of the text

Character origins are at the upper-left corner of the character cell. By default, the current position is not used or updated by the function.

The font used can be set using GfxSelectFont() function. Text color can be set using GfxSetTextColor() function.

If a formula needs to update the current position when it calls GfxTextOut, the formula can call the GfxSetTextAlign function with flags set to 1 (TA_UPDATECP Windows flag). When this flag is set, GfxTextOut function ignores the x and y parameters on subsequent calls to GfxTextOut, using the current position instead.

The output of this function is NOT clipped. If you want clip text to user-defined rectangle, use GfxDrawText() function instead.

NOTE: This is LOW-LEVEL graphic function. To learn more about low-level graphic functions please read TUTORIAL: Using low-level graphics.

EXAMPLE

```
GfxSelectFont("Times New Roman", 16, 700, True );
GfxTextOut("Percent of shares held by:", 10 , 10 );
```

SEE ALSO GfxLineTo() function , GfxMoveTo() function , GfxSetPixel() function

References:

The **GfxTextOut** function is used in the following formulas in AFL on-line library:

- Alphatrend
- Bid Vs Ask Dashboard
- Computing Cointegration and ADF Dashboard
- elliott wave manual labelling
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Get Moneycontrol News Snippets into Amiboker
- Gfx Toolkit
- GFX ToolTip
- Halftrend
- Harmonic Pattern Detection
- Market Profile
- Nadaraya-Watson Envelope

- Open Range Breakout Trading System
- pattenz
- Perceptron
- Peter Cooper
- Point & figure Chart India Securities
- Probability Density & Gaussian Distribution
- Range Filter - Trading Strategy
- Stan Weinstein strategy
- TAPE READING
- White Theme
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

GicsID
- get GICS category information

Information / Categories
(AmiBroker 5.40)

SYNTAX **GicsID(mode)**

RETURNS STRING

FUNCTION The function gets information about current symbol GICS category.

- mode = 0 - returns string containing GICS code alone (such as " 15103020")
- mode = 1 - returns string containing GICS category name (such as "Paper Packaging")
- mode = 2 - returns string containing both code and name (such as "15103020 Paper Packaging")

EXAMPLE

```
printf("GICS(0) = " + GicsID( 0 ));
printf("GICS(1) = " + GicsID( 1 ));
printf("GICS(2) = " + GicsID( 2 ));

for( i = 10; i < 90; i+= 1 )
{
 gics_code = StrFormat("%02.0f", i );
 printf("In Gics '"+ gics_code + "' = %g\n", InGics( gics_code ) );
}
```

SEE ALSO InGICS() function , INDUSTRYID() function , GROUPID() function , MARKETID() function , SECTORID() function

References:

The **GicsID** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GroupID
- get group ID/name

Information / Categories
(AmiBroker 3.80)

SYNTAX	**GroupID(mode = 0)**
RETURNS	NUMBER/STRING
FUNCTION	Retrieves current stock group ID/name When mode = 0 (the default value) this function returns numerical group ID (consecutive group number) When mode = 1 this function returns name of the group.

EXAMPLE	Filter = GroupID() == 7 OR GroupID() == 9;
	AddTextColumn(GroupID(1), "Group name");

SEE ALSO	

References:

The **GroupID** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiButton
- create on-chart button control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiButton("text", id, x, y, width, height, notifyflags, style = 0)**

RETURNS NUMBER

FUNCTION The function creates on-chart GUI button control.

Parameters:

- "text" - specifies button text
- id - unique control ID (must be integer greater than zero)
- x, y - coordinates of top left corner of the control
- width, height - dimensions of the control
- notifyflags - define which user actions cause formula to be notified
- style - defines windows control style (WS_* constants in Windows API) that gets OR-ed with default styles (that are added by AmiBroker automatically and typically are WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_TABSTOP)

Possible notify flags are:

- notifyClicked - when button/toggle/checkbox/radio is clicked
- notifySetFocus - when control gets input focus (via mouse or keyboard)
- notifyKillFocus - when control loses input focus (via mouse or keyboard)
- notifyHitReturn - when ENTER/RETURN key was pressed in edit field
- notifyEditChange - when control value changes (text inside edit field, date inside date/time, slider is moved to new position)
- notifyMouseEnter - when mouse hovers over the control
- notifyMouseLeave - when mouse leaves control area

The function returns **guiNew** when control was actually created (wasn't already present in the chart) or **guiExisting** when control already exists in given chart

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO

References:

The **GuiButton** function is used in the following formulas in AFL on-line library:

- Color Combination

More information:

See updated/extended version on-line.

GuiCheckBox
- creates on-chart checkbox control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiCheckBox("text", id, x, y, width, height, notifyflags, style = 0)**

RETURNS

FUNCTION The function creates on-chart GUI checkbox control.

Parameters:

- "text" - specifies checkbox text
- id - unique control ID (must be integer greater than zero)
- x, y - coordinates of top left corner of the control
- width, height - dimensions of the control
- notifyflags - define which user actions cause formula to be notified
- style - defines windows control style (WS_* constants in Windows API) that gets OR-ed with default styles (that are added by AmiBroker automatically and typically are WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_TABSTOP)

Possible notify flags are:

- notifyClicked - when button/toggle/checkbox/radio is clicked
- notifySetFocus - when control gets input focus (via mouse or keyboard)
- notifyKillFocus - when control loses input focus (via mouse or keyboard)
- notifyHitReturn - when ENTER/RETURN key was pressed in edit field
- notifyEditChange - when control value changes (text inside edit field, date inside date/time, slider is moved to new position)
- notifyMouseEnter - when mouse hovers over the control
- notifyMouseLeave - when mouse leaves control area

The function returns **guiNew** when control was actually created (wasn't already present in the chart) or **guiExisting** when control already exists in given chart

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO GuiButton() function , GuiEdit() function , GuiToggle() function

References:

The **GuiCheckBox** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiDateTime
- creates on-chart date-time picker control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiDateTime(id, x, y, width, height, notifyflags, style = 0)**

RETURNS NUMBER

FUNCTION The function creates on-chart GUI date-time control.

Parameters:

- id - unique control ID (must be integer greater than zero)
- x, y - coordinates of top left corner of the control
- width, height - dimensions of the control
- notifyflags - define which user actions cause formula to be notified
- style - defines windows control style (WS_* constants in Windows API) that gets OR-ed with default styles (that are added by AmiBroker automatically and typically are WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_TABSTOP)

Possible notify flags are:

- notifyClicked - when button/toggle/checkbox/radio is clicked
- notifySetFocus - when control gets input focus (via mouse or keyboard)
- notifyKillFocus - when control loses input focus (via mouse or keyboard)
- notifyHitReturn - when ENTER/RETURN key was pressed in edit field
- notifyEditChange - when control value changes (text inside edit field, date inside date/time, slider is moved to new position)
- notifyMouseEnter - when mouse hovers over the control
- notifyMouseLeave - when mouse leaves control area

The function returns **guiNew** when control was actually created (wasn't already present in the chart) or **guiExisting** when control already exists in given chart

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO GuiButton() function , GuiCheckBox() function , GuiEdit() function , GuiRadio() function , GuiToggle() function

References:

The **GuiDateTime** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiEdit - create on-chart edit control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiEdit(id, x, y, width, height, notifyflags, style = 0)**

RETURNS NUMBER

FUNCTION The function creates on-chart GUI edit control.

Parameters:

- id - unique control ID (must be integer greater than zero)
- x, y - coordinates of top left corner of the control
- width, height - dimensions of the control
- notifyflags - define which user actions cause formula to be notified
- style - defines windows control style (WS_* constants in Windows API) that gets OR-ed with default styles (that are added by AmiBroker automatically and typically are WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_TABSTOP)

Possible notify flags are:

- notifyClicked - when button/toggle/checkbox/radio is clicked
- notifySetFocus - when control gets input focus (via mouse or keyboard)
- notifyKillFocus - when control loses input focus (via mouse or keyboard)
- notifyHitReturn - when ENTER/RETURN key was pressed in edit field
- notifyEditChange - when control value changes (text inside edit field, date inside date/time, slider is moved to new position)
- notifyMouseEnter - when mouse hovers over the control
- notifyMouseLeave - when mouse leaves control area

The function returns **guiNew** when control was actually created (wasn't already present in the chart) or **guiExisting** when control already exists in given chart

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO

References:

The **GuiEdit** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit

More information:

See updated/extended version on-line.

GuiEnable
- enables or disables on-chart control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiEnable(id, enable)**

RETURNS NUMBER

FUNCTION Enables / disables GUI control. Disabled control can not be clicked and has "grayed" look.

Parameters:

- id - control ID
- enable - True if control should be enabled, or False if control should be disabled

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO GuiButton() function , GuiCheckBox() function , GuiDateTime() function , GuiEdit() function , GuiRadio() function , GuiSlider() function , GuiToggle() function

References:

The **GuiEnable** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiGetCheck - get checked state of control

GUI functions
(AmiBroker 6.30)

SYNTAX	**GuiGetCheck(id)**
RETURNS	NUMBER
FUNCTION	Get checked value of checkbox, toggle button or radio button. If control is checked (pressed) returns 1 (True), otherwise it returns 0 (False). Parameters: • id - control ID
EXAMPLE	
SEE ALSO	GuiCheckBox() function , GuiRadio() function , GuiToggle() function

References:

The **GuiGetCheck** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiGetEvent - get GUI event

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiGetEvent(num, what = 0)**

RETURNS NUMBER or STRING

FUNCTION The function retrieves queued GUI event (such as click, edit change, focus change, etc) that was triggered by on-chart GUI control

Parameters:

- *num* parameter defines the index of notification in the queue. 0 is first received (or "oldest") since last execution.
- *what,* defines what value is returned by GuiGetEvent function:
    - ♦ what = 0 - ID of control that received event (number)
    - ♦ what = 1 - the notification code (number)
    - ♦ what = 2 - the ID, the code and the notification description as text (string)

Usually there is zero (when formula execution was not triggered by UI event) or one event in the queue but note that there can be more than one event notification in the queue if your formula is slow to process. To retrieve them all use increasing "num" parameter as long as GuiGetEvent does not return zero (in what =0, =1 mode) or empty string (what=2).

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

```
// EXAMPLE 1. Just read all pending events
for( i = 0; id = GuiGetEvent( i ); i++ )
{
    code = GuiGetEvent( i, 1 );
    text = GuiGetEvent( i, 2 );
}
```

It is good practice to have a separate function that handles the events as shown below:

```
// EXAMPLE 2. Read and handle events using switch statement
idMyFirstButton = 1;
idMySecondButton = 2;

function CreateGUI()
{
     GuiButton( "enable", idMyFirstButton, 10, 60, 100, 30,
notifyClicked );
     GuiButton( "disable", idMySecondButton, 110, 60, 100, 30,
notifyClicked );
}

function HandleEvents()
{
    for ( n = 0; id = GuiGetEvent( n, 0 ); n++ ) // get the id of
the event
```

```
            {
                 code = GuiGetEvent( n, 1 );

                 switch ( id )
                 {
                     case idMyFirstButton:
                     // do something
                        break;

                     case idMySecondButton:
                     // do something else
                        break;

                     default:
                         break;
                 }
             }
        }

        CreateGUI();

        HandleEvents();
```

SEE ALSO GuiButton() function , GuiCheckBox() function , GuiDateTime() function , GuiEdit() function , GuiRadio() function , GuiSlider() function , GuiToggle() function

References:

The **GuiGetEvent** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiGetText - get text from on-chart control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiGetText(id)**

RETURNS STRING

FUNCTION The function gets the text value from on-chart GUI control (such as edit field, or date time).

Parameters:

- id - control ID

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO GuiEdit() function , GuiDateTime() function , GuiSlider() function

References:

The **GuiGetText** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit

More information:

See updated/extended version on-line.

GuiGetValue
- get numeric value of on-chart control

GUI functions
(AmiBroker 6.30)

SYNTAX	**GuiGetValue(id)**
RETURNS	NUMBER
FUNCTION	Returns the numeric value of GUI control (for example slider position). For non-slider controls value is number representing control text converted to number). For slider controls the value is within min-max range set by GuiSetRange. Parameters: • id - control ID
EXAMPLE	
SEE ALSO	GuiSlider() function , GuiSetRange() function

References:

The **GuiGetValue** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit

More information:

See updated/extended version on-line.

GuiRadio - creates on-chart radio button control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiRadio("text", id, x, y, width, height, notifyflags, style = 0)**

RETURNS NUMBER

FUNCTION The function creates on-chart GUI radio button control.

Parameters:

- "text" - specifies radio button text
- id - unique control ID (must be integer greater than zero)
- x, y - coordinates of top left corner of the control
- width, height - dimensions of the control
- notifyflags - define which user actions cause formula to be notified
- style - defines windows control style (WS_* constants in Windows API) that gets OR-ed with default styles (that are added by AmiBroker automatically and typically are WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_TABSTOP)

Possible notify flags are:

- notifyClicked - when button/toggle/checkbox/radio is clicked
- notifySetFocus - when control gets input focus (via mouse or keyboard)
- notifyKillFocus - when control loses input focus (via mouse or keyboard)
- notifyHitReturn - when ENTER/RETURN key was pressed in edit field
- notifyEditChange - when control value changes (text inside edit field, date inside date/time, slider is moved to new position)
- notifyMouseEnter - when mouse hovers over the control
- notifyMouseLeave - when mouse leaves control area

The function returns **guiNew** when control was actually created (wasn't already present in the chart) or **guiExisting** when control already exists in given chart

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO GuiButton() function , GuiCheckBox() function , GuiEdit() function , GuiToggle() function

References:

The **GuiRadio** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiSendKeyEvents - request GUI notifications for specified keys

GUI functions
(AmiBroker 6.40)

SYNTAX **GuiSendKeyEvents(``string``)**

RETURNS NOTHING

FUNCTION The function registers given characters to be sent as Gui event when keyboard key is pressed. GuiGetEvent will return code == notifyKeyDown and id of given character, for example id == 'D' for D letter

EXAMPLE

```
idMyFirstButton = 1;
idMySecondButton = 2;

function CreateGUI()
{
     GuiSendKeyEvents("ED"); // send notifications for keypresses of
"E" and "D"
     GuiButton( "Enable", idMyFirstButton, 10, 60, 100, 30,
notifyClicked );
     GuiButton( "Disable", idMySecondButton, 110, 60, 100, 30,
notifyClicked );
}

function HandleEvents()
{
    for ( n = 0; id = GuiGetEvent( n, 0 ); n++ ) // get the id of
the event
    {
         code = GuiGetEvent( n, 1 );

       if( code == notifyKeyDown )
       {
         switch( id )
         {
            case 'D':
               Say("pressed D key");
               break;

            case 'E':
               Say("pressed E key");
               break;
         }
       }
       else
         switch ( id )
         {
             case idMyFirstButton:
             // do something
            Say( "Pressed first button" );
                break;
```

```
                 case idMySecondButton:
                 // do something else
                Say( "Pressed second button" );
                    break;

                 default:
                     break;
             }
         }
     }
     CreateGUI();
     HandleEvents();
```

SEE ALSO GuiGetEvent() function

References:

The **GuiSendKeyEvents** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiSetCheck
- set checked state of on-chart control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiSetCheck(id, check)**

RETURNS

FUNCTION Sets checked value of checkbox, toggle button or radio button.

Parameters:

- id - control ID
- check - True if control should be checked, False otherwise

EXAMPLE

SEE ALSO GuiGetCheck() function , GuiCheckBox() function , GuiRadio() function , GuiToggle() function

References:

The **GuiSetCheck** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiSetColors
- set colors for GUI controls

GUI functions
(AmiBroker 4.30)

SYNTAX **GuiSetColors(idFrom, idTo, border , clrText = -1, clrBack = -1, clrBorder = -1, clrSelText = -1, clrSelBack = -1, clrSelBorder = -1, clrHoverText = -1, clrHoverBack = -1, clrHoverBorder = -1, clrDisText = -1, clrDisBack = -1, clrDisBorder = -1)**

RETURNS NOTHING

FUNCTION The function defines custom colors for on-chart GUI controls

Parameters:

- idFrom, idTo - define range of control IDs that will use new colors. To set color for single control use the same value for both idFrom and idTo
- border - defines border width of button (does not affect other control types)
- clrText, clrBack, clrBorder - define colors of control text (fgcolor), background (bgcolor) and border (if border width is > 0) in "default" control state (unselected) selectfgcolor. -1 means colorDefault and if all colors are set to default then control uses SYSTEM (Windows) look
- clrSelText, clrSelBack, clrSelBorder - define colors in selected state
- clrHoverText, clrHoverBack, clrHoverBorder - define colors in hover (mouse over) state
- clrDisText, clrDisBack, clrDisBorder - define colors in disabled state

Please note that currently only buttons support custom colors and custom border As for v6.21 Edit fields always use system look

EXAMPLE

SEE ALSO GuiSetFont() function

References:

The **GuiSetColors** function is used in the following formulas in AFL on-line library:

- Color Combination
- Gfx Toolkit

More information:

See updated/extended version on-line.

GuiSetFont - set the font for on-chart control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiSetFont("fontface", size)**

RETURNS NOTHING

FUNCTION Sets the font to be used by all chart controls.

Parameters:

- fontface - the name of font, such as "Verdana", or "Tahoma"
- size - point size of the font

EXAMPLE

SEE ALSO GuiButton() function , GuiCheckBox() function , GuiDateTime() function , GuiEdit() function , GuiRadio() function , GuiSlider() function , GuiToggle() function

References:

The **GuiSetFont** function is used in the following formulas in AFL on-line library:

- Color Combination
- Gfx Toolkit

More information:

See updated/extended version on-line.

GuiSetRange - set slider control range

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiSetRange(id, min, max, step, ticfreq)**

RETURNS NOTHING

FUNCTION The functions sets min-max range of slider control and it's tick frequency.

Parameters:

- id - control ID
- min, max - minimum and maximum value of slider position
- step - the smallest possible change of value (increment) of slider control
- ticfreq - set the frequency with which tick marks are displayed in a slider. For example, if the frequency is set to 2, a tick mark is displayed for every other increment in the slider’s range. The value of 1 means that every increment in the range is associated with a tick mark.

EXAMPLE

```
if( GuiSlider( 1, 10, 30, 200, 30, notifyEditChange ) == guiNew )
{
   // init values on control creation
  GuiSetValue( 1, 5 );
   GuiSetRange( 1, 1, 100, 0.1, 100 );
}

Title = "Value = " + GuiGetValue( 1 );
```

SEE ALSO GuiSlider() function

References:

The **GuiSetRange** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiSetText
- set text value of on-chart control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiSetText("text", id)**

RETURNS NUMBER

FUNCTION The function sets the text value of on-chart control (such as edit field, or date time).

Parameters:

- text - (STRING) to be set in the selected control
- id - control ID

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO GuiGetText() function , GuiEdit() function , GuiDateTime() function , GuiSlider() function

References:

The **GuiSetText** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit

More information:

See updated/extended version on-line.

GuiSetValue - set numeric value of on-chart control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiSetValue(id, value)**

RETURNS NUMBER

FUNCTION Sets the value of GUI control (for example slider position). For non-slider control value will be converted to text and assigned as GuiSetText(id, NumToStr(value)). For slider controls the value should be within min-max range set by GuiSetRange.

Parameters:

- id - control ID

Returns True (1) on success, False (0) on failure (when control does not exist)

EXAMPLE

SEE ALSO GuiSlider() function , GuiSetRange() function , GuiGetValue() function

References:

The **GuiSetValue** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit

More information:

See updated/extended version on-line.

GuiSetVisible
- shows or hides on-chart control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiSetVisible(id, visible)**

RETURNS NUMBER

FUNCTION The function makes on-chart GUI control visible or invisible (hidden). Hidden controls do not receive user input.

Parameters:

- id - control ID
- visible - True if control should be made visible, or False if control should be hidden

Returns True (1) on success, False (0) on failure

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO GuiButton() function , GuiCheckBox() function , GuiDateTime() function , GuiEdit() function , GuiEnable() function , GuiRadio() function , GuiSlider() function , GuiToggle() function

References:

The **GuiSetVisible** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

GuiSlider
- creates on-chart slider control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiSlider(id, x, y, width, height, notifyflags, style = 0)**

RETURNS NUMBER

FUNCTION The function creates on-chart GUI slider control.

Parameters:

- id - unique control ID (must be integer greater than zero)
- x, y - coordinates of top left corner of the control
- width, height - dimensions of the control
- notifyflags - define which user actions cause formula to be notified
- style - defines windows control style (WS_* constants in Windows API) that gets OR-ed with default styles (that are added by AmiBroker automatically and typically are WS_CHILD | WS_VISIBLE | WS_CLIPSIBLINGS | WS_TABSTOP)

Possible notify flags are:

- notifyClicked - when button/toggle/checkbox/radio is clicked
- notifySetFocus - when control gets input focus (via mouse or keyboard)
- notifyKillFocus - when control loses input focus (via mouse or keyboard)
- notifyHitReturn - when ENTER/RETURN key was pressed in edit field
- notifyEditChange - when control value changes (text inside edit field, date inside date/time, slider is moved to new position)
- notifyMouseEnter - when mouse hovers over the control
- notifyMouseLeave - when mouse leaves control area

The function returns **guiNew** when control was actually created (wasn't already present in the chart) or **guiExisting** when control already exists in given chart

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO GuiEdit() function , GuiDateTime() function , GuiButton() function , GuiCheckBox() function GuiEdit() function , GuiRadio() function , GuiToggle() function

References:

The **GuiSlider** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit

More information:

See updated/extended version on-line.

GuiToggle
- create on-chart toggle button control

GUI functions
(AmiBroker 6.30)

SYNTAX **GuiToggle("text", id, x, y, width, height, notifyflags)**

RETURNS NUMBER

FUNCTION The function creates on-chart GUI toggle button control.

Parameters:

- "text" - specifies button text
- id - unique control ID (must be integer greater than zero)
- x, y - coordinates of top left corner of the control
- width, height - dimensions of the control
- notifyflags - define which user actions cause formula to be notified

Possible notify flags are:

- notifyClicked - when button/toggle/checkbox/radio is clicked
- notifySetFocus - when control gets input focus (via mouse or keyboard)
- notifyKillFocus - when control loses input focus (via mouse or keyboard)
- notifyHitReturn - when ENTER/RETURN key was pressed in edit field
- notifyEditChange - when control value changes (text inside edit field, date inside date/time, slider is moved to new position)
- notifyMouseEnter - when mouse hovers over the control
- notifyMouseLeave - when mouse leaves control area

The function returns **guiNew** when control was actually created (wasn't already present in the chart) or **guiExisting** when control already exists in given chart

For more information about using GUI controls see Tutorial: Using on-chart GUI controls

EXAMPLE

SEE ALSO

References:

The **GuiToggle** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

HHV - highest high value

Lowest/Highest

SYNTAX **hhv(ARRAY, *periods*)**

RETURNS ARRAY

FUNCTION Calculates the highest value in the ARRAY over the preceding *periods* (*periods* includes the current day). HHV accepts periods parameter that can be constant as well as time-variant (array).

EXAMPLE The formula "hhv(close, 4)" returns the highest closing price over the preceding four periods; "hhv(high, 8)" returns the highest high price over the preceding eight periods.

SEE ALSO

References:

The **HHV** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- % B of Bollinger Bands With Adaptive Zones
- 10-20 Indicator
- 30 Week Hi Indicator - Calculate
- 52 Week New High-New Low Index
- Adaptave Zones O/B & O/S Oscillator
- Adaptive Price Channel
- Advanced MA system
- ADXVMA
- AFL Example
- AFL Example - Enhanced
- Against all odds
- AJDX system
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- An n bar Reversal Indicator
- Another FIb Level
- Aroon
- AR_Prediction.afl
- ATR Study
- Auto Fib Ext&Retracement
- Auto-Optimization Framework
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Awsome Oscilator
- babaloo chapora
- Brian Wild
- Bull Fear / Bear Fear
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Caleb Lawrence

- CAMSLIM Cup and Handle Pattern AFL
- Candle Stick Demo
- Candlestick Commentary
- Candlestick Commentary Modified
- Candlestick Commentary-modified
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI(20) Divergence Indicator
- CCT Kaleidoscope
- CCT StochasticRSI
- Chaikin Volume Accumulation
- Chandelier Exit
- Chandelier Exit or Advanced Trailing Stop
- Channel/S&R and trendlines
- Compare Sectors against Tickers
- Congestions detection
- Continuous Contract Rollover
- CVR--severe filter
- Dahl Oscillator modified
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Dave Landry PullBack Scan
- Dave Landry Pullbacks
- Demand Index
- Demand Index
- Dinapoli Perferred Stochastic
- Divergences
- Donchian Channel
- Double Smoothed Stochastic from W.Bressert
- Double top detection
- DT Oscillator
- Ed Seykota's TSP: Support and Resistance
- Ehlers Fisher Transform
- ekeko price chart
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's SafeZone Stop
- ElderSafeZoneStopLong
- FastStochK FullStochK-D
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator
- Fisher Relative Vigour Index
- Fre
- Frequency distribution of returns
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Fund Screener
- Gann level plotter
- Gann Swing chart v41212
- Gfx Toolkit
- Gordon Rose

- Harmonic Pattern Detection
- Head & Shoulders Pattern
- Hilbert Sine Wave Support & Resistance
- Ichimoku Chart
- Ichimoku charts
- Ichimoku Kinko Hyo
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Ichimoku with plot mofified to use cloud function
- IchimokuBrianViorelRO
- IFT of RSI - Multiple TimeFrames
- Improved NH-NH scan / indicator
- Index of 30 Wk Highs Vs Lows
- Inter-market Yield Linear Regression Divergence
- Intraday Strength
- JEEVAN'S SRI CHAKRA
- Kagi Chart
- Larry William's Volatility Channels
- Linear Regression Line & Bands
- MACD commentary
- MACD indicator display
- Meu Sistema de Trading - versão 1.0
- mitalpradip
- Modified Head & Shoulder Pattern
- Monthly bar chart
- MS Darvas Box with Exploration
- MultiCycle 1.0
- Multiple Ribbon Demo
- Murrey Math Price Lines
- N-period candlesticks (time compression)
- New HL Scanner
- nikhil
- Non-repaitning Zigzag line
- OBV with Linear Regression
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- Pattern - Rectangle Base Breakout on High Vol
- Pattern_-_Rectangle_Base_Breakout_on_High_Vol 2
- Perceptron
- Peterson
- PF Chart - Close - April 2004
- Pivot Finder
- Pivots And Prices
- Point & figure Chart India Securities
- Polyfit Lines
- prakash
- Prashanth
- Price with Woodies Pivots
- Probability Density & Gaussian Distribution
- PVT Trend Decider
- Rainbow Oscillator
- Rea Time Daily Price Levels

- Regression Analysis Line
- Relative Strength Index
- RSI Double-Bottom
- RSI Pointer
- RSIS
- RUTVOL timing signal with BB Scoring routine
- Sainath Sidgiddi
- Scan New High and New Low
- shailu lunia
- Sony
- Stan Weinstein strategy
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Stochastic %J - KDJ
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic Divergence, negative
- Stochastic Divergences, PDI, NDI
- Stochastic Fast%K and Full
- Stochastic OBV and Price Filter
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stop-loss Indicator bands
- Stops Implementation in AFS
- Stress with SuperSmoother
- Sun&Cloud
- Support and Resistance
- Support and resistance
- Support Resistance levels
- TD Moving Average I
- TD sequential
- TD Sequential
- The Stochastic CCI
- Three Day Balance Point
- Tracking Error
- Tracking Error
- Trailing Stop Loss
- Trend Exploration: Count Number of New Highs
- Trend Following System
- Trend Trigger Factor
- Triangle exploration using P&F Chart
- Triangle search
- Triangle Search Extended
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- ValueChart
- Vertical Horizontal Filter
- Vertical Horizontal Filter (VHF)
- visual turtle trading system
- Volatility System
- Weekly chart
- Weinberg's The Range Indicator

- William's Alligator System II
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- Zig Zag
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

HHVBars
- bars since highest high

Lowest/Highest

SYNTAX	**HHVBars(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates the number of periods that have passed since the ARRAY reached its *periods* period peak. HHVBars accepts periods parameter that can be constant as well as time-variant (array).
EXAMPLE	The formula "hhvbars(close, 30)" returns the number of periods that have passed since the closing price reached its 30-period peak.
SEE ALSO	

References:

The **HHVBars** function is used in the following formulas in AFL on-line library:

- Advanced MA system
- Aroon Indicators
- Aroon The Advisor
- babaloo chapora
- CAMSLIM Cup and Handle Pattern AFL
- CCI(20) Divergence Indicator
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Divergences
- ekeko price chart
- Fre
- Fund Screener
- Gordon Rose
- Halftrend
- Pivot Finder
- shailu lunia
- Stochastic Divergence, negative
- Stochastic Divergences, PDI, NDI
- Triangle search
- Triangle Search Extended
- Vivek Jain

More information:

See updated/extended version on-line.

Highest - highest value

Lowest/Highest

SYNTAX	**Highest(ARRAY)**
RETURNS	ARRAY
FUNCTION	Calculates the highest value in the ARRAY since the first day/bar present in the database.
EXAMPLE	The formula highest(mfi(14)) returns the highest Money Flow Index value; highest (close) returns the highest closing price.
SEE ALSO	HHV() function , LOWEST() function , LLV() function

References:

The **Highest** function is used in the following formulas in AFL on-line library:

- Alpha and Beta and R_Squared Indicator
- AR_Prediction.afl
- Auto-Optimization Framework
- Automatic Trend-line
- babaloo chapora
- Candle Stick Analysis
- Candle Stick Demo
- CVR--severe filter
- Darvas Johndeo Research
- Double top detection
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Gordon Rose
- Intraday Volume EMA
- Market Profile
- nth (1 - 8) Order Polynomial Fit
- Perceptron
- Pivot Finder
- shailu lunia
- Triangle exploration using P&F Chart
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

HighestBars
- bars since highest value

Lowest/Highest

SYNTAX	**HighestBars(ARRAY)**
RETURNS	ARRAY
FUNCTION	Calculates the number of periods that have passed since the ARRAY s highest value.
EXAMPLE	The formula "highestbars(close)" returns the number of periods that have passed since the closing price reached its highest peak.
SEE ALSO	

References:

The **HighestBars** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

HighestSince
- highest value since condition met

Lowest/Highest
(AmiBroker 3.40)

SYNTAX **HighestSince(EXPRESSION, ARRAY, *Nth* = 1)**

RETURNS ARRAY

FUNCTION Returns the highest ARRAY value since EXPRESSION was true on the Nth most recent occurrence.

EXAMPLE highestsince(Cross(macd(), 0), Close, 1) returns the highest close price since macd() has crossed above zero.

SEE ALSO

References:

The **HighestSince** function is used in the following formulas in AFL on-line library:

- AJDX system
- Gann Swing Chart
- Gann Swing Charts in 3 modes with text
- Harmonic Pattern Detection
- High Low Detection code
- Inverted Plotted Volume Overlay Indicator
- Last Five Trades Result Dashboard – AFL code
- Market Profile
- MO_CrashZone
- RSI of Weekly Price Array
- Stochastic of Weekly Price Array
- Time Frame Weekly Bars
- Visible Min and Max Value Demo
- Visualization of stoploses and profit in chart
- Williams Alligator system

More information:

See updated/extended version on-line.

HighestSinceBars - bars since highest value since condition met

Lowest/Highest
(AmiBroker 3.40)

SYNTAX	**HighestSinceBars(EXPRESSION, ARRAY, *Nth* = 1)**
RETURNS	ARRAY
FUNCTION	Returns the number of bars that have passed since highest ARRAY value since EXPRESSION was true on the Nth most recent occurrence.
EXAMPLE	highestsincebars(Cross(macd(), 0), Close, 1) returns the number of bars passed since the highest close price was detected from the time when macd() has crossed above zero.
SEE ALSO	

References:

The **HighestSinceBars** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

HighestVisibleValue - get the highest value within visible chart area

Indicators
(AmiBroker 5.30)

SYNTAX	**HighestVisibleValue(array)**
RETURNS	NUMBER
FUNCTION	The function calculates single value (not array) representing highest value of given array within VISIBLE range (on chart). Should be applied only in indicators as only indicators have concept of "visible" bars. The function will return Null value if no visible bars are present. The function is equivalent to the following coding:

```
function HighestVisibleValueEquivalent( array )
{
 bv = Status("barvisible");
 Hh = -1e8;
 for( i = 0; i < BarCount; i++ )
 {
  if( bv[ i ] AND array[ i ] > Hh ) Hh = array[ i ];
 }
 return hh;
}
```

EXAMPLE	
SEE ALSO	LowestVisibleValue() function

References:

The **HighestVisibleValue** function is used in the following formulas in AFL on-line library:

- Auto Fib Ext&Retracement
- Day Bar No
- Fibonacci Calculations & Speed Resistance
- Fre
- New HL Scanner

More information:

See updated/extended version on-line.

HMA - Hull Moving Average

Moving averages, summation
(AmiBroker 5.40)

SYNTAX **HMA(array, range)**

RETURNS ARRAY

FUNCTION Implements Hull Moving Average. It is functionally equivalent to the following code:

```
function HMA_AFL( array, period )
{
 fast = WMA( array, period / 2 );
 slow = WMA( array, period );

 return WMA( 2 * fast - slow, sqrt( period ) );
}
```

EXAMPLE

SEE ALSO MA() function , WMA() function

References:

The **HMA** function is used in the following formulas in AFL on-line library:

- MAVG

More information:

See updated/extended version on-line.

Hold
- hold the alert signal

Trading system toolbox

SYNTAX	**Hold(EXPRESSION, *periods*)**
RETURNS	ARRAY
FUNCTION	Holds a "true" result of EXPRESSION for the specified number of *periods*. This true result is held true over the number of periods specified even if a "false" result is generated.
EXAMPLE	hold(cross(rsi(14),70),5)
SEE ALSO	

References:

The **Hold** function is used in the following formulas in AFL on-line library:

- Peterson
- TD sequential
- Williams Alligator system

More information:

See updated/extended version on-line.

Hour
- get current bar's hour

Date/Time
(AmiBroker 40)

SYNTAX	**Hour()**
RETURNS	ARRAY
FUNCTION	Retrieves current bar's hour
EXAMPLE	Hour()*10000 + Minute() * 100 + Second()
SEE ALSO	Second(), Minute(), TimeNum() MilliSec() function

References:

The **Hour** function is used in the following formulas in AFL on-line library:

- Backup Data of 1min Interval
- Buyer Seller Force
- Export EOD or Intraday to .csv file
- Export Intraday Data
- For Auto Trading Setup
- High Low Detection code
- How to add IB Option Symbols
- Luna Phase
- New HL Scanner
- White Theme

More information:

See updated/extended version on-line.

IcbID
- get ICB category information

Information / Categories
(AmiBroker 5.60)

SYNTAX **IcbID(mode)**

RETURNS STRING

FUNCTION The function gets information about current symbol ICB category.

- mode = 0 - returns string containing ICB code alone (such as "9530")
- mode = 1 - returns string containing ICB category name (such as "Software & Computer Services")
- mode = 2 - returns string containing both code and name (such as "9530 Software & Computer Services")

EXAMPLE

```
printf( "ICB(0) = " + IcbID( 0 ) );
printf( "ICB(1) = " + IcbID( 1 ) );
printf( "ICB(2) = " + IcbID( 2 ) );

for( i = 10; i < 90; i+= 1 )
{
 ICB_code = StrFormat("%02.0f", i );
 printf("In ICB '"+ ICB_code + "' = %gn", InIcb( ICB_code ) );
}
```

SEE ALSO InICB() function

References:

The **IcbID** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

IIf - immediate IF function

Trading system toolbox

SYNTAX **IIf(EXPRESSION, TRUE_PART, FALSE_PART)**

RETURNS ARRAY or NUMBER

FUNCTION "Immediate-IF" - a conditional function that returns the value of the second parameter (TRUE_PART) if the conditional expression defined by the first parameter (EXPRESSION) is true; otherwise, the value of third parameter is returned (FALSE_PART). Please note that IIF is a function - so the result of evaluation is returned by that function and should be assigned to some variable. IIf always evaluates both TRUE_PART and FALSE_PART, even though it returns only one of them. Because of this, you should watch for undesirable side effects. The following example shows one common error made with IIF function: IIF(condition, result = 7, result = 9); // THIS IS WRONG Correct usage is: result = IIF(condition, 7, 9); /* 7 or 9 is *returned* and assigned to a variable depending on condition */

When all arguments are scalars (numbers) then resulting value is also a scalar (number). When any of arguments is an array then function returns an array.

When working on arrays, Iif function evaluates all bars, condition is checked on each bar and returned value for given bar is choosen appropriately on bar by bar basis. The following code shows how array operation of Iif is implemented internally. Take a look also at Understanding AFL chapter of the manual.

```
function IIF_AFL( condition, inputA, inputB )
{
     result = Null;

    for( bar = 0; bar < BarCount; bar++ )
     {
       if( condition[ bar ] )
           result[ bar ] = inputA[ bar ];
       else
           result[ bar ] = inputB[ bar ];
     }

    return result;
}
```

EXAMPLE

```
// The formula below
// will assign positive Volume values to the result variable on days
when
// MACD was below its Signal line, AND negative Volume values on
theother
// days.

result = IIf( MACD() < Signal(), Volume, -Volume );

// The formula below
// will assign colorRed to the dynamic_color variable on days when
```

```
        // Close < Open and color Green otherwise


        dynamic_color = IIf( Close < Open, colorRed, colorGreen );
        Plot( Volume, "Color volume", dynamic_color, styleHistogram |
        styleThick );
```

SEE ALSO

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2003-06-16 03:04:48	IIF can be re-implemented using new if-else flow control statements. The code below shows this and explains what IIF in fact does internally.
	
	function _IIF(ConditionArray, TrueArray, FalseArray)
	{
	for(i = 0; i < BarCount; i++)
	{
	if(ConditionArray[i])
	{
	result[i] = TrueArray[i];
	}
	else
	{
	result[i] = FalseArray[i];
	}
	}
	}

Tomasz Janeczko tj --at-- amibroker.com 2003-07-28 09:24:10	If you want to operate on STRINGS use WriteIF function: result = WriteIF(condition, "Text 1", "Text 2"); (note that this function returns single string, depending on 'selected value' of condition).

References:

The **IIf** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- % B of Bollinger Bands With Adaptive Zones
- 2 Timeframes Candlestick Bar Chart
- 3 Price Break
- 3 ways to use RMI in one script
- 3TF Candlestick Bar Chart
- Abhimanyu
- AC+ acceleration
- accum/dist mov avg crossover SAR
- Adaptave Zones O/B & O/S Oscillator
- Adaptive Centre of Gravity
- Adaptive Cyber Cycle
- Adaptive Relative Vigour Index
- Add SL/TGT other params to any strategy

- Advanced MA system
- Advanced Trend Lines with S & R
- Advisory NRx price chart display.
- ADX Indicator - Colored
- ADXVMA
- AFL-Excel
- AFL_Glossary_Converter
- Against all odds
- Alert Output As Quick Rewiev
- ALJEHANI
- AllinOneAlerts - Module
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Analytic RSI formula
- Andrews Pitchfork
- Andrews PitchforkV3.3
- AO+ Momentum indicator
- AO+Momentum
- Aroon
- Aroon The Advisor
- AR_Prediction.afl
- ATR Study
- Auto Fib Ext&Retracement
- Auto Trade Step by Step
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trend-line
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- Average Dollar Price Volatility Exploration
- Average Price Crossover
- Awsome Oscilator
- babaloo chapora
- Bad Tick Trim on 5 sec database
- Balance of Power
- balance of power
- Basket Trading System T101
- BB squeeze
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bollinger Band Gap
- Bollinger Band Squeeze
- Bollinger Fibonacci Bands
- Bottom Fisher Exploration
- Brian Wild
- Bull/Bear Volume
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004

- Caleb Lawrence
- CAMSLIM Cup and Handle Pattern AFL
- Candle Stick Analysis
- Candle Stick Demo
- Candlestick Commentary
- Candlestick Commentary Modified
- Candlestick Commentary-modified
- Candlestick Volume Bars with Moving Average
- CandleStochastics
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- CCI(20) Divergence Indicator
- Centre of Gravity
- Chande Momentum Oscillator
- Chande's Trend Score
- Chandelier Exit or Advanced Trailing Stop
- changing period moving avarage
- channel indicator
- Channel/S&R and trendlines
- CoinToss ver 1
- Cole
- Color Coded Short Term Reversal Signals
- Color Display.afl
- Color MACD Histogram Changes
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- com-out
- COMBO
- Compare Sectors against Tickers
- Congestions detection
- Connors TPS
- ConnorsRSI
- Continuous Contract Rollover
- Coppock Histogram
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Coral Trend Indicator
- crMathLib
- CVR--severe filter
- Cyber Cycle
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Wisestocktrader
- Dave Landry PullBack Scan
- Dave Landry Pullbacks
- Day Bar No
- Days to Third Friday

- De Mark's Range Projection
- DEBAJ
- Demand Index
- Demand Index
- DeMarker
- DiNapolis 3x Displaced Moving Averages
- Distance Coefficient Ehlers Filter
- Divergences
- DMI Spread Index
- Double Super Trend System
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- Effective Swing Indicator
- ekeko price chart
- EKEKO SAR-MF
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Ray Oscillator with MA
- Elder safe Zone Long + short
- Elder's Market Thermometer
- Elder's SafeZone Stop
- ElderSafeZoneStopLong
- ElderSafeZoneStopShort
- Elliott Wave Oscillator
- End Of Year Trading
- Expiry day/days - Last thursday of month
- Expiry Thursday for Indian markets
- Fast Refreshed KAGI Swing Charts (Price Swing)
- FastStochK FullStochK-D
- Fib CMO
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator
- Fisher Relative Vigour Index
- Follow the Leader
- For Auto Trading Setup
- Fre
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Fund Screener
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann HiLo Indicator and System
- Gann level plotter
- Gann Swing Chart
- Gann Swing chart v41212
- Gann Swing Charts in 3 modes with text

- garythompson
- garythompson
- Geometric Mean of Volume
- Gfx Toolkit
- GFX ToolTip
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- half-automated Trading System
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Head & Shoulders Pattern
- Heatmap V1
- Heikin Ashi Delta
- HH-LL-PriceBar
- High Low Detection code
- Hilbert Study
- How to add IB Option Symbols
- Hull Moving Average
- Hurst "Like" DE
- Hurst Constant
- IBD relative strength database Viewer
- Ichimoku Kinko Hyo
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- IchimokuBrianViorelRO
- IFT of RSI - Multiple TimeFrames
- Indicator Explorer (ZigZag)
- Instantaneous Trend Line
- Intraday Average Volume
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- IntraDay Open Marker
- Intraday Range and Periods Framer
- Intraday Trend Break System
- Intraday Volume EMA
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Jesse Livermore Secret Market Key
- Kagi Chart
- Kiss and Touch with the Modified True Range
- Lagging MA-Xover
- Larry William's Volatility Channels
- Last Five Trades Result Dashboard – AFL code
- lastNDaysBeforeDate
- Least Squares Channel Indicator
- Linear Candle
- Linear Regression Line w/ Std Deviation Channels
- Luna Phase
- Lunar Phases - original
- LunarPhase

- MA Difference 20 Period
- MACD BB Indicator
- MACD indicator display
- MACD-V
- Main price chart with Rainbow & SAR
- Market Facilitation Index VS Volume
- Market Profile
- MAVG
- MCDX (Multi Color Dragon Extended)
- MDYtoXLSerialDays and XLSerialDaysToDateNum
- Meu Sistema de Trading - versão 1.0
- MFE and MAE and plot trades as indicator
- mfimacd
- mitalpradip
- Modified Head & Shoulder Pattern
- Modified-DVI
- Momentum Volume Price (MVP) Indicator
- Monthly bar chart
- Moving Average "Crash" Test
- Moving Averages NoX
- MS Darvas Box with Exploration
- Multi-color Volume At Price (VAP)
- MultiCycle 1.0
- Multiple Ribbon Demo
- N Line Break Chart
- N-period candlesticks (time compression)
- Nadaraya-Watson Envelope
- NASDAQ 100 Volume
- New HL Scanner
- Next Date Format
- Nick
- nikhil
- Non-repaitning Zigzag line
- Nonlinear Ehlers Filter
- Noor_Doodie
- nth (1 - 8) Order Polynomial Fit
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- Ord Volume
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- Parametric Chande Trendscore
- pattenz
- Percentage Price Oscillator
- Perceptron
- Performance Check
- Periodically ReBalance a BUY & HOLD Portfolio
- Peterson
- PF Chart - Close - April 2004
- Pivot End Of Day Trading System
- Pivot Finder
- Pivots for Intraday Forex Charts

- Plot visual stop / target ratio.
- Point & figure Chart India Securities
- Polarized Fractal Efficiency
- Polyfit Lines
- Position Sizer vers2, stocks and CFDs
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- Prashanth
- Price Chart - Fundamental
- Price with Woodies Pivots
- Price-Volume Rank
- Prior Daily OHLC
- PVT Trend Decider
- QP2 Float Analysis
- R-Squared
- Rainbow Oscillator
- Range Expansion Index
- Range Filter - Trading Strategy
- Ranking Ticker WatchList
- Raw ADX
- Rea Time Daily Price Levels
- Regression Analysis Line
- Relative Momentum Index (RMI)
- Relative Strength Multichart of up to 10 tickers
- Relative Vigour Index
- Relative Vigour Index
- Relative Volume
- Renko Chart
- RI - Auto Trading System
- ROC of MACD Weekly
- RSI of Weekly Price Array
- RSI styleClipMinMax
- RSI Trendlines and Wedges
- RSIS
- RUTVOL timing signal with BB Scoring routine
- Sainath Sidgiddi
- Scale Out: Futures
- Schiff Lines
- SectorRSI
- shailu lunia
- Spearman Rank Correlation Coefficient
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Standard Error Bands (Native AFL)
- STD_STK Multi
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic Divergence, negative
- Stochastic Divergence, positive
- Stochastic Divergences, PDI, NDI

- Stochastic Fast%K and Full
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stochastics Trendlines
- Stress with SuperSmoother
- Sun&Cloud
- SUPER PIVOT POINTS
- Super Trend Indicator
- Super Trend Indicator
- Support and Resistance
- Support and resistance
- suresh
- TAPE READING
- TD Moving Average 2
- TD Moving Average I
- TD REI
- TD sequential
- TD Sequential
- The Fibonaccian behavior
- The Mean RSIt
- The Mean RSIt (variations)
- The Saturation Indicator D_sat
- Three Day Balance Point
- Three Pole Butterworth
- Time Left in Bar
- Time segment value
- tomy_frenchy
- Tracking Error
- Trades per second indicator
- Trading ATR 10-1
- Trailing Stop Loss
- Trend Continuation Factor
- Trend Detection
- Trend exploration with multiple timeframes
- Trend Exploration: Count Number of New Highs
- Trend Exploration: Slope Moving Average
- Trend Following System
- TRENDAdvisor
- Trending Ribbon
- TrendingRibbonArrowsADX
- Triangle exploration using P&F Chart
- Triangular Moving Average
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- Trix Bars number
- TSV
- TTM Squeeze
- Twiggs Money Flow
- Twiggs money flow weekly
- TWS auto-export Executions-file parser
- TWS trade plotter
- Updated Renko Chart

- Using From and To dates from Auto Analysis in Code
- ValueChart
- Vic Huebner
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- Visualization of stoploses and profit in chart
- Vivek Jain
- Volatility
- Volatility System
- Volume - compared with Moving Avg (100%)
- Volume Color with Dynamic Limit
- Volume Divided Histogram
- Volume Occilator
- Volume Oscillator
- Volume Spread for VSA
- Volume Zone Oscillator
- VolumeDivAvgV3_Study_BrS
- VSTOP (2)
- VSTOP (3)
- VWAP versus Average Price
- Weekly chart
- Weekly Trend in Daily Graph
- Weinberg's The Range Indicator
- White Theme
- William's Alligator System II
- Williams Alligator system
- WILSON RELATIVE PRICE CHANNEL
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- Woodies CCI
- Zig Zag
- Zig Zag Indicator with Valid Entry and Exit Points
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

IIR
- infinite impulse response filter

Moving averages, summation
(AmiBroker 60)

SYNTAX **IIR(input, b0 = 1, a1 = 0, b1 = 0, a2 = 0, b2 = 0, a3 = 0, b3 = 0, a4 = 0, b4 = 0)**

RETURNS ARRAY

FUNCTION The function implements fast 4th-order infinite impulse response filter.

Analytically it is:

y[n] = b0 * x[n] + b1 * x[n - 1] + b2 * x[n - 2] + b3 * x[n - 3] + b4 * x[n - 4] + a1 * y[n - 1] + a2 * y[n - 2] + a3 * y[n - 4] + a4 * y[n -4];

AFL equivalent:

```
y = x; // init so no glitches at the beginning appear
for( n = 4; n < BarCount; n++ )
{
y[ n ] = b0 * x[ n ] +
            b1 * x[ n - 1 ] +
            b2 * x[ n - 2 ] +
            b3 * x[ n - 3 ] +
            b4 * x[ n - 4 ] +
            a1 * y[ n - 1 ] +
            a2 * y[ n - 2 ] +
            a3 * y[ n - 4 ] +
            a4 * y[ n - 4 ];
}
```

Filters of orders 3 and 2 can be implemented by leaving unneeded arguments at default value of zero.

Coefficients b0, b1, b2, b3, b4 multiply the input signal x[n] and are referred to as the feedforward coefficients. Coefficients a1, a2, a3, a4 multiply the output signal y[n] and are referred to as the feedback coefficients. Pay careful attention to the sign of the feedback coefficients. Some design tools flip the sign of the feedback coefficients. In this case the feedback coefficients must be negated.

This convention is used so feedback coefficients work the same as in AMA2 in case of first order filter, so

IIR(array, factor, 1-factor)

is the same as

AMA2(array, factor, 1-factor)

(with very minor difference is that IIR uses internally double precision arithmetic while AMA2 uses single precision)

EXAMPLE

```
// simple ema:
factor = 2/(period+1);
y = IIR( input, factor, 1- factor );

// wilders:
factor = 1/period
z = IIR( input, factor, 1-factor );

// Ehlers Supersmoother

Periods = 10;

c1 = 1.41421 * 3.14159 / Periods;
c2 = 2.71828^-c1;
a1 = 2 * c2 * cos( c1 );
a2 = -c2^2;
b0 = (1 - a1 - a2)/2;
b1 = b0;

x = IIR( Close, b0, a1, b1, a2 );
Plot( x, "Super Smoother", colorRed );
```

SEE ALSO AMA() function , AMA2() function

References:

The **IIR** function is used in the following formulas in AFL on-line library:

- Ehlers Hilbert Transformer Indicator
- Even Better Sinewave Indicator
- Forward/Reverse EMA by John Ehlers
- Voss Predictive Filter (A Peek Into the Future)

More information:

See updated/extended version on-line.

IndustryID
- get industry ID / name

Information / Categories
(AmiBroker 3.80)

SYNTAX	**IndustryID(mode = 0)**
RETURNS	NUMBER/STRING
FUNCTION	Retrieves current stock industry ID/name When mode = 0 (the default value) this function returns numerical industry ID (consecutive industry number) When mode = 1 this function returns name of the industry.

EXAMPLE	Filter = IndustryID() == 7 OR IndustryID() == 9;
	AddTextColumn(IndustryID(1), "Industry name");

SEE ALSO	

References:

The **IndustryID** function is used in the following formulas in AFL on-line library:

- Dave Landry PullBack Scan
- Elder Triple Screen Trading System
- qp2 industry charts as a panel in the stocks chart

More information:

See updated/extended version on-line.

InGICS
- test GICS membership

Information / Categories
(AmiBroker 5.40)

SYNTAX **InGICS("gics_code")**

RETURNS NUMBER (0 or 1)

FUNCTION The function performs yes/no test if current symbol belongs to given GICS category for example if GICS set for the symbol is 15103020 InGics("15"), InGICS("1510"), InGics("151030") and InGics("15103020") will ALL return true but all others (like InGics("20")) will return false.

EXAMPLE

```
printf( "GICS(0) = " + GicsID( 0 ) );
printf( "GICS(1) = " + GicsID( 1 ) );
printf( "GICS(2) = " + GicsID( 2 ) );

for( i = 10; i < 90; i+= 1 )
{
 gics_code = StrFormat("%02.0f", i );
 printf("In Gics '"+ gics_code + "' = %g\n", InGics( gics_code ) );
}
```

SEE ALSO GicsID() function , InWatchList() function , InWatchListName() function

References:

The **InGICS** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

InICB
- test ICB membership

Information / Categories
(AmiBroker 5.60)

SYNTAX **InICB("icb_code")**

RETURNS NUMBER (0 or 1)

FUNCTION The function performs yes/no test if current symbol belongs to given ICB category for example if ICB set for the symbol is 9537, InICB("9000"), InICB("9500"), InICB("9530") and InICB("9537") will ALL return true but all others (like InICB("5000")) will return false.

EXAMPLE

```
printf( "ICB(0) = " + IcbID( 0 ) );
printf( "ICB(1) = " + IcbID( 1 ) );
printf( "ICB(2) = " + IcbID( 2 ) );

for( i = 10; i < 90; i+= 1 )
{
 ICB_code = StrFormat("%02.0f", i );
 printf("In ICB '"+ ICB_code + "' = %gn", InIcb( ICB_code ) );
}
```

SEE ALSO InICB() function

References:

The **InICB** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Inside - inside day

Basic price pattern detection

SYNTAX	**Inside()**
RETURNS	ARRAY
FUNCTION	Gives a "1" or "true" when an inside day occurs.Gives "0" otherwise. An inside day occurs when today's high is less than yesterday's high and today's low is greater than yesterday's low.
EXAMPLE	
SEE ALSO	

References:

The **Inside** function is used in the following formulas in AFL on-line library:

- Advisory NRx price chart display.
- Gann Five Day pullback
- Gann Swing Chart
- Gfx Toolkit
- NRx Exploration
- Vic Huebner

More information:

See updated/extended version on-line.

Int
- integer part

Math functions

SYNTAX	**Int(NUMBER)**
	int(ARRAY)

RETURNS	NUMBER ARRAY
FUNCTION	Removes the fractional portion of NUMBER or ARRAY and returns the integer part.
EXAMPLE	The formula "int(10.7)" returns 10; the formula "int(-19.8)" returns -19.
SEE ALSO	The ceil() function; the floor() function; the frac() function.

References:

The **Int** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- A simple AFL Revision Control System
- Adaptive Centre of Gravity
- Adaptive Relative Vigour Index
- AFL Timing functions
- AutoTrade using an Exploration
- Bad Tick Trim on 5 sec database
- Baseline Relative Performance Watchlist charts V2
- Bullish Percent Index 2 files combined
- Candle Stick Demo
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- Color Display.afl
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- Cybernertic Hilbert Sine Wave
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Dominant Cycle Phase
- Dynamtic Momentum Index
- elliott wave manual labelling
- Fibonacci Calculations & Speed Resistance
- Gartley 222 Pattern Indicator
- Gfx Toolkit
- Heatmap V1
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- Hull Moving Average
- Hull with DEMA
- Hurst "Like" DE
- Hurst Constant
- John Ehler
- Kagi Chart

- Log Time Scale
- Lunar Phases - original
- LunarPhase
- MDYtoXLSerialDays and XLSerialDaysToDateNum
- Modified Momentum Finder DDT-NB
- Modified-DVI
- Multiple Ribbon Demo
- Probability Density & Gaussian Distribution
- Profit Table (Color Coded)
- Randomize()
- Signal to Noise
- Sine Wave Indicator
- Smoothed Adaptive Momentum
- Standard Error Bands
- Support and resistance
- TAPE READING
- Time Left to Current Bar
- Triangle exploration using P&F Chart
- VAMA
- Visi-Trade
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

InternetClose - close Internet file handle

File Input/Output functions
(AmiBroker 6.20)

SYNTAX **InternetClose(handle)**

RETURNS NOTHING

FUNCTION The function closes Internet file handle that was open by InternetOpenURL

EXAMPLE

```
ih = InternetOpenURL(
"https://www.quandl.com/api/v3/datasets/SEC/AAPL_SALESREVENUENET_Q.csv?api_ke
);
printf( "AAPL Revenue: " );
if( ih )
{
     while( ( str = InternetReadString( ih ) ) != "" )
     {
         printf( "%s", str );
     }
     InternetClose( ih );
}
```

SEE ALSO

References:

The **InternetClose** function is used in the following formulas in AFL on-line library:

- AllinOneAlerts - Module
- Get Moneycontrol News Snippets into Amiboker
- Send Alerts from Amibroker to Telgram

More information:

See updated/extended version on-line.

InternetGetStatusCode - returns HTTP status code of last Internet call

File Input/Output functions
(AmiBroker 6.40)

SYNTAX **InternetGetStatusCode**

RETURNS NUMBER

FUNCTION The function returns HTTP status code of last InternetOpenURL or InternetPostRequest call.

HTTP status codes are listed here: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

EXAMPLE

```
// NON existing page (should result in status code 404)
ih = InternetOpenUrl("http://www.amibroker.com/index3434.html" );

if( ih )
{
   printf("HTTP status code: %g", InternetGetStatusCode( ih ) );

   InternetClose( ih );
}
else
{
   printf("Internet connection can not be open because: %s",
GetLastOSError() );
}
```

SEE ALSO InternetOpenURL() function , InternetPostRequest() function

References:

The **InternetGetStatusCode** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

InternetOpenURL - opens Internet web resource (URL)

File Input/Output functions
(AmiBroker 6.20)

SYNTAX **InternetOpenURL("http://url_to_your_web_resource")**

RETURNS HANDLE

FUNCTION The function opens specified URL and returns a special file handle to be used by InternetReadString/InternetClose functions. It allows to open any web page or REST API service.

EXAMPLE

```
ih = InternetOpenURL(
"https://www.quandl.com/api/v3/datasets/SEC/AAPL_SALESREVENUENET_Q.csv?api_ke
);
printf( "AAPL Revenue: " );
if( ih )
{
     while( ( str = InternetReadString( ih ) ) != "" )
     {
         printf( "%s", str );
     }
     InternetClose( ih );
}
```

SEE ALSO InternetClose() function

References:

The **InternetOpenURL** function is used in the following formulas in AFL on-line library:

- AllinOneAlerts - Module
- Get Moneycontrol News Snippets into Amiboker
- Send Alerts from Amibroker to Telgram

More information:

See updated/extended version on-line.

InternetPostRequest - send HTTP Post request to Internet web resource (URL)

File Input/Output functions
(AmiBroker 6.30)

SYNTAX **InternetPostRequest("http://url_to_your_web_resource", data, flags = 0)**

RETURNS HANDLE

FUNCTION The function sends URL-encoded data via HTTP Post request to the specified URL and returns a special file handle to be used by InternetReadString/InternetClose functions. It allows to open any web page or REST API service

EXAMPLE

```
ih = InternetPostRequest(
"http://server_name_here.com/testpost.php?param3=7&m4=8", /*POST
DATA*/ "param1=9&m2=12" );

if( ih )
{
    while( ( str = InternetReadString( ih ) ) != "" )
    {
       printf( "%s", str );
    }

    InternetClose( ih );
}


// testing code (server side ) in PHP:


echo "Post variables: \r\n";

foreach ($_POST as $key => $value){
   echo "{$key} = {$value} \r\n";
}

echo "Get variables: \r\n";

foreach ($_GET as $key => $value){
   echo "{$key} = {$value} \r\n";
}

?>
```

SEE ALSO InternetOpenURL() function , InternetClose() function , InternetReadString() function

References:

The **InternetPostRequest** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

InternetReadString - read a string from Internet resource

File Input/Output functions
(AmiBroker 6.20)

SYNTAX **InternetReadString(handle)**

RETURNS STRING

FUNCTION The function reads a string (line) from internet resource

NOTES:

- If End-Of-File has been reached, the function returns empty string ("")
- If End-of-File has NOT been reached, the function always returns non-empty string because eac line read by this function ends with new line character " ", so if server has sent empty line it will b read as " "
- You can trim new line characters using StrTrim

Internet* functions open wide area of applications including:

1. querying web APIs for extra data
2. using web/rest APIs for communication (sending messages/alerts to Twitter, SMS gateways, et

EXAMPLE

```
ih = InternetOpenURL(
"https://www.quandl.com/api/v3/datasets/SEC/AAPL_SALESREVENUENET_Q.csv?api_ke
);
printf( "AAPL Revenue: " );
if( ih )
{
     while( ( str = InternetReadString( ih ) ) != "" )
     {
         printf( "%s", str );
     }
     InternetClose( ih );
}
```

SEE ALSO

References:

The **InternetReadString** function is used in the following formulas in AFL on-line library:

- Get Moneycontrol News Snippets into Amiboker

More information:

See updated/extended version on-line.

InternetSetAgent - set agent string for Internet function

File Input/Output functions
(AmiBroker 6.30)

SYNTAX	**InternetSetAgent("agent_string")**
RETURNS	NOTHING
FUNCTION	The function AFL sets the user agent to be used when opening web pages via InternetOpenURL. By default user agent is "AmiBroker" but some internet sites may expect (or even only work with) specific "popular" agent strings of common browsers such as for example "Mozilla/5.0 (Windows NT x.y; rv:10.0) Gecko/20100101 Firefox/10.0". More information about user agent strings can be found at https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent/Firefox
EXAMPLE	
SEE ALSO	InternetOpenURL() function

References:

The **InternetSetAgent** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

InternetSetHeaders
- set custom HTTP headers for subsequent web requests

File Input/Output functions
(AmiBroker 6.40)

SYNTAX **InternetSetHeaders("headers")**

RETURNS NOTHING

FUNCTION Set custom HTTP headers for subsequent web requests (via subsequent InternetOpenURL or InternetPostRequest calls)

Note that InternetPostRequest will automatically add "Content-Type: application/x-www-form-urlencoded" header unless user specifies their own Content-Type in the InternetSetHeaders call

EXAMPLE

```
// For example to enable GZIP compression you need to use:

INTERNET_OPTION_HTTP_DECODING = 65;
InternetSetOption( INTERNET_OPTION_HTTP_DECODING, 1 );

InternetSetHeaders("Accept-Encoding: gzip, deflate");
```

SEE ALSO InternetOpenURL() function

References:

The **InternetSetHeaders** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

InternetSetOption - set HTTP option for internet session

File Input/Output functions
(AmiBroker 6.40)

SYNTAX **InternetSetOption(option, value)**

RETURNS NOTHING

FUNCTION Set the option for Internet session. Available Internet options are listed in Windows SDK docs: https://docs.microsoft.com/en-us/windows/win32/wininet/option-flags

EXAMPLE

```
// how to request and handle GZIP compressed HTTP

INTERNET_OPTION_HTTP_DECODING = 65;

InternetSetOption( INTERNET_OPTION_HTTP_DECODING, 1 );

InternetSetHeaders("Accept-Encoding: gzip, deflate");

ih = InternetOpenURL("http://www.amibroker.com/news.html" );

if( ih )
{
   while( ( text = InternetReadString( ih ) ) != "" )
   {
      printf( "%s", text );
   }

   InternetClose(ih);
}
```

SEE ALSO InternetOpenURL() function , InternetPostRequest() function

References:

The **InternetSetOption** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Interval
- get bar interval (in seconds)

Date/Time
(AmiBroker 4.10)

SYNTAX **Interval(format = 0)**

RETURNS NUMBER

FUNCTION Interval() function returns bar interval.

Possible formats:

- format = 0 - returns bar interval in seconds
- format = 1 - as above plus TICK bar intervals are returned with negative sign so Interval() function applied to 10 tick chart will return -10
- format = 2 - returns STRING with name of interval such as "Weekly/Monthly/Daily/Hourly/15-minute/5-tick"

Example time intervals in seconds:
tick bars = 0
5 sec bars = 5
1 min bars = 60 (inMinute constant)
hourly bars = 3600
daily bars = 86400 (inDaily constant)
weekly bars = 432001 (inWeekly constant)
monthly bars = 2160001 (inMonthly constant)

Caveat: it is bad idea to use text comparision like this if(Interval(2) == "Weekly") { }. Do NOT do that, because as soon as localized version of AmiBroker arrives, your code will be broken as names will be translated. Use NUMERIC comparison, such as: if(Interval() == inWeekly) { } instead.

EXAMPLE "Interval in seconds " + WriteVal(Interval());

SEE ALSO

References:

The **Interval** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- AFL Timing functions
- AllinOneAlerts - Module
- Alternative ZIG type function, multi TF
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Backup Data of 1min Interval
- Caleb Lawrence
- candlestick chart for Volume/RSI/OBV
- CCI 14 DrBobStyle
- CCI 50 DrBob Style

- CCI Woodies Style
- channel indicator
- Channel/S&R and trendlines
- Chart Zoom
- Commodity Selection Index (CSI)
- Continuous Contract Rollover
- DPO with shading
- ekeko price chart
- elliott wave manual labelling
- Export EOD or Intraday to .csv file
- Fibonacci Internal and External Retracements
- For Auto Trading Setup
- Gfx Toolkit
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Heatmap V1
- Heinkin-Ashi
- High Low Detection code
- How to add IB Option Symbols
- IBD relative strength database Viewer
- Improved NH-NH scan / indicator
- Intraday Average Volume
- Inverted Plotted Volume Overlay Indicator
- Kelly criterion
- Last Five Trades Result Dashboard – AFL code
- Least Squares Channel Indicator
- Market Breadth Chart-In-Chart
- MFE and MAE and plot trades as indicator
- Modified Head & Shoulder Pattern
- Nadaraya-Watson Envelope
- New HL Scanner
- pattenz
- Price with Woodies Pivots
- Range Filter - Trading Strategy
- Rea Time Daily Price Levels
- Super Trend Indicator
- suresh
- TAPE READING
- Time Left in Bar
- Time Left to Current Bar
- Updated Renko Chart
- Volume Color with Dynamic Limit
- White Theme
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

inverf
- inverse Gauss erf function

Math functions
(AmiBroker 6.40)

SYNTAX	**inverf(x)**
RETURNS	NUMBER or ARRAY
FUNCTION	Returns the inverse Gauss error function for given argument x. Error function is described here: https://en.wikipedia.org/wiki/Error_function
EXAMPLE	
SEE ALSO	erf() function

References:

The **inverf** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

InWatchList - watch list membership test (by ordinal number)

Information / Categories

SYNTAX	**InWatchList(listno)**
RETURNS	NUMBER
FUNCTION	Checks if the stock belongs to a watch list number *listno*. If yes - the function returns 1 otherwise 0.
EXAMPLE	Filter= InWatchList(3) OR InWatchList(5);
SEE ALSO	InWatchListName() function

References:

The **InWatchList** function is used in the following formulas in AFL on-line library:

- Count Tickers in Watchlist
- In Watch List

More information:

See updated/extended version on-line.

InWatchListName
- watch list membership test (by name)

Information / Categories
(AmiBroker 50)

SYNTAX	**InWatchListName("name")**
RETURNS	NUMBER
FUNCTION	Checks if the stock belongs to a watch list number *"listname"*. If yes - the function returns 1 otherwise 0.
EXAMPLE	Filter= InWatchListName("My Hotlist") OR InWatchList("My Second Hotlist");
SEE ALSO	InWatchList() function

References:

The **InWatchListName** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

IsContinuous
- checks 'continuous quotations' flag state

Information / Categories
(AmiBroker 4.60)

SYNTAX	**IsContinuous()**
RETURNS	NUMBER
FUNCTION	Returns 1 if current symbol has 'continuous quotations' flag turned on in Symbol->Information window. Returns zero otherwise.
EXAMPLE	
SEE ALSO	

References:

The **IsContinuous** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

IsEmpty
- empty value check

Miscellaneous functions
(AmiBroker 3.50)

SYNTAX	**IsEmpty(ARRAY)**
RETURNS	ARRAY
FUNCTION	returns 1 (or 'true') when given point in array is {empty} Note: {empty} value is used internaly by AFL to mark bars when the value is not available - for example for the first 20 bars the value of 20-day simple moving average is not available ({empty}) IsNull is a synonym for IsEmpty. It is suggested to use IsNull in new formulas, because of naming consistency with Null constant.

EXAMPLE	`movagv = ma( close, 30 );`
	`WriteIF( IsEmpty( movavg ), "Moving average not available yet", WriteVal( movavg ) );`

SEE ALSO	ISEMPTY() function , ISNAN() function , ISNULL() function , ISTRUE() function

References:

The **IsEmpty** function is used in the following formulas in AFL on-line library:

- AFL Timing functions
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- Caleb Lawrence
- channel indicator
- Channel/S&R and trendlines
- Elder safe Zone Long + short
- Fre
- Fund Screener
- Harmonic Pattern Detection
- lastNDaysBeforeDate
- Least Squares Channel Indicator
- MACD indicator display
- Probability Density & Gaussian Distribution
- QP2 Float Analysis
- Support and resistance
- Weekly chart
- Zig Zag

More information:

See updated/extended version on-line.

IsFavorite
- check if current symbol belongs to favorites

Information / Categories
(AmiBroker 4.50)

SYNTAX **IsFavorite()**

RETURNS NUMBER

FUNCTION The IsFavorite function returns True (1) if current symbol belongs to favorites, returns False (0) otherwise.

EXAMPLE

```
if( IsFavorite() )
{
  printf( Name() + " belongs to favourites " );
}
```

SEE ALSO IsIndex() function

References:

The **IsFavorite** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

IsFinite
- check if value is not infinite

Miscellaneous functions
(AmiBroker 4.30)

SYNTAX	**IsFinite(x)**
RETURNS	NUMBER, ARRAY
FUNCTION	returns a nonzero value (1 or TRUE) if its argument x is not infinite, that is, if INF < x < +INF. It returns 0 (FALSE) if the argument is infinite or a NaN. x can be number or array
EXAMPLE	IsFinite(1/0);
SEE ALSO	NZ() function , ISNAN() function

References:

The **IsFinite** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Automatic Linear Trend Channel
- tomy_frenchy
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

IsIndex
- check if current symbol is an index

Information / Categories
(AmiBroker 4.50)

SYNTAX **IsIndex()**

RETURNS NUMBER

FUNCTION The IsIndex function returns True (1) if current symbol is an index, returns False (0) otherwise.

EXAMPLE

```
if( IsIndex() )
{
  printf( Name() + " is an index" );
}
```

SEE ALSO IsFavorite() function

References:

The **IsIndex** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

IsNan
- checks for NaN (not a number)

Miscellaneous functions
(AmiBroker 4.30)

SYNTAX	**IsNan(x)**
RETURNS	NUMBER, ARRAY
FUNCTION	Returns a nonzero value (1 or TRUE) if the argument x is a NaN; otherwise it returns 0 (FALSE). A NaN is generated when the result of a floating-point operation cannot be represented in Institute of Electrical and Electronics Engineers (IEEE) format.
EXAMPLE	IsNan(0/0);
SEE ALSO	NZ() function

References:

The **IsNan** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- tomy_frenchy
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

IsNull
- check for Null (empty) value

Miscellaneous functions
(AmiBroker 4.30)

SYNTAX	**IsNull(x)**
RETURNS	NUMBER, ARRAY
FUNCTION	this function is synonym of IsEmpty(). Gives True if value is equal to Null (empty) value.
EXAMPLE	`movagv = ma( close, 30 );` `WriteIF( IsNull( movavg ), "Moving average not available yet", WriteVal( movavg ) );`
SEE ALSO	ISEMPTY() function

References:

The **IsNull** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- AR_Prediction.afl
- Auto-Optimization Framework
- AutoTrade using an Exploration
- Basket Trading System T101
- Chandelier Exit
- Congestions detection
- elliott wave manual labelling
- Heatmap V1
- Historical Volatility Index
- Market Meanness Index
- MCDX (Multi Color Dragon Extended)
- Open Range Breakout Trading System
- Peter Cooper
- Profit Table (Color Coded)
- Rene Rijnaars
- tomy_frenchy
- Trend Lines from 2 points
- Trigonometric Fit - TrigFit with AR for cos / sin
- TWS trade plotter
- Visi-Trade
- Volume based support resistance price finder

More information:

See updated/extended version on-line.

IsTrue
- true value (non-empty and non-zero) check

Miscellaneous functions
(AmiBroker 3.50)

SYNTAX	**istrue(ARRAY)**
RETURNS	ARRAY
FUNCTION	returns 1 (or 'true') when given point is not {empty} AND not zero
EXAMPLE	
SEE ALSO	

References:

The **IsTrue** function is used in the following formulas in AFL on-line library:

- Cycle Highlighter (auto best-fit)
- FirstBarIndex(), LastBarIndex()
- Relative Strength Multichart of up to 10 tickers

More information:

See updated/extended version on-line.

Kurtosis
- calculates kurtosis

Math functions
(AmiBroker 6.20)

SYNTAX **Kurtosis(ARRAY, range, population = True)**

RETURNS ARRAY

FUNCTION The function calculates Kurtosis.

Kurtosis(ARRAY, range, False) - works the same as Excel's KURT function

Kurtosis(ARRAY, range, True) - gives population Kurtosis (Excel does not have equivalent KURT.P function yet)

Note that these functions calculate excess kurtosis, so for normal distribution it is 0.

There is some controversy about what kurtosis really tells about distribution.

Most sources say that the kurtosis of a data set provides a measure of the peakedness of the distribution of the data, relative to the normal distribution. A positive kurtosis value indicates a relatively peaked distribution and a negative kurtosis value indicates a relatively flat distribution. But Dr. Peter Westfall published an article that addresses why kurtosis does not measure peakedness

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4321753/

He says that 'The kurtosis [...] is a measure of the combined weight of the tails relative to the rest of the distribution.'

EXAMPLE

SEE ALSO Skewness() function

References:

The **Kurtosis** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

LastValue - last value of the array

Trading system toolbox

SYNTAX **LastValue(ARRAY, lastmode = True)**

RETURNS NUMBER

FUNCTION Returns last calculated value of the specified ARRAY. The result of this function can be used in place of a constant (NUMBER) in any function argument.
If last bar of the ARRAY is undefined or Null (e.g., only 100-days loaded and you request the last value of a 200-day moving average) then the lastvalue function returns zero.
Caveat: since this function gets the LAST bar value of the array, it allows a formula to look into the future, if current bar is not the last one.

lastmode parameter:
(affects only commentary/interpretation)
When it is True - then true last value is used always
if it is False - then in commentary the 'selected' value is returned

In pre-4.08.1 versions commentary/interpretation/tooltip code evaluation was somewhat special because LastValue returned in fact not the last but the value of array at selected point (by vertical line or by tooltip) This caused some problems in displaying indicator values that used LastValue in its construction. To address this now LastValue used in Commentaries by default returns true last value. So you should modify your existing commentary/interpretation code that used LastValue to use now SelectedValue(array) function to maintain the same behaviour. Alternatively you can use LastValue(array, 0).

EXAMPLE

SEE ALSO SELECTEDVALUE() function

References:

The **LastValue** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 2 Timeframes Candlestick Bar Chart
- 3 Price Break
- 3TF Candlestick Bar Chart
- Advanced Trend Lines with S & R
- AllinOneAlerts - Module
- Alpha and Beta and R_Squared Indicator
- AR_Prediction.afl
- Auto Trade Step by Step
- Auto-Optimization Framework
- Automatic Trend-line
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- babaloo chapora
- Basket Trading System T101
- BEANS-Summary of Holdings

- Bullish Percent Index 2 files combined
- Button trading using AB auto trading interface
- Caleb Lawrence
- Candle Stick Analysis
- Candle Stick Demo
- CCI(20) Divergence Indicator
- Chart Zoom
- Color Display.afl
- Congestions detection
- Continuous Contract Rollover
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Dave Landry PullBack Scan
- DMI Spread Index
- ekeko price chart
- Elder Triple Screen Trading System
- elliott wave manual labelling
- Expiry day/days - Last thursday of month
- Export EOD or Intraday to .csv file
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- FirstBarIndex(), LastBarIndex()
- For Auto Trading Setup
- Frequency distribution of returns
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Futures - Dollar Move Indicator
- Gann Swing Chart
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- GFX ToolTip
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Harmonic Patterns
- Head & Shoulders Pattern
- Heatmap V1
- High Low Detection code
- How to add IB Option Symbols
- Hurst "Like" DE
- Hurst Constant
- Improved NH-NH scan / indicator
- Intraday Fibonacii Trend Break System
- Intraday Range and Periods Framer
- Intraday Trend Break System
- Intraday Volume EMA
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Kagi Chart

- Last Five Trades Result Dashboard – AFL code
- Linear Regression Line & Bands
- Linear Regression Line w/ Std Deviation Channels
- MACD commentary
- MACD indicator display
- Manual Bracket Order Trader
- Market Profile
- Modified Head & Shoulder Pattern
- Modified Momentum Finder DDT-NB
- Monthly bar chart
- Moving Average "Crash" Test
- Moving Averages NoX
- Multi-color Volume At Price (VAP)
- Multiple sinus noised
- Murrey Math Price Lines
- N-period candlesticks (time compression)
- Now Send Push Notifications From Amibroker
- nth (1 - 8) Order Polynomial Fit
- P&F Chart - High/Low prices Sept2003
- pattenz
- Pattern Recognition Exploration
- PF Chart - Close - April 2004
- Pivot Finder
- Pivot Point with S/R Trendlines
- Point & figure Chart India Securities
- Prashanth
- Price Persistency
- Prior Daily OHLC
- QP2 Float Analysis
- Ranking and sorting stocks
- Rea Time Daily Price Levels
- Regression Analysis Line
- Relative Strength Multichart of up to 10 tickers
- Renko Chart
- Revised Renko chart
- RSI Trendlines and Wedges
- shailu lunia
- Stan Weinstein strategy
- STD_STK Multi
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Stochastics Trendlines
- StochD_StochK Single.afl
- SUPER PIVOT POINTS
- Support and resistance
- Support Resistance levels
- TAPE READING
- The Fibonaccian behavior
- Time Left in Bar
- Time Left to Current Bar
- Tom DeMark Trend Lines
- Triangle exploration using P&F Chart

- Triangle search
- Triangle Search Extended
- Trigonometric Fit - TrigFit with AR for cos / sin
- Using From and To dates from Auto Analysis in Code
- Visible Min and Max Value Demo
- Weekly chart
- White Theme
- Williams Alligator system
- WLBuildProcess
- Zig Zag Indicator with Valid Entry and Exit Points
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount
- ZigZag Retracements

More information:

See updated/extended version on-line.

LastVisibleValue - get last visible value of array

Indicators
(AmiBroker 5.40)

SYNTAX **LastVisibleValue(array)**

RETURNS NUMBER

FUNCTION When used in charts / indicators /interpretation the function returns the values of array at the last visible bar.

In other (non-indicator) modes the function returns array element with subscripts of BarCount-1.

Note that these functions do not affect QuickAFL, so LastVisibleValue may be used instead of LastValue These functions are intended to complement functionality already available via HighestVisibleValue and LowestVisibleValue.

EXAMPLE

```
x = C;
Plot( x, "x", colorRed );
Plot( FirstVisibleValue( x ), "fvv", colorGreen );
Plot( LastVisibleValue( x ), "lvv", colorBlue );
```

SEE ALSO FirstVisibleValue() function , HighestVisibleValue() function , LowestVisibleValue() function

References:

The **LastVisibleValue** function is used in the following formulas in AFL on-line library:

- Alternative ZIG type function, multi TF
- channel indicator
- Channel/S&R and trendlines
- Congestions detection
- Harmonic Pattern Detection
- High Low Detection code
- Least Squares Channel Indicator
- Multi-color Volume At Price (VAP)
- Renko Chart
- Support and resistance
- Updated Renko Chart
- White Theme

More information:

See updated/extended version on-line.

LineArray - generate trend-line array

Exploration / Indicators
(AmiBroker 4.50)

SYNTAX **LineArray(*x0, y0, x1, y1, extend = 0, usebarindex = False*)**

RETURNS ARRAY

FUNCTION The **LineArray** function generates array equivalent to trend line drawn from point (x0, y0) to point (x1, y1). x coordinates are in bars (zero based), y coordinates are in dollars.

Note: x0 must be SMALLER than x1.

Note 2: the function accepts only numbers therefore generates single line. To produce multiple lines you have to call it many times with different co-ordinates.

extend parameter controls automatic extension of the trend line: if extend is 1 then line is right extended. if extend is 2 then line is left extended if extend is 3 then line is left and right extended

usebarindex parameter controls if x coordinates are interpreted as current array indexes (from 0..BarCount-1) (when *usebarindex* = False) or as absolute bar indexes (returned by BarIndex() function) when *usebarindex* = True. These two may differ if QuickAFL feature is turned on.

EXAMPLE

```
y0=LastValue(Trough(L,5,2));
y1=LastValue(Trough(L,5,1));
x0=BarCount - 1 - LastValue(TroughBars(L,5,2));
x1=BarCount - 1 - LastValue(TroughBars(L,5,1));
Line = LineArray( x0, y0, x1, y1, 1 );
Plot(C, "C", colorWhite, styleCandle);
Plot( Line, "Trend line", colorBlue );
```

SEE ALSO

References:

The **LineArray** function is used in the following formulas in AFL on-line library:

- 3 Price Break
- Advanced Trend Lines with S & R
- Another FIb Level
- CCI(20) Divergence Indicator
- Congestions detection
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gartley 222 Pattern Indicator
- Gordon Rose
- Halftrend
- Harmonic Patterns
- Intraday Fibonacii Trend Break System

- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- Non-repaitning Zigzag line
- pattenz
- PF Chart - Close - April 2004
- Pivot Point with S/R Trendlines
- Point & figure Chart India Securities
- Range Filter - Trading Strategy
- Rea Time Daily Price Levels
- Square of Nine Roadmap Charts
- Support and resistance
- TD Sequential
- Tom DeMark Trend Lines
- Trend Lines from 2 points
- Wolfe Wave Patterns
- Woodie's CCI Panel Full Stats
- ZigZag Retracements

More information:

See updated/extended version on-line.

LinearReg
- linear regression end-point

Statistical functions
(AmiBroker 4.20)

SYNTAX	**LinearReg(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates linear regression line end-point value according to a + b * x (where a and b are intercept and slope of linear regression line) from the ARRAY using *periods* range. The function accepts periods parameter that can be constant as well as time-variant (array).
EXAMPLE	LinearReg(close, 10);
SEE ALSO	

References:

The **LinearReg** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Average Price Crossover
- CCI Woodies Style
- DEBAJ
- Hull Range Indicator
- Hull Rate of Return Indicator
- Linear Candle
- Moving Trend Bands (MTB)
- Price with Woodies Pivots
- Standard Error Bands (Native AFL)
- Trend Detection
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots

More information:

See updated/extended version on-line.

LinRegIntercept -

Statistical functions
(AmiBroker 4.20)

SYNTAX	**LinRegIntercept(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates intercept of linear regression line - the "a" coefficient in a + b*x (LinRegSlope calculates b) from the ARRAY using *periods* range. The function accepts periods parameter that can be constant as well as time-variant (array).

EXAMPLE	x = Cum(1);
	lastx = LastValue(x); Daysback = 10; aa = LastValue(LinRegIntercept(Close, Daysback));
	bb = LastValue(LinRegSlope(Close, Daysback));
	
	y = Aa + bb * (x - (Lastx - DaysBack)); Plot(Close, "Close", colorBlack, styleCandle);
	Plot(IIf(x >= (lastx - Daysback), y, -1e10), "LinReg", colorRed);

SEE ALSO	

References:

The **LinRegIntercept** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- correlerror
- Dave Landry PullBack Scan
- Linear Regression Line w/ Std Deviation Channels
- OBV with Linear Regression
- regavg
- Trigonometric Fit - TrigFit with AR for cos / sin
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

LinRegSlope - linear regression slope

Statistical functions
(AmiBroker 3.40)

SYNTAX **LinRegSlope(ARRAY, *periods*)**

RETURNS ARRAY

FUNCTION Calculates linear regression line slope from the ARRAY using *periods* range. The function accepts periods parameter that can be constant as well as time-variant (array).

EXAMPLE

```
x = Cum(1);
lastx = LastValue( x ); Daysback = 10; aa = LastValue( LinRegIntercept( Close, Daysback) );
bb = LastValue( LinRegSlope( Close, Daysback ) );

y = Aa + bb * ( x - (Lastx - DaysBack) ); Plot( Close, "Close", colorBlack, styleCandle );
Plot( IIf( x >= (lastx - Daysback), y, -1e10 ), "LinReg", colorRed );
```

SEE ALSO

References:

The **LinRegSlope** function is used in the following formulas in AFL on-line library:

- Advanced Trend Lines with S & R
- AR_Prediction.afl
- ATR Study
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- CCT Kaleidoscope
- correlerror
- Dave Landry PullBack Scan
- ekeko price chart
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Gabriel Linear Regression Angle Indicator
- Gann Five Day pullback
- JEEVAN'S SRI CHAKRA
- Linear Regression Line w/ Std Deviation Channels
- pattenz
- Perceptron
- prakash
- R-Squared
- regavg
- Regression Analysis Line
- RSIS
- Trend Analysis_Comentary
- Trend exploration with multiple timeframes
- Trend Exploration: Slope Moving Average
- Trigonometric Fit - TrigFit with AR for cos / sin
- Visualization of stoploses and profit in chart
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

LLV
- lowest low value

Lowest/Highest

SYNTAX	**llv(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates the lowest value in the ARRAY over the preceding *periods* (*periods* includes the current day). The function accepts periods parameter that can be constant as well as time-variant (array).
EXAMPLE	The formula "llv(close, 14)" returns the lowest closing price over the preceding 14 periods.
SEE ALSO	The hhv() function (see Highest High Value).

References:

The **LLV** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- % B of Bollinger Bands With Adaptive Zones
- 10-20 Indicator
- 30 Week Hi Indicator - Calculate
- 52 Week New High-New Low Index
- Adaptave Zones O/B & O/S Oscillator
- Adaptive Price Channel
- Advanced MA system
- Advisory NRx price chart display.
- ADXVMA
- Against all odds
- AJDX system
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- An n bar Reversal Indicator
- Another FIb Level
- Aroon
- AR_Prediction.afl
- ATR Study
- Auto Fib Ext&Retracement
- Auto-Optimization Framework
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Awsome Oscilator
- babaloo chapora
- Brian Wild
- Bull Fear / Bear Fear
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Caleb Lawrence
- CAMSLIM Cup and Handle Pattern AFL
- Candle Stick Demo

- Candlestick Commentary
- Candlestick Commentary Modified
- Candlestick Commentary-modified
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI(20) Divergence Indicator
- CCT Kaleidoscope
- CCT StochasticRSI
- Chaikin Volume Accumulation
- Chandelier Exit
- Chandelier Exit or Advanced Trailing Stop
- Channel/S&R and trendlines
- Compare Sectors against Tickers
- Congestions detection
- CVR--severe filter
- Dahl Oscillator modified
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Dave Landry PullBack Scan
- Dave Landry Pullbacks
- Demand Index
- Demand Index
- Dinapoli Perferred Stochastic
- Divergences
- Donchian Channel
- Double Smoothed Stochastic from W.Bressert
- DT Oscillator
- Ed Seykota's TSP: Support and Resistance
- Ehlers Fisher Transform
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- ElderSafeZoneStopShort
- FastStochK FullStochK-D
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator
- Fisher Relative Vigour Index
- Fre
- Frequency distribution of returns
- Fund Screener
- Gann level plotter
- Gann Swing chart v41212
- Gfx Toolkit
- Gordon Rose
- Harmonic Pattern Detection
- Head & Shoulders Pattern
- Hilbert Sine Wave Support & Resistance
- IBD relative strength database ranker
- Ichimoku Chart
- Ichimoku charts
- Ichimoku Kinko Hyo

- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Ichimoku with plot mofified to use cloud function
- IchimokuBrianViorelRO
- IFT of RSI - Multiple TimeFrames
- Improved NH-NH scan / indicator
- Index of 30 Wk Highs Vs Lows
- Inter-market Yield Linear Regression Divergence
- Intraday Strength
- JEEVAN'S SRI CHAKRA
- Kagi Chart
- Larry William's Volatility Channels
- MACD commentary
- MACD indicator display
- Meu Sistema de Trading - versão 1.0
- mitalpradip
- Modified Head & Shoulder Pattern
- Monthly bar chart
- MS Darvas Box with Exploration
- MultiCycle 1.0
- Multiple Ribbon Demo
- Murrey Math Price Lines
- N-period candlesticks (time compression)
- New HL Scanner
- nikhil
- Non-repaitning Zigzag line
- NR4 Historical Volatility System
- NRx Exploration
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- Pattern - Rectangle Base Breakout on High Vol
- Pattern_-_Rectangle_Base_Breakout_on_High_Vol 2
- Peterson
- PF Chart - Close - April 2004
- Pivot Finder
- Pivots And Prices
- Point & figure Chart India Securities
- Polyfit Lines
- prakash
- Prashanth
- Price with Woodies Pivots
- Probability Density & Gaussian Distribution
- Pullback System No. 1
- PVT Trend Decider
- Rainbow Oscillator
- Rea Time Daily Price Levels
- Regression Analysis Line
- Relative Strength Index
- RSI Double-Bottom
- RSI Pointer
- RSIS
- RUTVOL timing signal with BB Scoring routine

- Sainath Sidgiddi
- Scan New High and New Low
- shailu lunia
- Sony
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Stochastic %J - KDJ
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic Divergence, positive
- Stochastic Divergences, PDI, NDI
- Stochastic Fast%K and Full
- Stochastic OBV and Price Filter
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stop-loss Indicator bands
- Stops Implementation in AFS
- Stress with SuperSmoother
- Sun&Cloud
- Support and Resistance
- Support and resistance
- Support Resistance levels
- TD Moving Average I
- TD sequential
- TD Sequential
- The Stochastic CCI
- Three Day Balance Point
- Trend Exploration: Count Number of New Highs
- Trend Trigger Factor
- Triangle exploration using P&F Chart
- Triangle search
- Triangle Search Extended
- Trigonometric Fit - TrigFit with AR for cos / sin
- ValueChart
- Vertical Horizontal Filter
- Vertical Horizontal Filter (VHF)
- visual turtle trading system
- Volatility System
- Weekly chart
- Weinberg's The Range Indicator
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- Zig Zag
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

LLVBars
- bars since lowest low

Lowest/Highest

SYNTAX	**LLVBars(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates the number of periods that have passed since the ARRAY reached its *periods* period trough. The function accepts periods parameter that can be constant as well as time-variant (array).
EXAMPLE	The formula "llvbars(close,50)" returns the number of periods that have passed since the closing price reached its 50 period trough.
SEE ALSO	

References:

The **LLVBars** function is used in the following formulas in AFL on-line library:

- Advanced MA system
- Aroon Indicators
- Aroon The Advisor
- babaloo chapora
- CAMSLIM Cup and Handle Pattern AFL
- CCI(20) Divergence Indicator
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Divergences
- ekeko price chart
- Fre
- Fund Screener
- Gordon Rose
- Halftrend
- Linear Regression Line & Bands
- Pivot Finder
- shailu lunia
- Stochastic Divergence, positive
- Stochastic Divergences, PDI, NDI
- Triangle search
- Triangle Search Extended
- Vivek Jain

More information:

See updated/extended version on-line.

log
- natural logarithm

Math functions

SYNTAX	**log(NUMBER)**
	log(ARRAY)

RETURNS	NUMBER ARRAY
FUNCTION	Calculates the natural logarithm of NUMBER or ARRAY.
EXAMPLE	
SEE ALSO	exp() Exponential function

Comments:

Tomasz Janeczko 2006-03-02 04:26:40	The synonym to 'log' is 'ln' function.

References:

The **log** function is used in the following formulas in AFL on-line library:

- Andrews Pitchfork
- Andrews PitchforkV3.3
- AR_Prediction.afl
- CAMSLIM Cup and Handle Pattern AFL
- Dave Landry PullBack Scan
- Ehler's filters and indicators
- Ehlers Fisher Transform
- Elder Triple Screen Trading System
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator
- Fisher Relative Vigour Index
- Frequency distribution of returns
- Historical Volotility Scan - 6/100
- Historical Volotility Scan - 50 Day
- Hurst Constant
- IFT of RSI - Multiple TimeFrames
- MultiCycle 1.0
- NR4 Historical Volatility System
- NRx Exploration
- Probability Calculator
- Schiff Lines
- Signal to Noise
- Sony
- Trigonometric Fit - TrigFit with AR for cos / sin
- Volume Occilator
- Volume Oscillator
- Zig Zag

More information:

See updated/extended version on-line.

log10
- decimal logarithm

Math functions

SYNTAX	**log10(NUMBER)**
	log10(ARRAY)

RETURNS	NUMBER ARRAY
FUNCTION	Calculates the decimal logarithm of NUMBER or ARRAY.
EXAMPLE	
SEE ALSO	.

References:

The **log10** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Bad Tick Trim on 5 sec database
- Cycle Period
- elliott wave manual labelling
- Geometric Mean of Volume
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

Lookup
- search the array for bar with specified date/time

Date/Time
(AmiBroker 5.40)

SYNTAX **Lookup(array, datetime, mode = 0)**

RETURNS NUMBER

FUNCTION The function searches for the bar with specified datetime and returns the value from the same position of the input array.

Parameter 'mode' decides how search is performed in case when exact match is not found:

- mode = 0 - find exact match, otherwise return Null
- mode = -1 - find exact match, otherwise return nearest predecesor (if datetime is past last bar it will return last bar value)
- mode = -2 - find exact match, otherwise return nearest predecessor EXCEPT the very last bar (if searched datetime is past last bar it will return Null)
- mode = 1 - find exact match, otherwise return nearest successor (if datetime is before first bar it will return first bar value)
- mode = 2 - find exact match, otherwise return nearest successor EXCEPT the very first bar (if searched datetime is before first bar it will return Null)

This function uses very fast binary search and it is many times faster than previous AFL-based methods such as FindValueAtDateTime() presented in the past. Any call to FindValueAtDateTime (input, dt, value) can be now replaced with Lookup(input, value) (here is no need to pass dt- datetime).

NOTE: This function does not affect QuickAFL required bars, therefore it will only search bars that are actually loaded in arrays. For indicators it may mean that it won't be able to find value if it is invisible, unless you use SetBarsRequired() function to ensure that more bars are loaded.

EXAMPLE
```
InputDate = "2011-04-05";
Title = "Close value at (or before) " + InputDate + " is " + Lookup(
Close, _DT( InputDate ), -1 );
```

SEE ALSO DateTime() function , StrToDateTime() function , _DT() function

References:

The **Lookup** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- GFX ToolTip
- High Low Detection code

More information:

See updated/extended version on-line.

Lowest
- lowest value

Lowest/Highest

SYNTAX	**Lowest(ARRAY)**
RETURNS	ARRAY
FUNCTION	Calculates the lowest value in the ARRAY since the first day/bar present in the database.
EXAMPLE	The formula `lowest( rsi(14) );` returns the lowest Relative Strength Index value ; `lowest ( close )` returns the lowest closing price.
SEE ALSO	HHV() function , LLV() function , HIGHEST() function

References:

The **Lowest** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Candle Stick Analysis
- Candle Stick Demo
- CVR--severe filter
- Multiple sinus noised
- Perceptron
- Triangle exploration using P&F Chart
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

LowestBars
- bars since lowest

Lowest/Highest

SYNTAX	**LowestBars(ARRAY)**
RETURNS	ARRAY
FUNCTION	Calculates the number of periods that have passed since the ARRAY s lowest value.
EXAMPLE	The formula"lowestbars(close)" returns the number of periods that have passed since the closing price reached its lowest point.
SEE ALSO	

References:

The **LowestBars** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

LowestSince - lowest value since condition met

Lowest/Highest
(AmiBroker 3.40)

SYNTAX	**LowestSince(EXPRESSION, ARRAY, *Nth* = 1)**
RETURNS	ARRAY
FUNCTION	Returns the lowest ARRAY value since EXPRESSION was true on the Nth most recent occurrence.
EXAMPLE	lowestsince(Cross(macd(), 0), Close, 1) returns the lowest close price since macd() has crossed above zero.
SEE ALSO	

References:

The **LowestSince** function is used in the following formulas in AFL on-line library:

- Gann Swing Chart
- Gann Swing Charts in 3 modes with text
- Harmonic Pattern Detection
- High Low Detection code
- Last Five Trades Result Dashboard – AFL code
- Market Profile
- MO_CrashZone
- RSI of Weekly Price Array
- Stochastic of Weekly Price Array
- Time Frame Weekly Bars
- Visible Min and Max Value Demo

More information:

See updated/extended version on-line.

LowestSinceBars - barssince lowest value since condition met

Lowest/Highest
(AmiBroker 3.40)

SYNTAX	**LowestSinceBars(EXPRESSION, ARRAY, *Nth* = 1)**
RETURNS	ARRAY
FUNCTION	Returns the number of bars that have passed since lowest ARRAY value since EXPRESSION was true on the Nth most recent occurrence.
EXAMPLE	lowestsincebars(Cross(macd(), 0), Close, 1) returns the number of bars passed since the lowest close price was detected from the time when macd() has crossed above zero.
SEE ALSO	

References:

The **LowestSinceBars** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

LowestVisibleValue - get the lowest value within visible chart area

Indicators
(AmiBroker 5.30)

SYNTAX **LowestVisibleValue(array)**

RETURNS NUMBER

FUNCTION The function calculates single value (not array) representing lowest value of given array within VISIBLE range (on chart).

Should be applied only in indicators as only indicators have concept of "visible" bars. The function will return Null value if no visible bars are present. The function is equivalent to the following coding:

```
function LowestVisibleValueEquivalent( array )
{
 bv = Status("barvisible");
 ll = 1e8;
 for( i = 0; i < BarCount; i++ )
 {
  if( bv[ i ] AND array[ i ] < ll ) ll = array[ i ];
 }
 return ll;
}
```

EXAMPLE

SEE ALSO HighestVisibleValue() function

References:

The **LowestVisibleValue** function is used in the following formulas in AFL on-line library:

- Auto Fib Ext&Retracement
- Day Bar No
- Fre
- New HL Scanner
- Renko Chart
- Updated Renko Chart
- Volume Charts
- White Theme

More information:

See updated/extended version on-line.

MA - simple moving average

Moving averages, summation

SYNTAX	**ma(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates a *periods* simple moving average of ARRAY The function accepts periods parameter that can be constant as well as time-variant (array).
EXAMPLE	ma(CLOSE, 5)
SEE ALSO	TEMA() function , AMA() function , AMA2() function , DEMA() function , WMA() function , WILDERS() function , EMA() function

References:

The **MA** function is used in the following formulas in AFL on-line library:

- % B of Bollinger Bands With Adaptive Zones
- 'R' Channel
- 3 ways to use RMI in one script
- AC+ acceleration
- accum/dist mov avg crossover SAR
- AccuTrack
- Advanced MA system
- Adverse Move Ratio
- ADXbuy
- AFL Example
- AFL Example - Enhanced
- AFL to Python COM Link
- AFL-Excel
- Against all odds
- ALJEHANI
- Alphatrend
- Andrews PitchforkV3.3
- AO+ Momentum indicator
- AO+Momentum
- Application of Ehler filter
- Arnaud Legoux Moving Average (ALMA)
- AR_Prediction.afl
- ATR Study
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- automatic trendlines using fractal patterns
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Dollar Price Volatility Exploration
- Average Price Crossover
- Awsome Oscilator
- B-Xtrender

- babaloo chapora
- Bad Tick Trim on 5 sec database
- Balance of Power
- balance of power
- BB squeeze
- BBAreacolor&TGLCROSSNEW
- Better Bollinger Bands
- Bman's HaDiffCO
- BMTRIX Intermediate Term Market Trend Indicator
- Bollinger - Keltner Bands
- Bollinger band normalization
- Bollinger Band Squeeze
- Bollinger Fibonacci Bands
- Bollinger oscillator
- Bottom Fisher Exploration
- Bow tie
- Breadth Thrust
- Bull/Bear Volume
- Bullish Percent Index 2 files combined
- CAMSLIM Cup and Handle Pattern AFL
- Candle Identification
- Candle Pattern Function
- Candle Stick Analysis
- candlestick chart for Volume/RSI/OBV
- Candlestick Volume Bars with Moving Average
- CandleStochastics
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- CCI/DI+- COMBO indicator
- CCT Bollinger Band Oscillator
- CCT FibAccordion
- CCT Kaleidoscope
- Chaikin's Volatility
- Chande Momentum Oscillator
- Chande's Trend Score
- Chandelier Exit
- Chandelier Exit
- Chandelier Exit or Advanced Trailing Stop
- changing period moving avarage
- CoinToss ver 1
- Color Price Bar - Impulse System
- com-out
- COMBO
- Commodity Selection Index (CSI)
- Compare Sectors against Tickers
- Composite Index
- Congestions detection
- Connors TPS
- Continuous Contract Rollover
- Coppock Curve
- Coppock Histogram

- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- correlerror
- crMathLib
- CVR--severe filter
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Dahl Oscillator modified
- Daily High Low in Advance
- danningham penetration
- Darvas Amibroker
- Dave Landry PullBack Scan
- Dave Landry Pullbacks
- DEBAJ
- Demand Index
- Demand Index
- Derivative Oscillator
- DiNapoli Detrended Oscillator
- DiNapolis 3x Displaced Moving Averages
- Divergences
- DMI Spread Index
- Dominant Cycle Phase
- Double Smoothed Stochastic from W.Bressert
- DPO with shading
- DT Oscillator
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Effective Swing Indicator
- ekeko price chart
- Elder Bear Power
- Elder Bull Power
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Ray - Bull Bear
- Elder Ray Oscillator with MA
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- Elliott Wave Oscillator
- Ema bands
- EMA Crossover
- EMA Crossover Price
- Envelope System
- Ergodic Oscillator
- FastStochK FullStochK-D
- Fibonacci Moving averages
- fishnet
- Follow the Leader
- Force index
- Forward/Reverse EMA by John Ehlers
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Fund Screener

- Future MA Projection
- Futures - Dollar Move Today Indicator
- Gabriel Linear Regression Angle Indicator
- Gann Five Day pullback
- Gann HiLo Indicator and System
- Gann Swing chart v41212
- Gann Swing Charts in 3 modes with text
- garythompson
- garythompson
- Gfx Toolkit
- Gordon Rose
- Guppy Cloud
- Guppy moving averages
- Halftrend
- Heatmap V1
- Heikin Ashi Candles
- Heikin Ashi Delta
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- Hilbert Study
- HLspread
- Hull Moving Average
- Hull Multiple Moving Averages
- Hull with DEMA
- Hurst "Like" DE
- ICHIMOKU SIGNAL TRADER
- IFT of RSI - Multiple TimeFrames
- Index and ETF trading
- Inter-market Yield Linear Regression Divergence
- INTRADAY HEIKIN ASHI new
- IntraDay Open Marker
- Intraday Volume EMA
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- John Ehler
- Kairi Relative Index
- Kelly criterion
- Keltner Channel
- Know Sure Thing
- Lagging MA-Xover
- Laugerre PPO Oscillator
- Linear Candle
- MA Difference 20 Period
- MACD BB Indicator
- MACD Histogram - Change in Direction
- MACD-V
- Main price chart with Rainbow & SAR
- MAM
- Market Direction

- MAVG
- McClellan Oscillator
- McClellan Summation Index
- Meu Sistema de Trading - versão 1.0
- mfimacd
- MOCS
- Modified Head & Shoulder Pattern
- Modified-DVI
- Momentum
- Momentum
- Momentum Volume Price (MVP) Indicator
- Moving Average "Crash" Test
- Moving Averages NoX
- MS Darvas Box with Exploration
- MultiCycle 1.0
- Multiple Ribbon Demo
- Nadaraya-Watson Envelope
- NASDAQ 100 Volume
- Neural Network Powered Smooth/Predictive RSI V2
- nikhil
- Noor_Doodie
- Parametric Chande Trendscore
- Pattern - Rectangle Base Breakout on High Vol
- Pattern_-_Rectangle_Base_Breakout_on_High_Vol 2
- Percentage Price Oscillator
- Perceptron
- Peterson
- Pivots for Intraday Forex Charts
- Plot the Equity Curve without Backtesting?
- Polarized Fractal Efficiency
- prakash
- Price with Woodies Pivots
- Price-Volume Rank
- Probability Density & Gaussian Distribution
- Projection Oscillator
- Pullback System No. 1
- QP2 Float Analysis
- QStick
- Rainbow Charts
- Rainbow Oscillator
- Random Walk Index
- Random Walk Index, base formula included
- Range Filter - Trading Strategy
- Rapid Prototyping Method for System Development
- regavg
- Relative Strength
- Relative strength comparison with moving average
- Relative Volume
- Renko Chart
- Reverse EMA function
- RSI + Avgs
- RSI Double-Bottom

- RSI Pointer
- RSIS
- RUTVOL timing signal with BB Scoring routine
- Sainath Sidgiddi
- Scale Out: Futures
- Schiff Lines
- Sector Tracking
- SectorRSI
- SF Entry,Stop, PT Indicator
- shailu lunia
- Signal to Noise
- SIROC Momentum
- Stan Weinstein strategy
- STARC Bands
- STD_STK Multi
- STO & MACD Buy Signals with Money-Management
- Stochastic %J - KDJ
- Stochastic Fast%K and Full
- Stochastic of Weekly Price Array
- Stochastic RSI
- StochD_StochK Single.afl
- Stops on percentages
- Sun&Cloud
- Support and Resistance
- Support Resistance levels
- T3
- T3 Function
- TAZ Trading Method Exploration
- TD Channel-1
- TD Channel-2
- TD Moving Average 2
- TD Moving Average I
- testing multiple system simulataneously
- The D_oscillator
- The Mean RSIt (variations)
- The Relative Slope
- The Relative Slope Pivots
- The Saturation Indicator D_sat
- tomy_frenchy
- Trend Detection
- Trend exploration with multiple timeframes
- Trend Exploration: Slope Moving Average
- TRENDAdvisor
- Triangle exploration using P&F Chart
- Triangle search
- Triangle Search Extended
- Triangular Moving Average
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- TRIX
- TRIXXX
- TSV

- TTM Squeeze
- Tushar Chande's Projected Range
- Updated Renko Chart
- ValueChart
- VAMA
- Varexlist
- Vertical Horizontal Filter (VHF)
- Vic Huebner
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- Visualization of stoploses and profit in chart
- Vivek Jain
- Volatility
- Volatility Breakout with Bollinger Bands
- Volatility Quality Index
- Volume - compared with Moving Avg (100%)
- Volume Color with Dynamic Limit
- Volume Occilator
- Volume Oscillator
- Volume Spread for VSA
- Volume Zone Oscillator
- VolumeDivAvgV3_Study_BrS
- Weekly Trend in Daily Graph
- Weinberg's The Range Indicator
- William's Alligator System II
- Williams Alligator system
- WILSON RELATIVE PRICE CHANNEL
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- Woodie's Price Panel With Woodie's Pivots
- z-distance from vwap
- Z-Score Indicator
- ZeroLag MACD(p,q,r)
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

MACD
- moving average convergence/divergence

Indicators

SYNTAX	**macd(*fast* = 12, *slow* = 26)**
RETURNS	ARRAY
FUNCTION	Calculates the MACD indicator using *fast* and *slow* averaging periods.
EXAMPLE	The formula "macd()" returns the value of the MACD indicator (i.e., the red line). The formula "signal()" returns the value of the MACD's signal line (i.e., the blue line).
SEE ALSO	The signal() function.

References:

The **MACD** function is used in the following formulas in AFL on-line library:

- Adaptave Zones O/B & O/S Oscillator
- AFL Example - Enhanced
- ALJEHANI
- BBAreacolor&TGLCROSSNEW
- Bollinger band normalization
- Button trading using AB auto trading interface
- Color MACD Histogram Changes
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Compare Sectors against Tickers
- Customised Avg. Profit %, Avg. Loss % etc
- Dinapoli Guru Commentary
- ekeko price chart
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Triple Screen Trading System
- Fund Screener
- hassan
- ICHIMOKU SIGNAL TRADER
- Indicator Explorer (ZigZag)
- Last Five Trades Result Dashboard – AFL code
- MACD commentary
- MACD Histogram - Change in Direction
- MACD indicator display
- MACD optimize
- Meu Sistema de Trading - versão 1.0
- STO & MACD Buy Signals with Money-Management
- swing chart
- The Mean RSIt
- The Mean RSIt (variations)
- Trend Analysis_Comentary
- Trending or Trading ?
- Trending Ribbon
- TrendingRibbonArrowsADX
- Vivek Jain

- ZeroLag MACD(p,q,r)

More information:

See updated/extended version on-line.

MapCreate
- creates a map/dictionary object (holding key-value pairs)

Map functions
(AmiBroker 6.90)

SYNTAX **MapCreate(hash_size = 0)**

RETURNS MAP

FUNCTION Creates a new Map object and returns the map. In AFL, a Map is a data structure that stores key-value pairs, similar to dictionaries in other programming languages. Map keys are strings. Values can be of any type.

Advanced users may pass hash table size if they want to store lots of data in a map in hash_size parameter. Hash size should be prime number larger than maximum expected number of elements in the map

EXAMPLE

```
// Map creation
// larger hash table size improves performance should be prime
number larger than number of elements expected
m = MapCreate( 997 );

// assigning values
m["MSFT"] = 1;
m["NVDA"] = 2;
m["CSCO"] = 3;

// accessing value by key
value = m[ Name() ]; // using function return value as a key
value2 = m[ "MSFT" ]; // literal key

// checking for key existence
value = m[ Name() ];
if (IsNull(value))
{
  printf("Key not found in Map");
}
else
{
  printf("%g", value);
}

// EXAMPLE 2

// Passing map arguments to user functions
function Test(x)
{
  x["key"] = 1;
}

m2 = Map();

// IMPORTANT CAVEAT:
```

```
// UNLIKE other data types,
// Maps are always passed by reference so this call modifies its
contents
Test(m2);

// Check the modified map
value = m2["key"];

if (IsNull(value))
{
  printf("Key not found in Map");
}
else
{
  printf("%g", value);
}
```

SEE ALSO

References:

The **MapCreate** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

MarketID - market ID / name

Information / Categories
(AmiBroker 3.80)

SYNTAX	**MarketID(mode = 0)**
RETURNS	NUMBER/STRING
FUNCTION	Retrieves current stock market ID/name When mode = 0 (the default value) this function returns numerical marketID (consecutive market number) When mode = 1 this function returns name of the market.

EXAMPLE	Filter = MarketID() == 7 OR MarketID() == 9;
	AddTextColumn(MarketID(1), "Market name");

SEE ALSO	

References:

The **MarketID** function is used in the following formulas in AFL on-line library:

- Alert Output As Quick Rewiev
- Auto-Optimization Framework
- Compare Sectors against Tickers
- Dave Landry PullBack Scan
- Elder Triple Screen Trading System

More information:

See updated/extended version on-line.

Matrix
- create a new matrix

Matrix functions
(AmiBroker 60)

SYNTAX **Matrix(rows, cols, initvalue, increment = 0)**

RETURNS Matrix

FUNCTION The function creates a new matrix of user specified dimensions with all elements filled with initvalue. An optional parameter 'increment' that allows to create a matrix with monotonically increasing elements.

To create a matrix use

my_var_name = Matrix(rows, cols, initvalue)

To access matrix elements, use:

my_var_name[row][col]

where
row is a row index (0... number of rows-1)
and
col is a column index (0... number of columns-1)

Matrices and their elements support all scalar (element-wise) arithmetic and logical operations

So you can for example add, subtract, multiply, divide two matrices if they have same dimensions with one call.

EXAMPLE

```
x = Matrix( 5, 6, 9 ); // matrix 5 rows 6 columns, initial value 9
y = Matrix( 5, 6, 10 ); // matrix 5 rows 6 columns, initial value 10

z = y - z; // will give you matrix 5 rows and 6 columns filled with
elements holding value 1 (difference between 10 and 9).
```

SEE ALSO

References:

The **Matrix** function is used in the following formulas in AFL on-line library:

- channel indicator
- elliott wave manual labelling
- Gfx Toolkit
- Least Squares Channel Indicator
- Polyfit Lines
- Spearman Rank Correlation Coefficient

More information:

See updated/extended version on-line.

Max
- maximum value of two numbers / arrays

Math functions

SYNTAX	**Max(ARRAY1, ARRAY2)**
RETURNS	ARRAY
FUNCTION	Returns the largest of the two parameters.
EXAMPLE	The formula "max(CLOSE, 10)" returns either the closing price or 10, whichever is greater. The formula "max(-14, 13)" always returns 13.
SEE ALSO	

References:

The **Max** function is used in the following formulas in AFL on-line library:

- 'R' Channel
- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Advanced MA system
- ADXVMA
- AFL_Glossary_Converter
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- AR_Prediction.afl
- ATR Study
- Auto Fib Ext&Retracement
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- Bad Tick Trim on 5 sec database
- BBAreacolor&TGLCROSSNEW
- Bman's HaDiffCO
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Caleb Lawrence
- candlestick chart for Volume/RSI/OBV
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI(20) Divergence Indicator
- Channel/S&R and trendlines
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- correlerror
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period

- Dominant Cycle Phase
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- Ehler's filters and indicators
- elliott wave manual labelling
- EMA Crossover
- Fisher Oscillator
- For Auto Trading Setup
- Gann Swing Charts in 3 modes with text
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Heatmap V1
- Heikin Ashi Candles
- Heikin Ashi Delta
- Heikin Ashi System
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- HLspread
- interactively test discretionary trading
- INTRADAY HEIKIN ASHI new
- JEEVAN'S SRI CHAKRA
- John Ehler
- Kiss and Touch with the Modified True Range
- Linear Candle
- MACD indicator display
- Market Breadth Chart-In-Chart
- mitalpradip
- MO_CrashZone
- Multi-color Volume At Price (VAP)
- New HL Scanner
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- Parabolic SAR in VBScript
- pattenz
- Performance Check
- PF Chart - Close - April 2004
- Pivots for Intraday Forex Charts
- Polyfit Lines
- Probability Density & Gaussian Distribution
- Profit Table (Color Coded)
- Projection Oscillator
- PVT Trend Decider
- Rainbow Oscillator
- Random Walk Index, base formula included
- Rebalancing Backtest avoiding leverage

- Renko Chart
- Renko Chart
- Revised Renko chart
- Scale Out: Futures
- shailu lunia
- Signal to Noise
- Stops Implementation in AFS
- Stops on percentages
- Support and resistance
- Three Line Break - TLB
- Trailing Stop Loss
- Trend Detection
- Trend Following System
- Trend Lines from 2 points
- Triangle exploration using P&F Chart
- Triangle Search Extended
- Trigonometric Fit - TrigFit with AR for cos / sin
- Twiggs Money Flow
- Twiggs money flow weekly
- Updated Renko Chart
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- Volume Divided Histogram
- VolumeDivAvgV3_Study_BrS
- VSTOP (2)
- VSTOP (3)
- William's Alligator System II
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- Zig Zag
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

MDI
- minus directional movement indicator (-DI)

Indicators
(AmiBroker 3.30)

SYNTAX	**mdi(period = 14)**
RETURNS	ARRAY
FUNCTION	Calculates Minus Directional Movement Indicator (-DI line)
EXAMPLE	mdi()
SEE ALSO	

References:

The **MDI** function is used in the following formulas in AFL on-line library:

- ADX Indicator - Colored
- ADXbuy
- AJDX system
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- CCI/DI+- COMBO indicator
- Commodity Selection Index (CSI)
- Dave Landry Pullbacks
- DMI Spread Index
- ekeko price chart
- Heatmap V1
- ICHIMOKU SIGNAL TRADER
- Index and ETF trading
- Mndahoo ADX
- Multiple Ribbon Demo
- swing chart
- The Three Day Reversal
- Trend Analysis_Comentary
- Trending Ribbon
- TrendingRibbonArrowsADX
- Vivek Jain

More information:

See updated/extended version on-line.

Median
- calculate median (middle element)

Statistical functions
(AmiBroker 4.50)

SYNTAX	**Median(** ***array, period*** **)**
RETURNS	ARRAY
FUNCTION	The Median function - finds median (middle element) value of the *array* over *period* elements. Note that LOWER median is returned when 'period' is an even number. If you want to get average of upper and lower median for even 'periods' you need to use Percentile(array, period, 50) instead. It will do the averaging for you but runs slower.

EXAMPLE

```
// list only symbols which volume is greater than
// median Volume from past 50 days
Filter = Volume > Median( Volume, 50 );
AddColumn( V, "Volume" );
```

SEE ALSO Percentile() function

References:

The **Median** function is used in the following formulas in AFL on-line library:

- Adaptive Laguerre Filter, from John Ehlers
- AR_Prediction.afl
- Bad Tick Trim on 5 sec database
- Cycle Period
- Elder's Market Thermometer
- Market Meanness Index
- Noor_Doodie
- Trigonometric Fit - TrigFit with AR for cos / sin
- Zig Zag

More information:

See updated/extended version on-line.

MFI - money flow index

Indicators

SYNTAX	**mfi(** ***periods*** **= 14)**
RETURNS	ARRAY
FUNCTION	Calculates the Money Flow Index with *period* range
EXAMPLE	mfi(16)
SEE ALSO	The rsi() function (see Relative Strength Index (RSI)).

References:

The **MFI** function is used in the following formulas in AFL on-line library:

- Adaptave Zones O/B & O/S Oscillator
- Against all odds
- Alphatrend
- BBAreacolor&TGLCROSSNEW
- Bollinger band normalization
- DateNum_DateStr
- ICHIMOKU SIGNAL TRADER
- Market Facilitation Index VS Volume
- mfimacd
- Ranking Ticker WatchList
- Reverse MFI Crossover
- Volatility Breakout with Bollinger Bands

More information:

See updated/extended version on-line.

MicroSec - get bar's microsecond part of the timestamp

Date/Time

SYNTAX **MicroSec()**

RETURNS ARRAY

FUNCTION get bar's microsecond part of the timestamp (0..999)

The function will return zero if data source does not support sub-second resolution

EXAMPLE

```
Title = StrFormat("Timestamp is =
%02.0f:%02.0f:%02.0f.%03.0f%03.0f", Hour(), Minute(), Second(),
MilliSec(), MicroSec() );
```

SEE ALSO HOUR() function , MINUTE() function , SECOND() function , MilliSec() function

References:

The **MicroSec** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

MilliSec - get bar's millisecond part of the timestamp

Date/Time

SYNTAX **MilliSec()**

RETURNS ARRAY

FUNCTION get bar's millisecond part of the timestamp (0..999)

The function will return zero if data source does not support sub-second resolution

EXAMPLE

```
Title = StrFormat("Timestamp is =
%02.0f:%02.0f:%02.0f.%03.0f%03.0f", Hour(), Minute(), Second(),
MilliSec(), MicroSec() );
```

SEE ALSO HOUR() function , MINUTE() function , SECOND() function , MicroSec()() function

References:

The **MilliSec** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Min - minimum value of two numbers / arrays

Math functions

SYNTAX	**Min(ARRAY1, ARRAY2)**
RETURNS	ARRAY
FUNCTION	Returns the smallest of the two parameters.
EXAMPLE	The formula "min(CLOSE, 10)" returns the closing price or 10, whichever is less. The formula "min(-14, 13)" always returns -14.
SEE ALSO	The max() function.

References:

The **Min** function is used in the following formulas in AFL on-line library:

- 'R' Channel
- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Advanced MA system
- ADXVMA
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Andrews Pitchfork
- Andrews PitchforkV3.3
- AR_Prediction.afl
- ATR Study
- Auto Fib Ext&Retracement
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- Bad Tick Trim on 5 sec database
- BBAreacolor&TGLCROSSNEW
- Bman's HaDiffCO
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Caleb Lawrence
- candlestick chart for Volume/RSI/OBV
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI(20) Divergence Indicator
- Channel/S&R and trendlines
- correlerror
- CVR--severe filter
- Cycle Period
- Demand Index
- Dominant Cycle Phase
- Ed Seykota's TSP: Support and Resistance
- Elder Impulse Indicator V2

- elliott wave manual labelling
- EMA Crossover
- Fibonacci Internal and External Retracements
- Fisher Oscillator
- Gann Swing Charts in 3 modes with text
- Graphical sector stock amalysis
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Heatmap V1
- Heikin Ashi Candles
- Heikin Ashi Delta
- Heikin Ashi System
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- HLspread
- interactively test discretionary trading
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- JEEVAN'S SRI CHAKRA
- John Ehler
- Linear Candle
- MACD commentary
- mitalpradip
- MO_CrashZone
- MultiCycle 1.0
- New HL Scanner
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- Parabolic SAR in VBScript
- Parametric Chande Trendscore
- pattenz
- Performance Check
- PF Chart - Close - April 2004
- Pivots for Intraday Forex Charts
- Polyfit Lines
- Probability Density & Gaussian Distribution
- Profit Table (Color Coded)
- Projection Oscillator
- PVT Trend Decider
- Rainbow Oscillator
- Relative Strength Index
- Renko Chart
- Renko Chart
- Revised Renko chart
- Scale Out: Futures
- Schiff Lines
- shailu lunia
- Signal to Noise

- Stops Implementation in AFS
- Stops on percentages
- Three Line Break - TLB
- Trend Lines from 2 points
- Triangle exploration using P&F Chart
- Triangle Search Extended
- Trigonometric Fit - TrigFit with AR for cos / sin
- Twiggs Money Flow
- Twiggs money flow weekly
- TWS trade plotter
- Updated Renko Chart
- Vikram's Floor Pivot Intraday System
- VSTOP (2)
- VSTOP (3)
- Weekly chart
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

Minute
- get current bar's minute

Date/Time
(AmiBroker 40)

SYNTAX	**Minute()**
RETURNS	ARRAY
FUNCTION	Retrieves current bar's minute
EXAMPLE	Hour()*10000 + Minute() * 100 + Second()
SEE ALSO	Hour(), Second(), TimeNum()

References:

The **Minute** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- AR_Prediction.afl
- Backup Data of 1min Interval
- Buyer Seller Force
- Export EOD or Intraday to .csv file
- Export Intraday Data
- For Auto Trading Setup
- High Low Detection code
- How to add IB Option Symbols
- Luna Phase
- New HL Scanner
- Trigonometric Fit - TrigFit with AR for cos / sin
- White Theme

More information:

See updated/extended version on-line.

Month - month

Date/Time
(AmiBroker 3.40)

SYNTAX	**Month()**
RETURNS	ARRAY
FUNCTION	Returns the array with months(1-12)
EXAMPLE	buy = (month() == 1) and day < 3; // buy in January
SEE ALSO	

References:

The **Month** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Auto Trade Step by Step
- Backup Data of 1min Interval
- Binance Data Download
- Coinbase Data Download
- Days to Third Friday
- Expiry day/days - Last thursday of month
- Expiry Thursday for Indian markets
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- For Auto Trading Setup
- FTX Data Download
- High Low Detection code
- How to add IB Option Symbols
- Luna Phase
- Lunar Phases - original
- LunarPhase
- Monthly bar chart
- N-period candlesticks (time compression)
- New HL Scanner
- Next Date Format
- Option Calls, Puts and days till third friday.
- Periodically ReBalance a BUY & HOLD Portfolio
- Prashanth
- Profit Table (Color Coded)
- Relative Strength Multichart of up to 10 tickers
- White Theme
- Wolfe Wave Patterns

More information:

See updated/extended version on-line.

mtRandom
- Mersene Twister random number generator

Statistical functions
(AmiBroker 50)

SYNTAX	**mtRandom(seed = Null)**
	mtRandomA(seed = Null)

RETURNS	NUMBER or ARRAY
FUNCTION	mtRandom(seed = Null) - returns single random number (scalar) in the range [0,1) mtRandomA(seed = Null) - returns array of random numbers in the range of [0,1) seed is random generator seed value. If you don't specify one, the random number generator is automatically initialized with current time as a seed that guarantees unique sequence Both functions use Mersene Twister mt19973ar-cok algorithm. (Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura.) Mersene Twister is vastly superior to C-runtime pseudo-random generator available via Random() function. It has a period of 2^19973 = approx 2.9*10^6012 For more information visit: http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html See also: M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator", ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3--30.
EXAMPLE	printf("Random number: %g", mtRandom());
SEE ALSO	RANDOM() function

References:

The **mtRandom** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically
- How to add IB Option Symbols
- MFE and MAE and plot trades as indicator

More information:

See updated/extended version on-line.

mtRandomA
- Mersene Twister random number generator (array version)

Statistical functions
(AmiBroker 50)

SYNTAX **mtRandomA(seed = Null)**

RETURNS ARRAY

FUNCTION This is array version of mtRandom function

For more details please check mtRandom function.

EXAMPLE

SEE ALSO mtRandom() function

References:

The **mtRandomA** function is used in the following formulas in AFL on-line library:

- CoinToss ver 1

More information:

See updated/extended version on-line.

MxCopy - copy rectangular block from one matrix to another

Matrix functions
(AmiBroker 6.40)

SYNTAX **MxCopy(& dstmatrix, src_matrix, dst_start_row, dst_endrow, dst_start_col, dst_end_col, src_matrix, src_start_row = -1, src_end_row = -1, src_start_col = -1, src_end_col = -1)**

RETURNS NOTHING

FUNCTION Copy rectangular block from one matrix to the other (copy portions of one matrix to the other matrix)

The function works in-place (ie. no allocation occurs - first argument is a reference to existing array and that array content would be overwritten)

Parameters:

- dstmatrix - destination matrix (must be passed as reference using & operator)
- src_matrix - source matrix
- dst_start_row, dst_endrow - start and ending rows in the destination matrix
- dst_start_col, dst_end_col - start and ending columns in the destination matrix
- src_start_row, src_end_row - start and ending rows in the source matrix
- src_start_col, src_end_col - start and ending columns in the destination matrix

src_start/src_end values equal to -1 mean "same value as corresponding dst_start/dst_end value"

As you noticed shape of source and destination "rectangles" does not need to be the same - for example you can copy vertical column into horizontal row, but the number of elements to be copied must be matching. So, to perform a copy the number of columns multiplied by number of rows in source and destination "rectangles" must be the same. In other words:

(dst_end_row-dst_start_row+1)*(dst_end_col-dst_start_col+1) ==
(src_end_row-src_start_row+1)*(src_end_col-src_start_col+1)

EXAMPLE

SEE ALSO

References:

The **MxCopy** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

MxDet - calculate determinant of the matrix

Matrix functions
(AmiBroker 6.10)

SYNTAX **MxDet(mx, method = 0)**

RETURNS NUMBER

FUNCTION The function calculates determinant of the matrix

- method = 0 - auto (use slow method for matrices of upto and including 5x5, fast for larger matrices)
- method = 1 - slow (Laplace expansion method, more accurate)
- method = 2 - fast (LU decomposition, less accurate)

"Slow" method for small matrices (1x1, 2x2, 3x3, 4x4) is actually faster than "fast", equally fast for matrix 5x5 and slower than "fast" method for matrices larger than 5x5. For this reason "auto" method uses "fast" LU method only for matrices larger than 5x5

LU decomposition is fast but subject to higher numerical errors. "Slow" method is slower yet produces much more reliable results.

For example Octave/MatLab that use LU decomposition would say that determinant of singular matrix like this

{ {16, 2, 3, 13}, { 5, 11, 10, 8}, {9, 7, 6, 12}, {4, 14, 15, 1 } }

is -1.4495e-012 due to roundoff errors of LU method.

If you want to calculate determinant using fast (LU decomposition) method, call MxDet with fast parameter set to 2.

CAVEAT: Laplace method has complexity of O(N!) and for this reason, even if you use method = 1, the maximum dimension for this method is limited to 10x10. Matrices larger than that are always calculated using LU method

EXAMPLE

SEE ALSO Matrix() function , MxSolve() function

References:

The **MxDet** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

MxFromString - creates a new matrix out of string

Matrix functions
(AmiBroker 6.10)

SYNTAX **MxFromString("string")**

RETURNS Matrix

FUNCTION creates a new matrix out of string in

- Mathematica/Wolfram list-style: "{ { 1, 2, 3 }, { 4, 5, 6 } }", or
- Matlab/Maple style "[[1, 2, 3], [4, 5, 6]]", or
- GNU Octave comma-semicolon style [1, 2, 3; 4, 5, 6]

EXAMPLE

```
mx = MxFromString("{ { 1, 2, 3 }, { 4, 5, 6 } }" );
```

SEE ALSO Matrix() function , MxToString() function

References:

The **MxFromString** function is used in the following formulas in AFL on-line library:

- Spearman Rank Correlation Coefficient

More information:

See updated/extended version on-line.

MxGetBlock - get rectangular block of items from matrix

Matrix functions
(AmiBroker 6.10)

SYNTAX **MxGetBlock(matrix, startrow, endrow, startcol, endcol, asArray = False)**

RETURNS Matrix or Array

FUNCTION Retrieves items from rectangular submatrix (block) and returns either smaller matrix (when asArray is set to False) or "normal" AFL array (when asArray is set to True). If array has different number of bars, unused elements are filled with Null.

EXAMPLE

```
z = Matrix( 2, 20, 0 );
// first row
z = MxSetBlock( z, 0, 0, 0, 19, Close );
// second row
z = MxSetBlock( z, 1, 1, 0, 19, RSI( 5 ) );
printf("Matrix z ");
printf( MxToString( z ) );

x = MxGetBlock( z, 0, 1, 0, 19, True );

printf("Items are now in regular array (data series): " );
for( i = 0; i < 20; i++ )
printf( NumToStr( x[ i ] ) + " " );

z = MxGetBlock( z, 0, 1, 0, 1 ); // retrieve upper 2x2 submatrix
printf("Upper submatrix z ");
printf( MxToString( z ) );
```

SEE ALSO Matrix() function , MxSetBlock() function

References:

The **MxGetBlock** function is used in the following formulas in AFL on-line library:

- Polyfit Lines

More information:

See updated/extended version on-line.

MxGetSize
- get size of the matrix

Matrix functions
(AmiBroker 60)

SYNTAX **MxGetSize(matrix, dim)**

RETURNS Number

FUNCTION The function retrieves the matrix size in given dimension.

- *matrix* is the matrix variable to query for size
- *dim* is the dimension to query - 0 means rows, 1 means columns

EXAMPLE

```
my_matrix = Matrix( 9, 3 );

rows = MxGetSize( my_matrix, 0 );
columns = MxGetSize( my_matrix, 1 );
```

SEE ALSO Matrix() function

References:

The **MxGetSize** function is used in the following formulas in AFL on-line library:

- channel indicator
- Multi-color Volume At Price (VAP)
- Polyfit Lines

More information:

See updated/extended version on-line.

MxIdentity
- create an identity matrix

Matrix functions
(AmiBroker 60)

SYNTAX	**MxIdentity(size)**
RETURNS	Matrix
FUNCTION	The function creates an identity matrix of defined size (square matrix with rows and columns equal to size argument filled with ones on the main diagonal and zeros elsewhere).
EXAMPLE	m = MxIdentity(5); // create 5x5 identity matrix
SEE ALSO	Matrix() function , MxGetSize() function

References:

The **MxIdentity** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

MxInverse
- calculate inverse matrix

Matrix functions
(AmiBroker 6.10)

SYNTAX	**MxInverse(mx)**
RETURNS	Matrix
FUNCTION	The function calculates inverse of the matrix. Matrix can only be inverted if it is not singular, i.e. when its determinant is not equal zero. Inverse matrix can be used for example to solve linear equation system, but it is faster and slightly more accurate to use MxSolve for this purpose. For more info on usage of inverse matrices see MxSolve documentation.
EXAMPLE	
SEE ALSO	Matrix() function , MxSolve() function

References:

The **MxInverse** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

MxSetBlock - sets values in the rectangular block of matrix cells

Matrix functions
(AmiBroker 6.10)

SYNTAX **MxSetBlock(matrix, startrow, endrow, startcol, endcol, values = 0)**

RETURNS Matrix

FUNCTION Sets values in the rectangular block of cells (rows in the range startrow..endrow and columns in the range startcol..endcol inclusive).

This allows to fill entire or partial rows, columns and all other kind of rectangular areas in the matrix with user specified data. Row and column numbers are zero based.

If values parameter is scalar, all cells in specified block are filled with that value. If values parameter is an array, cells in the block are filled from left to right and from top to bottom with consecutive values taken from that array. If there are more cells in the block than values in the array, the array item counter wraps around to zero and starts taking values from the beginning

Note: the function creates new matrix as a result (so source matrix is unaffected unless you do the assignment of the result back to the original variable)

EXAMPLE

```
// Example 1:
// Create a matrix 6x6
// and fill 4x4 interior (except edges with consecutively increasing
numbers)

y = Matrix( 6, 6, 0 );
y = MxSetBlock( y, 1, 4, 1, 4, Cum(1));
printf("Matrix y ");
printf( MxToString( y ) );

// Example 2:
// Create a matrix 2 rows x 20 columns and fill rows 0, 1 with first
20 values of Close and RSI(5) arrays respectively

z = Matrix( 2, 20, 0 );
// first row
z = MxSetBlock( z, 0, 0, 0, 19, Close );
// second row
z = MxSetBlock( z, 1, 1, 0, 19, RSI( 5 ) );
printf("Matrix z ");
printf( MxToString( z ) );
```

SEE ALSO Matrix() function , MxGetSize() function , MxToString() function

References:

The **MxSetBlock** function is used in the following formulas in AFL on-line library:

- Least Squares Channel Indicator
- Polyfit Lines

More information:

See updated/extended version on-line.

MxSolve
- solves linear equation system A @ X = B

Matrix functions
(AmiBroker 6.10)

SYNTAX **MxSolve(A, B)**

RETURNS Matrix

FUNCTION The function solves linear equation system A@X = B.

A needs to be square matrix NxN
B has to have N rows and at least one column (vertical vector).

Then calling

X = MxSolve(A, B);

would give vertical vector holding solution of the system of equations A @ X = B

B can also be a matrix,with each of its column representing different vector B. This way single call to MxSolve can solve several systems with same matrix A but different right hand vectors. If B is a matrix NxM then MxSolve will produce result also having NxM cells with each column representing single solution.

(Highly) Technical note about numerical precision

Despite the fact that both MxSolve and MxInverse use double precision arithmetic solving/inverting matrices is subject to numerical precision of double IEEE and for example zero result may come up as something like 1.4355e-16 (0.0000000000000001) due to the fact that double precision is still limited in accuracy (16 digits).

The result of

X = MxInverse(A) @ B;

although mathematically the same as solving the system of equations, would yield slightly different result because if you do the inverse the returned matrix is converted back to single precision and matrix product is performed with single precision. When you use MxSolve you are performing all calcs using 64-bit (double) precision and only end result is converted back to single precision. So for example polynomial fit code works better with MxSolve than MxInverse.

EXAMPLE

```
// Example 1:

A = MxFromString("[ 1, 1, 1, 1; 0, 2, 5, -1; 2, 5, -1, 1; 2, 2, 2, 1
]");
B = MxFromString("[ 7; -5; 28; 13 ]" ); // single vertical vector B

printf( "Solving A * X = B " );
printf("Matrix A ");
printf( MxToString( A ) );
printf(" Matrix B ");
printf( MxToString( B ) );

X = MxSolve( A, B );

printf(" Solution X ");

// Example 2:

A = MxFromString("[ 1, 1, 1, 1; 0, 2, 5, -1; 2, 5, -1, 1; 2, 2, 2, 1
]");
B = MxFromString("[ 7, 14 ; -5, -10; 28, 56; 13, 26 ]" ); // 2
right-hand side vertical vectors

printf( "Solving A * X = B " );
printf("Matrix A ");
printf( MxToString( A ) );
printf(" Matrix B ");
printf( MxToString( B ) );

X = MxSolve( A, B );

printf(" Solutions X ");

printf( MxToString( X ) ); // two solutions

////////////////////////////////////////
// Example 3
// N-th order Polynomial Fit example
/////
order = Param( "n-th Order", 10, 1, 16, 1 );
length = 60;

lvb = BarCount - 1;
fvb = lvb - length;

yy = Matrix( length + 1, 1, 0 );
xx = Matrix( length + 1, order + 1, 1 );

yy = MxSetBlock( yy, 0, length, 0, 0, Ref( C, fvb ) );

x = BarIndex() - length/2;

for( j = 1; j <= order; j++ )
{
     xx = MxSetBlock( xx, 0, length, j, j, x ^ j );
}

xxt = MxTranspose( xx );
aa = MxSolve( xxt @ xx, xxt ) @ yy;
//aa = MxInverse( xxt @ xx ) @ xxt @ yy; // alternative way
```

```
if( aa ) // check if matrix is not null (so solution exists)
{
     rr = Null; // store the fit in rr
   for( i = fvb; i <= lvb; i++ )
     {
       rr[i] = aa[0][0];

       for( j = 1; j <= order; j++ )
       {
           rr[i] += aa[j][0] * x[ i - fvb ] ^ j;
       }
     }

     if( IsNan( rr[ fvb ] ) )
     {
       // our polynomial yields infinite or not-a-number result due
to overflow/underflow
      Title = "Polyfit failed. The order of polynomial is too High";
     }
     else
     {
       SetChartOptions( 0, chartShowDates );
       SetBarFillColor( IIf( C > O, ColorRGB( 0, 75, 0 ), IIf( C <=
O, ColorRGB( 75, 0, 0 ), colorLightGrey ) ) );
       Plot( rr, "rr", colorWhite, styleLine | styleThick);
     }
}
else
{
   Title = "Matrix is singular. The order of polynomial is too
high";
}

Plot( C, "", IIf( C > O, ColorRGB( 0, 255, 0 ), IIf( C <= O,
ColorRGB( 255, 0, 0 ), colorLightGrey ) ), styleDots | styleNoLine
);
```

SEE ALSO Matrix() function , MxGetSize() function , MxIdentity() function , MxTranspose() function , MxToString() function

References:

The **MxSolve** function is used in the following formulas in AFL on-line library:

- Least Squares Channel Indicator
- Polyfit Lines

More information:

See updated/extended version on-line.

MxSort - sorts the matrix

Matrix functions
(AmiBroker 6.10)

SYNTAX **MxSort(mx, dim = -1, ascening = True)**

RETURNS Matrix

FUNCTION Sorts all items in a matrix

When dim == -1 (the default) it would sort:

- a row if there is only one row (vector is horizontal)
- a column if there is only one column (vector is vertical)
- each column separately if there are more rows and columns than one (so we have actual 2D matrix).

When dim == 0 the function sorts the items in each row separately
When dim == 1 the function sorts the items in each column separately

EXAMPLE

```
m = MxFromString("[ 9, 5, 6; 8, 7, 3 ]");

printf( MxToString( m ) + " " );

printf("%g, %g ", MxGetSize( m, 0 ), MxGetSize( m, 1 ) );

m2 = MxSort( m, 0 ) ;

printf( MxToString( m2 ) + " " );

m3 = MxSort( m, 1 ) ;

printf( MxToString( m3 ) + " " );
```

SEE ALSO Matrix() function , MxSortRows() function , MxTranspose() function

References:

The **MxSort** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

MxSortRows - sort the rows of the matrix

Matrix functions
(AmiBroker 6.10)

SYNTAX **MxSortRows(mx, ascending = True, col1 = 0, col2 = -1, col3 = -1)**

RETURNS Matrix

FUNCTION Sorts the rows of the matrix in ascending/descending order of the col1 column. When the col1 column has equal values, SortRows sorts according to the col2 and col3 columns in succession (if col2 and col3 are specified and >= 0).Column numbers are zero based.

Hint: if you want to sort columns instead you can Transpose/Sort rows/Transpose back.

EXAMPLE

```
m = MxFromString("[ 9, 1, 6; 40, 30, 20; 8, 7, 3; 3, 5, 1 ]");

printf("Input matrix ");

printf( MxToString( m ) + " " );

printf("Rows %g, Cols %g ", MxGetSize( m, 0 ), MxGetSize( m, 1 ) );

printf("Sorting every row separately ");
m2 = MxSort( m, 0 ) ;

printf( MxToString( m2 ) + " " );

printf("Sorting every column separately ");
m3 = MxSort( m, 1 ) ;

printf( MxToString( m3 )+ " ");

printf("Sorting rows by contents of first column ");
m4 = MxSortRows( m, True, 0 ) ;

printf(MxToString( m4 )+ " ");

printf("Sorting rows by contents of second column ");
m5 = MxSortRows( m, True, 1 ) ;

printf(MxToString( m5 )+ " ");
```

SEE ALSO Matrix() function , MxTranspose() function

References:

The **MxSortRows** function is used in the following formulas in AFL on-line library:

- Spearman Rank Correlation Coefficient
- Volume based support resistance price finder

More information:

See updated/extended version on-line.

MxSum
- calculate grand sum of the matrix

Matrix functions
(AmiBroker 6.20)

SYNTAX	**MxSum(matrix)**
RETURNS	NUMBER
FUNCTION	The function calculates sum of all elements of matrix (grand sum)
EXAMPLE	
SEE ALSO	MxDet() function , MxFromString() function , MxGetBlock() function , MxGetSize() function , MxIdentity() function , MxInverse() function , MxSetBlock() function , MxSolve() function , MxSort() function , MxSortRows() function , MxToString() function , MxTranspose() function

References:

The **MxSum** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

MxToString
- convert matrix to string

Matrix functions
(AmiBroker 6.10)

SYNTAX	**MxToString(mx)**
RETURNS	String
FUNCTION	Creates string out of matrix variable in the Wolfram list style like this (for 3x3 matrix): "{ { x00, x01, x02 }, { x10, x11, x12 }, { x20, x21, x22 } }"
EXAMPLE	
SEE ALSO	Matrix() function , MxGetSize() function , MxTranspose() function

References:

The **MxToString** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- Polyfit Lines

More information:

See updated/extended version on-line.

MxTranspose
- creates transpose of an input matrix

Matrix functions
(AmiBroker 60)

SYNTAX	**MxTranspose(matrix)**
RETURNS	Matrix
FUNCTION	The function creates a transpose of an input matrix matrix. Transpose of a matrix is a new matrix whose rows are the columns of the original.
EXAMPLE	
SEE ALSO	Matrix() function , MxGetSize() function , MxIdentity() function

References:

The **MxTranspose** function is used in the following formulas in AFL on-line library:

- Least Squares Channel Indicator
- Polyfit Lines

More information:

See updated/extended version on-line.

Name - ticker symbol

Information / Categories
(AmiBroker 3.10)

SYNTAX	**Name()**
RETURNS	STRING
FUNCTION	It is used to display the stock short name (ticker)
EXAMPLE	name()
SEE ALSO	

References:

The **Name** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3 Price Break
- 3TF Candlestick Bar Chart
- AccuTrack
- Advanced MA system
- Advanced Search and Find
- Advanced Trend Lines with S & R
- ADX Indicator - Colored
- AFL Example
- AFL Example - Enhanced
- AFL to Python COM Link
- AFL-Excel
- AFL_Glossary_3
- AFL_Glossary_Converter
- Against all odds
- Alert Output As Quick Rewiev
- ALJEHANI
- AllinOneAlerts - Module
- Alpha and Beta and R_Squared Indicator
- Alternative ZIG type function, multi TF
- AO+Momentum
- Aroon Indicators
- Auto Trade Step by Step
- Auto-Optimization Framework
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Dollar Price Volatility Exploration
- Average Price Crossover
- babaloo chapora
- Backup Data of 1min Interval
- Baseline Relative Performance Watchlist charts V2
- BBAreacolor&TGLCROSSNEW
- Bollinger - Keltner Bands

- Bull Fear / Bear Fear
- Bullish Percent Index 2004
- Button trading using AB auto trading interface
- Caleb Lawrence
- candlestick chart for Volume/RSI/OBV
- CCI Woodies Style
- CCI(20) Divergence Indicator
- channel indicator
- Channel/S&R and trendlines
- Cole
- Color Coded Short Term Reversal Signals
- colored CCI
- Commodity Channel Index
- Commodity Selection Index (CSI)
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- Coppock Trade Signal on Price Chart
- Darvas Amibroker
- Dave Landry PullBack Scan
- De Mark's Range Projection
- DEBAJ
- Dinapoli Guru Commentary
- DiNapolis 3x Displaced Moving Averages
- Double Smoothed Stochastic from W.Bressert
- Double top detection
- DPO with shading
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- Elder's SafeZone Stop
- ElderSafeZoneStopLong
- ElderSafeZoneStopShort
- elliott wave manual labelling
- Elliott Wave Oscillator
- EMA Crossover Price
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- FastStochK FullStochK-D
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- For Auto Trading Setup
- Fre
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Futures - Dollar Move Indicator
- Futures - Dollar Move Today Indicator
- Gann level plotter
- Gfx Toolkit
- Gordon Rose
- Graphical sector analysis

- Graphical sector stock amalysis
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- hassan
- Heikin Ashi Candles
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- Herman
- High Low Detection code
- How to add IB Option Symbols
- IB Backfiller
- IBD relative strength database Viewer
- Improved NH-NH scan / indicator
- In Watch List
- Indicator Explorer (ZigZag)
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- Larry William's Volatility Channels
- Last Five Trades Result Dashboard – AFL code
- Least Squares Channel Indicator
- Linear Candle
- MA Difference 20 Period
- MACD commentary
- MACD Histogram - Change in Direction
- Main price chart with Rainbow & SAR
- Manual Bracket Order Trader
- Market Facilitation Index VS Volume
- MFE and MAE and plot trades as indicator
- mitalpradip
- Mndahoo ADX
- Modified-DVI
- Monthly Coppock Guide
- Moving Averages NoX
- MS Darvas Box with Exploration
- N Line Break Chart
- Nadaraya-Watson Envelope
- Neural Network Powered Smooth/Predictive RSI V2
- New HL Scanner
- Nick
- Noor_Doodie
- Now Send Push Notifications From Amibroker
- Optimal Weights
- P&F Chart - High/Low prices Sept2003
- Parabolic SAR in JScript
- Parabolic SAR in VBScript
- ParabXO
- pattenz
- Periodically ReBalance a BUY & HOLD Portfolio
- Peter Cooper

- Peterson
- PF Chart - Close - April 2004
- Pivot End Of Day Trading System
- Pivot Finder
- Pivot Point and Support and Resistance Points
- Pivots for Intraday Forex Charts
- Point & figure Chart India Securities
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- prakash
- Price with Woodies Pivots
- Probability Calculator
- Profit Table (Color Coded)
- PVT Trend Decider
- QP2 Float Analysis
- Rainbow Charts
- Rainbow Oscillator
- Random Walk Index
- Range Filter - Trading Strategy
- Ranking Ticker WatchList
- Rea Time Daily Price Levels
- Reconnect TWS
- Relative Strength
- Relative Strength Index
- Rene Rijnaars
- Renko Chart
- Renko Chart
- Revised Renko chart
- RI - Auto Trading System
- Robert Antony
- ROC of MACD Weekly
- RSI indicator with Upper & Lower Zone Bars
- RSI of Weekly Price Array
- RSI styleClipMinMax
- RSIS
- Say Notes
- Scan New High and New Low
- shailu lunia
- Shares To Buy Price Graph
- Simple Momentum
- Stan Weinstein strategy
- STD_STK Multi
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Steve Woods' Float Channel Lines
- Stochastic Fast%K and Full
- Stochastics Trendlines
- StochD_StochK Single.afl
- Stock price AlertIf
- Stress with SuperSmoother
- Sun&Cloud
- Super Trend Indicator

- Support Resistance levels
- suresh
- TD REI
- TD Sequential
- The D_oscillator
- Three Line Break - TLB
- Trend Exploration: Count Number of New Highs
- Trend Exploration: Slope Moving Average
- Triangular Moving Average
- Triangular Moving Average new
- TRIX
- TSV
- Twiggs Money Flow
- TWS trade plotter
- Ultimate plus
- UltraEdit editor highlight wordfile
- Updated Renko Chart
- ValueChart
- Vikram's Floor Pivot Intraday System
- Visi-Trade
- visual turtle trading system
- Volume Divided Histogram
- Volume Occilator
- VolumeDivAvgV3_Study_BrS
- Weinberg's The Range Indicator
- White Theme
- William's Alligator System II
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- Woodies CCI
- Zig Zag Indicator with Valid Entry and Exit Points
- ZigZag filter rewrited from scratch in AFL
- ZigZag Hi Lo Barcount
- ZigZag Retracements

More information:

See updated/extended version on-line.

NormDist - normal distribution function

Statistical functions
(AmiBroker 6.20)

SYNTAX **NormDist(x, mean = 0, sigma = 1, cumulative = True)**

RETURNS NUMBER

FUNCTION The function calculates Normal distribution

The function uses hi-precision algorithm that gives accurate results even for extreme left and right tails (+5)

The results are compatible with Excel 2003 and later which introduced higher precision NORMDIST than previous versions (Excel 2002 and earlier).

EXAMPLE

```
x = 19;
m = 2;
s = 9;

printf("PDF %g CDF %g", NormDist( x, m, s, False ), NormDist( x, m,
s, True ) );

x = -7;
m = 0;
s = 1;

printf("PDF %g CDF %g", NormDist( x, m, s, False ), NormDist( x, m,
s, True ) );
```

SEE ALSO

References:

The **NormDist** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

NoteGet
- retrieves the text of the note

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX **NoteGet("Symbol")**

RETURNS STRING

FUNCTION Retrieves note linked to "symbol". If symbol is "" (empty string) then current symbol is used. If symbol is "" (empty string) then current symbol is used.

EXAMPLE
```
"You have entered the following text in the notepad" + NoteGet("");
```

SEE ALSO NoteSet() function

References:

The **NoteGet** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

NoteSet - sets text of the note

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX **NoteSet("Symbol", "Text..");**

RETURNS NUMBER

FUNCTION Sets text of the note linked to "symbol".
If symbol is "" (empty string) then current symbol is used.

If you overwrite note from AFL level that is opened at the same time in Notepad editor the editor will ask you (when you switch the focus to it) if it should reload new text or allow to save your manually entered text.

Returns True (1) on success, 0 on failure.

EXAMPLE

```
NoteSet("AMD", "Jun 15, 2004: AMD will deliver its first multi-core
processors next year");
```

SEE ALSO NoteGet() function

References:

The **NoteSet** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Now
- gets current system date/time

Date/Time
(AmiBroker 4.30)

SYNTAX **Now(format = 0)**

RETURNS STRING or NUMBER

FUNCTION Returns current date / time in numerous of formats:

- format = 0 - returns string containing current date/time formatted according to system settings
- format = 1 - returns string containing current date only formatted according to system settings
- format = 2 - returns string containing current time only formatted according to system settings
- format = 3 - returns DATENUM number with current date
- format = 4 - returns TIMENUM number with current time
- format = 5 - returns DATETIME number with current date/time
- format = 6 - returns current DAY (1..31)
- format = 7 - returns current MONTH (1..12)
- format = 8 - returns current YEAR (four digit)
- format = 9 - returns current DAY OF WEEK (1..7, where 1=Sunday, 2=Monday, and so on)
- format = 10 - returns current DAY OF YEAR (1..366)

EXAMPLE AddTextColumn(Now(), "Current time");

SEE ALSO DATENUM() function , DATETIME() function , DATE() function , TIMENUM() function

References:

The **Now** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- A simple AFL Revision Control System
- AFL Timing functions
- AFL_Glossary_1
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI(20) Divergence Indicator
- channel indicator
- Create a list of functions in your program
- Extract specific lines of code from your program
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Gfx Toolkit
- Harmonic Pattern Detection
- Heatmap V1
- Herman
- Least Squares Channel Indicator

- MFE and MAE and plot trades as indicator
- Next Date Format
- Time Left in Bar
- Time Left to Current Bar
- TWS auto-export Executions-file parser
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

NullCount
- count consecutive Null values

Miscellaneous functions
(AmiBroker 5.90)

SYNTAX **NullCount(array, mode = 1)**

RETURNS NUMBER

FUNCTION Counts the number of consecutive nulls at the beginning of the array (mode = 1), at the end of the array (mode=2), from both ends (mode=3) and all nulls in the array (including non-consecutive) (mode=0)

EXAMPLE
```
x = MA( C, 20 );
printf("Empty values at the beginning = " + NullCount( x ) );
```

SEE ALSO

References:

The **NullCount** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit
- Polyfit Lines

More information:

See updated/extended version on-line.

NumToStr
- convert number to string

String manipulation
(AmiBroker 4.50)

SYNTAX **NumToStr(NUMBER, *format* = 1.3, *separator*=True, roundAndPad = False)**
NumToStr(ARRAY, *format* = 1.3, *separator*=True, roundAndPad = False)

RETURNS STRING

FUNCTION It is used to convert numeric value of NUMBER or ARRAY to string.

The second parameter - *format* - allows you to control output formatting (decimal places and leading spaces). The integer part of the number controls minimum number of characters used to display the number (if you specify high number the output will be space-padded). The fractional part defines how many decimal places to display, for example 1.0 - will give you a number without fractional part at all, and 1.2 - will give two digits past the decimal point

There is also a special format constant formatDateTime that allows to convert date/time returned by DateTime() function formatted according to Windows regional settings.

Since 6.20 also supported is formatDateTimeISO which will return date time in international format (ISO) i.e, YYYY-MM-DD HH:MM:SS

Third parameter *separator* (true by default) controls if thousand separator is added or not. Thousands separator is definable in Tools->Preferences->Misc.

Fourth parameter *roundAndPad* controls whenever function rounds output beyond 7th significant digit (and pads the rest with zeros), By default rounding is OFF now because it was off in 5.90 and earlier and rounding introduced in 5.91 could confuse old time users

Note: **NumToStr** is a synonym for **WriteVal** function.

EXAMPLE 1. Simple use (no custom format)

```
printf( NumToStr( StochK(39) - StochK(12)) );
```

2. Display rate of change with 2 decimal digits and % appened to the end

```
printf( NumToStr( ROC( Close, 20 ), 1.2 ) + "%%");
```

3. Display date/time according to regional settings

```
printf( NumToStr( DateTime(), formatDateTime ));
```

SEE ALSO WRITEVAL() function , StrToNum() function , DateTimeFormat() function

References:

The **NumToStr** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 2 Timeframes Candlestick Bar Chart
- 3 Price Break

- 3TF Candlestick Bar Chart
- A simple AFL Revision Control System
- Advanced Trend Lines with S & R
- AFL Timing functions
- AFL_Glossary_1
- AFL_Glossary_Converter
- ALJEHANI
- Aroon The Advisor
- AR_Prediction.afl
- Auto Fib Ext&Retracement
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Dollar Price Volatility Exploration
- Average Price Crossover
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bottom Fisher Exploration
- Button trading using AB auto trading interface
- Candle Identification
- Candle Stick Analysis
- Candle Stick Demo
- channel indicator
- Computing Cointegration and ADF Dashboard
- Congestions detection
- Continuous Contract Rollover
- Coppock Trade Signal on Price Chart
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- DateNum_DateStr
- DEBAJ
- elliott wave manual labelling
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fibonacci Moving averages
- For Auto Trading Setup
- Fre
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- Gordon Rose
- Halftrend
- Heatmap V1
- Heikin Ashi Candles
- High Low Detection code
- How to add IB Option Symbols
- Hurst "Like" DE
- IBD relative strength database ranker
- IBD relative strength database Viewer
- JEEVAN'S SRI CHAKRA

- lastNDaysBeforeDate
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Market Breadth Chart-In-Chart
- Market Profile
- MFE and MAE and plot trades as indicator
- Nadaraya-Watson Envelope
- New HL Scanner
- nth (1 - 8) Order Polynomial Fit
- Option Calls, Puts and days till third friday.
- Ord Volume
- pattenz
- Pivots And Prices
- Point & figure Chart India Securities
- prakash
- Probability Density & Gaussian Distribution
- Range Filter - Trading Strategy
- Ranking and sorting stocks
- Square of Nine Roadmap Charts
- suresh
- TAPE READING
- TD Sequential
- Time Left in Bar
- Time Left to Current Bar
- tomy_frenchy
- Trigonometric Fit - TrigFit with AR for cos / sin
- TWS auto-export Executions-file parser
- Visi-Trade
- Volume Color with Dynamic Limit
- Volume Occilator
- White Theme
- WLBuildProcess
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

NVI
- negative volume index

Indicators

SYNTAX	**nvi()**
RETURNS	ARRAY
FUNCTION	Calculates the Negative Volume Index.
EXAMPLE	
SEE ALSO	The pvi() function

References:

The **NVI** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Nz - Null (Null/Nan/Infinity) to zero

Miscellaneous functions
(AmiBroker 4.30)

SYNTAX **Nz(x, valueifnull = 0)**

RETURNS NUMBER, ARRAY

FUNCTION Converts Null/Nan/Infinity values to zero (or user defined value)

x can be number or array.

You can use the Nz function to return zero, or another specified value when argument x is Null or Nan or Infinite.

For example, you can use this function to convert a Null (empty) value to another value and prevent it from propagating through an expression. If the optional valueifnull argument is included, then the Nz function will return the value specified by that argument if the x argument is Null (or Nan or Infinity).

Since AmiBroker 6.90 valueifnull argument can be an array.

EXAMPLE You can use the Nz function as an alternative to the IIf function.

Instead of:

```
varTemp = IIf( IsFinite( (H-L)/(C-L) ), (H-L)/(C-L), 0 );
```

You can write:

```
varTemp = Nz( (H-L)/(C-L) );
```

SEE ALSO

References:

The **Nz** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- A simple AFL Revision Control System
- ADXVMA
- AFL to Python COM Link
- Alphatrend
- AR_Prediction.afl
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- Candle Stick Analysis
- channel indicator
- Cycle Period
- Dominant Cycle Phase
- Dynamic Momentum Index

- Dynamic Momentum Index
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- elliott wave manual labelling
- For Auto Trading Setup
- Gfx Toolkit
- GFX ToolTip
- Harmonic Pattern Detection
- Heatmap V1
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- interactively test discretionary trading
- John Ehler
- Least Squares Channel Indicator
- Price Chart - Fundamental
- Profit Table (Color Coded)
- Range Filter - Trading Strategy
- Signal to Noise
- Stochastic RSI
- Stress with SuperSmoother
- TAPE READING
- Trigonometric Fit - TrigFit with AR for cos / sin
- VolumeDivAvgV3_Study_BrS
- Woodie's CCI Panel Full Stats
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

OBV - on balance volume

Indicators

SYNTAX	**obv()**
RETURNS	ARRAY
FUNCTION	Calculates the On Balance Volume indicator.
EXAMPLE	
SEE ALSO	

References:

The **OBV** function is used in the following formulas in AFL on-line library:

- candlestick chart for Volume/RSI/OBV
- OBV with Linear Regression
- Stochastic OBV and Price Filter

More information:

See updated/extended version on-line.

Optimize - define optimization variable

Trading system toolbox
(AmiBroker 3.70)

SYNTAX	**Optimize("*description*", *default, min* , *max, step*)**
RETURNS	NUMBER
FUNCTION	Defines optimization process parameters. With normal backtesting, scanning, exploration and comentary modes the optimize function returns *default* value, so the above function call returns *default*; In optimization mode optimize function returns successive values from *min* to *max* (inclusively) with *step* stepping. "*description"* is a string that is used to identify the optimization variable and is displayed as a column name in the optimization result list. *default* is a default value that optimize function returns in exploration, indicator, commentary, scan and normal back test modes *min* is a minimum value of the variable being optimized *max* is a maximum value of the variable being optimized *step* is an interval used for increasing the value from *min* to *max*
EXAMPLE	variable = optimize("my optimization var", 9, 2, 20, 1);
SEE ALSO	

Comments:

Herman van den Bergen psytek [at] magma.ca 2003-06-09 05:23:31	You can Optimize parameters with custom number series by using the numbers generated by the Optimize() function as an index to access numbers in a custom array. Here is an example using a custom array FB[] of Fibonacci numbers:
	
	FB[0] = 0.0; FB[1] = 23.6; FB[2] = 38.2; FB[3] = 50.0; FB[4] = 61.8; FB[5] = 100; FB[6] = 161.8; FB[7] = 261.8; FB[8] = 423.6;
	FBindex = Optimize("FBindex",0,0,8,1);
	FibNum = FB[FBindex];
	... place your Code using FibNum here ...

Herman van den Bergen psytek [at] magma.ca 2003-07-20 17:26:08	You can refresh your Equity chart after each Optimization step and observe (like a slide show) how the linearity of your Equity curve is effected by adding these two lines to the very end of your code:
	
	AB = CreateObject("Broker.Application");
	AB.RefreshAll();
	
	Important note: Do not use in commentary, interpretation, or indicator builder because it will cause loop. (Thanks for the tip TJ!)

Graham Kavanagh gkavanagh [at] e-wire.net.au 2004-08-21 23:31:39	When optimising for 2 or more variables make sure you have different names for each variable.
	eg
	x = Optimize("Short",5,5,10,1);
	y = Optimize("Short",15,25,55,1);
	
	I made mistake of copy/paste and did not change the optimize name (as above) within the brackets and got all zeroes as results.
	
	This below gets results

	x = Optimize("Short",5,5,10,1);
	y = Optimize("Long",15,25,55,1);
	
	Graham

Tomasz Janeczko
tj --at-- amibroker.com
2006-12-12 11:30:18
	Some asked for function that combines Param() and Optimize(). Here it is:
	
	function ParamOptimize(pname, defaultval, minv, maxv, step)
	{
	return Optimize(pname,
	Param(pname, defaultval, minv, maxv, step),
	minv, maxv, step);
	}

References:

The **Optimize** function is used in the following formulas in AFL on-line library:

- Advanced MA system
- ADXbuy
- ATR Study
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Awsome Oscilator
- BBAreacolor&TGLCROSSNEW
- Bollinger band normalization
- Bull Fear / Bear Fear
- CoinToss ver 1
- Dahl Oscillator modified
- danningham penetration
- Darvas Wisestocktrader
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- Effective Swing Indicator
- ekeko price chart
- EMA Crossover
- Evaluating Candle Patterns in a trading system
- FastStochK FullStochK-D
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Fund Screener
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Inter-market Yield Linear Regression Divergence
- Lagging MA-Xover
- MACD optimize
- Moving Averages NoX
- Optimal Weights
- OptimizationBatch.js
- Perceptron
- Peterson
- Plot the Equity Curve without Backtesting?
- Projection Oscillator

- Rapid Prototyping Method for System Development
- RUTVOL timing signal with BB Scoring routine
- SectorRSI
- Stan Weinstein strategy
- STD_STK Multi
- Stochastic Fast%K and Full
- Stochastic optimize
- StochD_StochK Single.afl
- Super Trend Indicator
- The D_oscillator
- Trend Continuation Factor
- Trend Following System
- Trend Trigger Factor
- TRIX
- Vivek Jain
- Volatility System
- Volume Zone Oscillator
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

OptimizerSetEngine
- select external optimization engine

Trading system toolbox
(AmiBroker 5.20)

SYNTAX **OptimizerSetEngine("name")**

RETURNS NOTHING

FUNCTION The function selects external optimization engine defined by name. The following optimization engines are shipped with AmiBroker as of version 5.20

- Standard Particle Swarm Optimizer ("spso")
- Tribes (improved PSO) ("trib")
- Covariance Matrix Adaptation Evolutionary Strategy ("cmae")

New engines may be added in the future.

EXAMPLE OptimizerSetEngine("spso");

SEE ALSO

References:

The **OptimizerSetEngine** function is used in the following formulas in AFL on-line library:

- Optimal Weights
- Perceptron
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

OptimizerSetOption - set the value of external optimizer engine parameter

Trading system toolbox
(AmiBroker 5.20)

SYNTAX	**OptimizerSetOption("name", value)**
RETURNS	NOTHING
FUNCTION	The function set additional parameters for external optimization engine. The parameters are engine-dependent. For example SPSO, TRIBES and CMAE optimizers support "Runs" (number of runs) and "MaxEval" (maximum evaluations (tests)per single run) parameters.
EXAMPLE	OptimizerSetOption("Runs", 2);
SEE ALSO	OptimizerSetEngine() function

References:

The **OptimizerSetOption** function is used in the following formulas in AFL on-line library:

- Perceptron

More information:

See updated/extended version on-line.

OscP - price oscillator

Indicators

SYNTAX	**OscP(*fast* , *slow*)**
RETURNS	ARRAY
FUNCTION	Calculates price oscillator based on exponential moving averages
EXAMPLE	oscp(9, 18)
SEE ALSO	

References:

The **OscP** function is used in the following formulas in AFL on-line library:

- Indicator Explorer (ZigZag)
- MACD commentary

More information:

See updated/extended version on-line.

OscV - volume oscillator

Indicators

SYNTAX	**OscV(** ***fast***, ***slow*** **)**
RETURNS	ARRAY
FUNCTION	Calculates volume oscillator based on exponential moving averages
EXAMPLE	oscv(9, 18)
SEE ALSO	

References:

The **OscV** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Outside - outside bar

Basic price pattern detection

SYNTAX	**Outside()**
RETURNS	ARRAY
FUNCTION	Gives "true" (or 1) when an outside day occurs
EXAMPLE	outside()
SEE ALSO	

References:

The **Outside** function is used in the following formulas in AFL on-line library:

- AC+ acceleration
- AO+ Momentum indicator
- Fund Screener
- Gann Swing Chart
- Gann Swing Charts in 3 modes with text
- Vic Huebner
- Williams Alligator system

More information:

See updated/extended version on-line.

Param
- add user user-definable numeric parameter

Exploration / Indicators
(AmiBroker 4.30)

SYNTAX **Param(*"name", defaultval, min, max, step, sincr* = *0*)**

RETURNS NUMBER

FUNCTION Adds a new user-definable parameter, which will be accessible via Parameters dialog :
right click over chart pane and select "Parameters" or press Ctrl+R allows to change chart parameters - changes are reflected immediatelly.

- "name" - defines parameter name that will be displayed in the parameters dialog
- defaultval - defines default value of the parameter
- min, max - define minimum and maximum values of the parameter
- step - defines minimum increase of the parameter via slider in the Parameters dialog
- sincr - automatic section increment value (used by drag-drop interface to increase default values for parameters)

WARNING: default/min/max/step parameters have to be CONSTANT numbers. This is because these values are cached and are not re-read during subsequent formula evaluations.

IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).

EXAMPLE Sample code 1:

```
ticker = ParamStr( "Ticker", "MSFT" );
sp = Param( "MA Period", 12, 2, 100 );
PlotForeign( ticker, "Chart of "+ticker, ParamColor( "Price Color",
colorLightYellow ), styleCandle );
Plot( MA( Foreign( ticker, "C" ), sp ), "MA(" + WriteVal( sp, 1.0 )
+ ")", ParamColor( "MA Color", colorRed ) );
```

Sample code 2:

```
sp = Param( "RSI Period", 12, 2, 100 );
r = RSI( sp );
Plot( r, "RSI("+WriteVal(sp,1.0)+")", ParamColor("RSI Color",
colorRed ) );

Buy = Cross( r, 30 );
Sell = Cross( 70, r );

PlotShapes( shapeUpArrow * Buy + shapeDownArrow * Sell, IIf( Buy,
colorGreen, colorRed ) );
```

SEE ALSO PARAMCOLOR() function , PARAMSTR() function PARAMCOLOR() function , ParamTime() function , ParamDate() function

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2006-02-19 06:18:23	Note that Parameters are INDEPENDENT for each chart pane and for Automatic Analysis window. In Automatic Analysis window parameters can be modified using "Parameters" button and they are independent from ones you use for any chart.
Tomasz Janeczko tj --at-- amibroker.com 2006-02-19 06:19:59	To change the parameters for the indicator, please click with RIGHT mouse button over chart pane and select "Parameters" from the menu.

References:

The **Param** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 2 Timeframes Candlestick Bar Chart
- 3 Price Break
- 3 ways to use RMI in one script
- 3TF Candlestick Bar Chart
- Adaptive Centre of Gravity
- Adaptive Cyber Cycle
- Adaptive Laguerre Filter, from John Ehlers
- Adaptive Price Channel
- Adaptive Relative Vigour Index
- Add SL/TGT other params to any strategy
- Advanced MA system
- Advanced Trend Lines with S & R
- Adverse Move Ratio
- ADX Indicator - Colored
- ADXVMA
- AFL Example
- AFL Example - Enhanced
- AFL Timing functions
- AFL to Python COM Link
- AFL_Glossary_Converter
- ALJEHANI
- AllinOneAlerts - Module
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- An n bar Reversal Indicator
- Andrews Pitchfork
- Andrews PitchforkV3.3
- Animated BackGround 1.1
- AO+Momentum
- Application of Ehler filter
- Arnaud Legoux Moving Average (ALMA)
- Aroon The Advisor

- AR_Prediction.afl
- Auto Fib Ext&Retracement
- Auto Trade Step by Step
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Dollar Price Volatility Exploration
- Average Price Crossover
- babaloo chapora
- Bad Tick Trim on 5 sec database
- Baseline Relative Performance Watchlist charts V2
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Better Bollinger Bands
- Black Scholes Option Pricing
- Bman's HaDiffCO
- bolingerbands
- Bollinger - Keltner Bands
- Bollinger Band Squeeze
- Bollinger Fibonacci Bands
- bonlinger bands
- Bottom Fisher Exploration
- Brian Wild
- Buff Volume Weighted Moving Averages
- Bull/Bear Volume
- Button trading using AB auto trading interface
- Caleb Lawrence
- Candle Stick Analysis
- Candle Stick Demo
- candlestick chart for Volume/RSI/OBV
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI(20) Divergence Indicator
- Centre of Gravity
- Chandelier Exit
- Chandelier Exit
- Chandelier Exit or Advanced Trailing Stop
- channel indicator
- Channel/S&R and trendlines
- Chart Zoom
- Color MACD Histogram Changes
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- com-out
- Commodity Selection Index (CSI)
- Computing Cointegration and ADF Dashboard
- Congestions detection

- ConnorsRSI
- Controlling Height of Volume Bars
- Coppock Histogram
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Coral Trend Indicator
- Cyber Cycle
- Cybernertic Hilbert Sine Wave
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period
- Darvas Amibroker
- Dave Landry PullBack Scan
- Day Bar No
- DEBAJ
- Demand Index
- DiNapoli Detrended Oscillator
- Dinapoli Perferred Stochastic
- Dominant Cycle Phase
- Double Super Trend System
- DPO with shading
- DT Oscillator
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Effective Swing Indicator
- Ehlers Center of Gravity Oscillator
- Ehlers Hilbert Transformer Indicator
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- ekeko price chart
- EKEKO SAR-MF
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Ray Oscillator with MA
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- Elder's SafeZone Stop
- ElderSafeZoneStopLong
- ElderSafeZoneStopShort
- Ergodic Oscillator
- Even Better Sinewave Indicator
- FastStochK FullStochK-D
- Fib CMO
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator

- Fisher Relative Vigour Index
- For Auto Trading Setup
- Forward/Reverse EMA by John Ehlers
- Fre
- Frequency distribution of returns
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future MA Projection
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann level plotter
- Gann Swing chart v41212
- Gann Swing Charts in 3 modes with text
- Geometric Mean of Volume
- Gfx Toolkit
- GFX ToolTip
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Guppy Cloud
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Heatmap V1
- Heikin Ashi Delta
- Heikin-Ashi(Koma-Ashi) with Moving Average
- High Low Detection code
- Hilbert Sine Wave
- Hilbert Sine Wave with Hull Moving Average
- Historical Volatility Index
- How to add IB Option Symbols
- Hull Moving Average
- Hurst "Like" DE
- Ichimoku Kinko Hyo
- ICHIMOKU SIGNAL TRADER
- IchimokuBrianViorelRO
- Indicator Explorer (ZigZag)
- Intraday Average Volume
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Intraday Trend Break System
- Intraday Volume EMA
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Jesse Livermore Secret Market Key
- John Ehler
- Kairi Relative Index
- Kiss and Touch with the Modified True Range
- Larry William's Volatility Channels
- Laugerre PPO Oscillator
- Least Squares Channel Indicator
- Linear Candle
- Linear Regression Line w/ Std Deviation Channels

- Log Time Scale
- LSMA
- Lunar Phases - original
- LunarPhase
- MACD BB Indicator
- MACD Histogram - Change in Direction
- MACD indicator display
- Market Breadth Chart-In-Chart
- Market Meanness Index
- Market Profile
- MCDX (Multi Color Dragon Extended)
- Meu Sistema de Trading - versão 1.0
- MFE and MAE and plot trades as indicator
- Mndahoo ADX
- Model Four
- Modified Head & Shoulder Pattern
- Modified-DVI
- Moving Averages NoX
- MS Darvas Box with Exploration
- Multi-color Volume At Price (VAP)
- MultiCycle 1.0
- Multiple Ribbon Demo
- Multiple sinus noised
- N Line Break Chart
- Nadaraya-Watson Envelope
- NASDAQ 100 Volume
- New HL Scanner
- Non-repaitning Zigzag line
- Noor_Doodie
- NRx Exploration
- nth (1 - 8) Order Polynomial Fit
- Open Range Breakout Trading System
- Optimal Weights
- Option Calls, Puts and days till third friday.
- Ord Volume
- ParabXO
- Parametric Chande Trendscore
- pattenz
- Pattern_-_Rectangle_Base_Breakout_on_High_Vol 2
- Percentage Price Oscillator
- Perceptron
- Peter Cooper
- Pivot Finder
- Pivot Point with S/R Trendlines
- Pivots And Prices
- Plot visual stop / target ratio.
- Point & figure Chart India Securities
- Position Sizer vers2, stocks and CFDs
- Position Sizing and Risk Price Graph - 2
- prakash
- Probability Density & Gaussian Distribution
- Projection Oscillator

- PVT Trend Decider
- Random Walk
- Random Walk Index
- Random Walk Index, base formula included
- Range Filter - Trading Strategy
- Ranking and sorting stocks
- Raw ADX
- Rea Time Daily Price Levels
- Relative Momentum Index (RMI)
- Relative strength comparison with moving average
- Relative Vigour Index
- Relative Volume
- Renko Chart
- Renko Chart
- Revised Renko chart
- RI - Auto Trading System
- RSI + Avgs
- RSI indicator with Upper & Lower Zone Bars
- RSI styleClipMinMax
- SAR-ForNextBarStop
- Scale Out: Futures
- Scan New High and New Low
- Schiff Lines
- shailu lunia
- Shares To Buy Price Graph
- Signal to Noise
- Simple Momentum
- Smoothed Adaptive Momentum
- Spearman Rank Correlation Coefficient
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Standard Error Bands (Native AFL)
- STARC Bands
- Stochastic %J - KDJ
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic Oscillator
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stop-loss Indicator bands
- Stops on percentages
- Stress with SuperSmoother
- SUPER PIVOT POINTS
- Super Trend Indicator
- Support and Resistance
- Support and resistance
- Support Resistance levels
- TAPE READING
- TD Moving Average 2
- TD REI
- TD sequential
- The Fibonaccian behavior

- The Three Day Reversal
- Three Day Balance Point
- tomy_frenchy
- Tracking Error
- Tracking Error
- Trades per second indicator
- Trailing Stop Loss
- Trend Detection
- Trend Lines from 2 points
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- Trix Bars number
- TTM Squeeze
- Twiggs Money Flow
- Twiggs money flow weekly
- Ultimate plus
- Updated Renko Chart
- ValueChart
- VAMA
- Vikram's Floor Pivot Intraday System
- Visi-Trade
- visual turtle trading system
- Visualization of stoploses and profit in chart
- Vivek Jain
- Volatility
- Volume Color with Dynamic Limit
- Volume Occilator
- Volume Spread for VSA
- Volume wieghted moving average
- Volume Zone Oscillator
- VolumeDivAvgV3_Study_BrS
- Voss Predictive Filter (A Peek Into the Future)
- VSTOP (2)
- VSTOP (3)
- White Theme
- William's Alligator System II
- WILSON RELATIVE PRICE CHANNEL
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- Woodies CCI
- Z-Score Indicator
- ZeroLag MACD(p,q,r)
- Zig Explorer
- Zig Zag
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag filter rewrited from scratch in AFL
- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

Comments:

More information:

See updated/extended version on-line.

ParamColor - add user user-definable color parameter

Exploration / Indicators
(AmiBroker 4.30)

SYNTAX	**ParamColor(*"name", defaultcolor*)**
RETURNS	NUMBER
FUNCTION	Adds a new user-definable parameter, which will be accessible via Parameters dialog : right click over chart pane and select "Parameters" or press Ctrl+R allows to change chart parameters - changes are reflected immediatelly. • "name" - defines parameter name that will be displayed in the parameters dialog • defaultcolor - defines default color value of the parameter colorCycle - accepted only by ParamColor function as default value, causes that default color cycles through orange, blue, green, turquoise, gold, violet, bright green, dark yellow IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).
EXAMPLE	`Plot( RSI(), "RSI", ParamColor( "RSI Color", colorRed ) );`
SEE ALSO	PARAM() function , PARAMSTR() function

References:

The **ParamColor** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- Adaptive Laguerre Filter, from John Ehlers
- Add SL/TGT other params to any strategy
- Advanced MA system
- Adverse Move Ratio
- ADXVMA
- AFL to Python COM Link
- ALJEHANI
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Andrews Pitchfork
- Andrews PitchforkV3.3
- AO+Momentum
- Arnaud Legoux Moving Average (ALMA)
- Aroon The Advisor
- AR_Prediction.afl
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trendlines using multiple timeframes
- Average Price Crossover
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Bman's HaDiffCO
- bolingerbands

- bonlinger bands
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- Chandelier Exit
- Chandelier Exit
- CoinToss ver 1
- Color MACD Histogram Changes
- Colorfull Price
- Computing Cointegration and ADF Dashboard
- Congestions detection
- Controlling Height of Volume Bars
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Darvas Amibroker
- Darvas Johndeo Research
- Dave Landry PullBack Scan
- DEBAJ
- DiNapoli Detrended Oscillator
- Dinapoli MACD (DEMA)
- DiNapolis 3x Displaced Moving Averages
- Double Super Trend System
- DT Oscillator
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elliott Wave Oscillator
- FastStochK FullStochK-D
- Fib CMO
- Fib Fan Based on ZZ
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann level plotter
- Gann Swing Charts in 3 modes with text
- GFX ToolTip
- Guppy Cloud
- Halftrend
- Heikin Ashi Delta
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- HH-LL-PriceBar
- Hilbert Sine Wave Support & Resistance
- Ichimoku Kinko Hyo
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Intraday Average Volume
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- IntraDay Open Marker

- Intraday Strength
- JEEVAN'S SRI CHAKRA
- Jesse Livermore Secret Market Key
- Kairi Relative Index
- Last Five Trades Result Dashboard – AFL code
- Linear Candle
- Linear Regression Line w/ Std Deviation Channels
- Lunar Phases - original
- LunarPhase
- MACD BB Indicator
- Market Breadth Chart-In-Chart
- Market Meanness Index
- MAVG
- MCDX (Multi Color Dragon Extended)
- Mndahoo ADX
- Modified-DVI
- MS Darvas Box with Exploration
- Multiple sinus noised
- Nadaraya-Watson Envelope
- Neural Network Powered Smooth/Predictive RSI V2
- nifty
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- Ord Volume
- ParabXO
- pattenz
- Perceptron
- Peter Cooper
- Pivot Point with S/R Trendlines
- Pivots And Prices
- plot tomorrows pivots on an intraday database
- Plot visual stop / target ratio.
- Point & figure Chart India Securities
- prakash
- Price Chart - Fundamental
- Prior Daily OHLC
- PVT Trend Decider
- Random Walk
- Random Walk Index
- Range Filter - Trading Strategy
- RI - Auto Trading System
- RSI + Avgs
- RSI indicator with Upper & Lower Zone Bars
- RSI styleClipMinMax
- SAR-ForNextBarStop
- Schiff Lines
- shailu lunia
- Simple Momentum
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Stochastic Oscillator
- Stops on percentages

- Super Trend Indicator
- Super Trend Indicator
- suresh
- TD Channel-1
- TD Channel-2
- TD Moving Average I
- Three Day Balance Point
- Trend Lines from 2 points
- TRENDAdvisor
- TrendingRibbonArrowsADX
- Trigonometric Fit - TrigFit with AR for cos / sin
- TSV
- Twiggs Money Flow
- Twiggs money flow weekly
- Ultimate plus
- VAMA
- Vikram's Floor Pivot Intraday System
- Visible Min and Max Value Demo
- visual turtle trading system
- Volume Color with Dynamic Limit
- Volume Occilator
- Volume Spread for VSA
- Volume wieghted moving average
- VSTOP (2)
- VSTOP (3)
- VWAP versus Average Price
- William's Alligator System II
- WILSON RELATIVE PRICE CHANNEL
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- Woodie's Price Panel With Woodie's Pivots
- Woodies CCI
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag filter rewrited from scratch in AFL
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

ParamDate
- add user user-definable date parameter

Exploration / Indicators
(AmiBroker 4.60)

SYNTAX **ParamDate("Name", "Default date", format = 0);**

RETURNS NUMBER or STRING

FUNCTION Adds a new user-definable date parameter, which will be accessible via Parameters dialog : right click over chart pane and select "Parameters" or press Ctrl+R allows to change chart parameters - changes are reflected immediatelly.

- "name" - defines parameter name that will be displayed in the parameters dialog
- "default date" - is a string holding date in any any format: YYYY-MM-DD, MM/DD/YY, DD-MM-YY, etc, etc.
- format - defines return value format, allowable values are:
  0 - return value is a NUMBER and holds DateNum. Ie: 990503 for May 3, 1999,
  1 - return value is a STRING formatted holding date according to your windows regional settings
  2 - return value is a NUMBER and holds DateTime. (new in 6.20)

WARNING: default parameter has to be CONSTANT. This is because these values are cached and are not re-read during subsequent formula evaluations.

IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).

EXAMPLE

```
start = ParamDate( "Start Date", "2003-05-03" );
```

SEE ALSO PARAM() function , PARAMCOLOR() function , PARAMSTR() function , ParamTime() function

References:

The **ParamDate** function is used in the following formulas in AFL on-line library:

- Continuous Contract Rollover
- High Low Detection code
- MFE and MAE and plot trades as indicator
- Plot the Equity Curve without Backtesting?

More information:

See updated/extended version on-line.

ParamField - creates price field parameter

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **ParamField("name", field = 3)**

RETURNS ARRAY

FUNCTION Allows to pick the **Price field** for the indicator (field which is used to calculate values of the indicator). Function returns the array defined by *field* parameter. Default value = 3 returns Close array. The possible values of *field* parameter are:

- **-1** - ParamField returns the values of the indicator that was inserted as a first one into the pane, or Close if no indicator was present
- **0** - returns **Open** array
- **1** - returns **High** array
- **2** - returns **Low** array
- **3** - returns **Close** array (default)
- **4** - returns **Average** array = (H+L+C)/3
- **5** - returns **Volume** array
- **6** - returns **Open Interest** array
- **7,8,9,....** - return values of indicators inserted into the pane.

IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32). Since ParamField uses names passed to Plot(), it means that Plot() calls must not contain non-printable characters too (ASCII codes < 32), so control characters such as new lines are NOT allowed.

EXAMPLE

SEE ALSO PARAM() function

References:

The **ParamField** function is used in the following formulas in AFL on-line library:

- Adaptive Laguerre Filter, from John Ehlers
- Advanced MA system
- AFL to Python COM Link
- AO+Momentum
- Arnaud Legoux Moving Average (ALMA)
- AR_Prediction.afl
- Average Price Crossover
- B-Xtrender
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Bman's HaDiffCO
- bolingerbands
- Bollinger Band Squeeze
- bonlinger bands
- Coral Trend Indicator
- Dave Landry PullBack Scan
- DEBAJ
- Elder Triple Screen Trading System

- Elder's Market Thermometer
- Fibonacci Moving averages
- Guppy Cloud
- Historical Volatility Index
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Kairi Relative Index
- Linear Regression Line w/ Std Deviation Channels
- Nadaraya-Watson Envelope
- Noor_Doodie
- prakash
- Range Filter - Trading Strategy
- regavg
- Square of Nine Roadmap Charts
- Stops on percentages
- tomy_frenchy
- Tracking Error
- Tracking Error
- Trigonometric Fit - TrigFit with AR for cos / sin
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

ParamList - creates the parameter that consist of the list of choices

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **ParamList("Name", "Values", defaultval = 0)**

RETURNS STRING

FUNCTION Creates the parameter that consist of the list of choices (specified in *"values"* parameter - | or comma separated). *defaultval* parameter defines ordinal position of the default string value specified in *"values"* parameter. Returned value is a STRING representing choosen item.

IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).

EXAMPLE

```
OrderType = ParamList("Order Type", "MKT|LMT|STP" );
```

SEE ALSO ParamDate() function , PARAMSTR() function , ParamTime() function , ParamTrigger() function , PARAMCOLOR() function , PARAM() function

References:

The **ParamList** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- Advanced Trend Lines with S & R
- Advisory NRx price chart display.
- AFL_Glossary_Converter
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- AR_Prediction.afl
- Auto Fib Ext&Retracement
- Auto Trade Step by Step
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Dollar Price Volatility Exploration
- Average Price Crossover
- babaloo chapora
- Backup Data of 1min Interval
- Bad Tick Trim on 5 sec database
- BBAreacolor&TGLCROSSNEW
- Bottom Fisher Exploration
- Button trading using AB auto trading interface
- candlestick chart for Volume/RSI/OBV
- Daily High Low in Advance
- Day Bar No
- DEBAJ
- Demand Index
- elliott wave manual labelling
- For Auto Trading Setup
- Gann Swing Charts in 3 modes with text

- Get Moneycontrol News Snippets into Amiboker
- Graphical sector stock amalysis
- Harmonic Patterns
- Heatmap V1
- Herman
- High Low Detection code
- How to add IB Option Symbols
- Hurst "Like" DE
- Intraday Fibonacii Trend Break System
- JEEVAN'S SRI CHAKRA
- Linear Candle
- MACD BB Indicator
- Market Breadth Chart-In-Chart
- Market Profile
- MFE and MAE and plot trades as indicator
- Multiple Ribbon Demo
- Murrey Math Price Lines
- New HL Scanner
- Open Range Breakout Trading System
- Periodically ReBalance a BUY & HOLD Portfolio
- Point & figure Chart India Securities
- Probability Density & Gaussian Distribution
- Robert Antony
- Scan New High and New Low
- Spearman Rank Correlation Coefficient
- Square of Nine Roadmap Charts
- SUPER PIVOT POINTS
- TD Channel-1
- TD Channel-2
- Three Day Balance Point
- Trend Lines from 2 points
- Trigonometric Fit - TrigFit with AR for cos / sin
- TSV
- Volume Zone Oscillator
- William's Alligator System II
- ZigZag Retracements

More information:

See updated/extended version on-line.

ParamStr
- add user user-definable string parameter

Exploration / Indicators
(AmiBroker 4.30)

SYNTAX	**ParamStr(*"name", "default"*)**
RETURNS	STRING
FUNCTION	Adds a new user-definable parameter, which will be accessible via Parameters dialog : right click over chart pane and select "Parameters" or press Ctrl+R allows to change chart parameters - changes are reflected immediatelly. • "name" - defines parameter name that will be displayed in the parameters dialog • "default" - defines default value of the parameter IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).
EXAMPLE	`ticker = ParamStr( "Ticker", "MSFT" );`
SEE ALSO	PARAM() function , PARAMCOLOR() function

References:

The **ParamStr** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically
- Advanced Search and Find
- AFL_Glossary_Converter
- AllinOneAlerts - Module
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- Create a list of functions in your program
- Extract specific lines of code from your program
- How to add IB Option Symbols
- MFE and MAE and plot trades as indicator
- Open Range Breakout Trading System
- Relative Strength Multichart of up to 10 tickers
- Rene Rijnaars
- Send Alerts from Amibroker to Telgram
- Stan Weinstein strategy
- Tracking Error
- Tracking Error

More information:

See updated/extended version on-line.

ParamStyle
- select styles applied to the plot

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **ParamStyle("name", defaultstyle = styleLine, mask = maskDefault)**

RETURNS NUMBER

FUNCTION Allows to select the styles applied to plot.

Parameters

- name - parameter name
- defaultstyle - default value of style , takes combination of style* constants
- mask - binary mask that defines which styles should be visible in the drop down list
  maskDefault - show thick, dashed, hidden, own scale styles (this is default mask for ParamStyle)
  maskAll - show all style flags
  maskPrice - show thick, hidden, own scale, candle, bar
  maskHistogram - show histogram, thick, hidden, own scale, area

  IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).

EXAMPLE

SEE ALSO

References:

The **ParamStyle** function is used in the following formulas in AFL on-line library:

- Adaptive Laguerre Filter, from John Ehlers
- Advanced MA system
- Adverse Move Ratio
- Advisory NRx price chart display.
- ADXVMA
- AFL to Python COM Link
- ALJEHANI
- Alphatrend
- AO+Momentum
- Arnaud Legoux Moving Average (ALMA)
- Aroon The Advisor
- AR_Prediction.afl
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Bman's HaDiffCO
- bolingerbands
- bonlinger bands
- Chandelier Exit
- Chandelier Exit

- changing period moving avarage
- Color MACD Histogram Changes
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- Commodity Selection Index (CSI)
- Controlling Height of Volume Bars
- Coppock Trade Signal on Price Chart
- Coral Trend Indicator
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Dave Landry PullBack Scan
- DEBAJ
- DiNapoli Detrended Oscillator
- Dinapoli MACD (DEMA)
- DiNapolis 3x Displaced Moving Averages
- Double Super Trend System
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elliott Wave Oscillator
- FastStochK FullStochK-D
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Gann level plotter
- Gann Swing Charts in 3 modes with text
- Guppy Cloud
- Halftrend
- Harmonic Patterns
- Heikin Ashi Delta
- HH-LL-PriceBar
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- interactively test discretionary trading
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Intraday Strength
- Intraday Trend Break System
- Kairi Relative Index
- Last Five Trades Result Dashboard – AFL code
- Linear Candle
- Linear Regression Line w/ Std Deviation Channels
- LunarPhase
- Market Meanness Index
- MAVG
- Meu Sistema de Trading - versão 1.0
- Mndahoo ADX
- MS Darvas Box with Exploration
- Multiple sinus noised

- Murrey Math Price Lines
- Nadaraya-Watson Envelope
- Neural Network Powered Smooth/Predictive RSI V2
- nifty
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- pattenz
- Perceptron
- Pivot End Of Day Trading System
- plot tomorrows pivots on an intraday database
- Plot visual stop / target ratio.
- prakash
- Price Chart - Fundamental
- Prior Daily OHLC
- PVT Trend Decider
- Random Walk
- Random Walk Index
- Range Filter - Trading Strategy
- Rebalancing Backtest avoiding leverage
- RI - Auto Trading System
- RSI + Avgs
- RSI indicator with Upper & Lower Zone Bars
- RSI styleClipMinMax
- SAR-ForNextBarStop
- Simple Momentum
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Stochastic Oscillator
- Stress with SuperSmoother
- Super Trend Indicator
- Super Trend Indicator
- suresh
- TD Channel-1
- TD Channel-2
- TD Moving Average 2
- TD Moving Average I
- TrendingRibbonArrowsADX
- Trigonometric Fit - TrigFit with AR for cos / sin
- Twiggs Money Flow
- Twiggs money flow weekly
- Ultimate plus
- Vikram's Floor Pivot Intraday System
- Visible Min and Max Value Demo
- visual turtle trading system
- Volume Color with Dynamic Limit
- Volume Occilator
- Volume wieghted moving average
- William's Alligator System II
- WILSON RELATIVE PRICE CHANNEL
- Zig-Hi Zap-Lo
- ZigZag filter rewrited from scratch in AFL
- ZigZag Retracements

More information:

See updated/extended version on-line.

ParamTime - add user user-definable time parameter

Exploration / Indicators
(AmiBroker 4.60)

SYNTAX **ParamTime("Name", "Default time", format = 0);**

RETURNS NUMBER or STRING

FUNCTION Adds a new user-definable time parameter, which will be accessible via Parameters dialog : right click over chart pane and select "Parameters" or press Ctrl+R allows to change chart parameters - changes are reflected immediatelly.

- "name" - defines parameter name that will be displayed in the parameters dialog
- "default time" - is a string holding time in any any format: HH:MM:SS, HH:MM, etc.
- format - defines return value format, allowable values are:
  0 - return value is a NUMBER and holds TimeNum. Ie: 133515 for 13:35:15
  1 - return value is a STRING formatted holding time according to your windows regional settings

WARNING: default parameter has to be CONSTANT. This is because these values are cached and are not re-read during subsequent formula evaluations.

IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).

EXAMPLE

```
start = ParamTime( "Start Time", "09:30" );
```

SEE ALSO PARAM() function , PARAMCOLOR() function , ParamDate() function , PARAMSTR() function

References:

The **ParamTime** function is used in the following formulas in AFL on-line library:

- MFE and MAE and plot trades as indicator
- Open Range Breakout Trading System

More information:

See updated/extended version on-line.

ParamToggle - create Yes/No parameter

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **ParamToggle("name","values",defaultval=0)**

RETURNS NUMBER

FUNCTION Function that allows to use boolean (Yes/No) parameters.

- "name" - the name of the parameter
- "values" - parameter values (separated with | character, e.g. "No|Yes" - first string represents false value and second string represents true value)
- defaultval - default value of the parameter

IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).

EXAMPLE

SEE ALSO PARAM() function , ParamTrigger() function

References:

The **ParamToggle** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Adaptive Centre of Gravity
- Adaptive Cyber Cycle
- Adaptive Relative Vigour Index
- Add SL/TGT other params to any strategy
- Advanced Search and Find
- Advanced Trend Lines with S & R
- AFL_Glossary_Converter
- AllinOneAlerts - Module
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Aroon The Advisor
- AR_Prediction.afl
- Auto Fib Ext&Retracement
- Auto Trade Step by Step
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- B-Xtrender
- Basket Trading System T101

- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bid Vs Ask Dashboard
- Brian Wild
- Button trading using AB auto trading interface
- Caleb Lawrence
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- Centre of Gravity
- channel indicator
- Channel/S&R and trendlines
- Congestions detection
- Controlling Height of Volume Bars
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Cyber Cycle
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period
- DEBAJ
- Evaluating Candle Patterns in a trading system
- Export EOD or Intraday to .csv file
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator
- Fisher Relative Vigour Index
- Fre
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann Swing Charts in 3 modes with text
- Geometric Mean of Volume
- Get Moneycontrol News Snippets into Amiboker
- Gfx Toolkit
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Heatmap V1
- Heikin Ashi Delta
- Hilbert Sine Wave Support & Resistance
- ICHIMOKU SIGNAL TRADER
- IchimokuBrianViorelRO
- interactively test discretionary trading
- Intraday Fibonacii Trend Break System
- Intraday Range and Periods Framer
- Intraday Volume EMA
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Least Squares Channel Indicator

- Linear Candle
- MACD BB Indicator
- Manual Bracket Order Trader
- Market Breadth Chart-In-Chart
- Multiple sinus noised
- N Line Break Chart
- NASDAQ 100 Volume
- NRx Exploration
- Peter Cooper
- Pivots And Prices
- Point & figure Chart India Securities
- prakash
- Random Walk
- Range Filter - Trading Strategy
- Relative Vigour Index
- Renko Chart
- RSI + Avgs
- RSI indicator with Upper & Lower Zone Bars
- Say Notes
- Smoothed Adaptive Momentum
- Square of Nine Roadmap Charts
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stress with SuperSmoother
- Support and resistance
- TAPE READING
- TD Sequential
- Three Day Balance Point
- Tracking Error
- Trend Lines from 2 points
- Triangle Search Extended
- Trigonometric Fit - TrigFit with AR for cos / sin
- TSV
- Updated Renko Chart
- Vikram's Floor Pivot Intraday System
- Visi-Trade
- Visualization of stoploses and profit in chart
- Volume Divided Histogram
- Volume Spread for VSA
- VolumeDivAvgV3_Study_BrS
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount
- ZigZag Retracements

More information:

See updated/extended version on-line.

ParamTrigger - creates a trigger (button) in the parameter dialog

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **ParamTrigger("Name", "Button text")**

RETURNS NUMBER

FUNCTION Creates trigger (button) in the Parameter dialog.

If you place ParamTrigger in the indicator code it will create a "button" in Parameter dialog that can be pressed. Normally ParamTrigger will return zero (0) but when button in the param window is pressed then it will refresh the chart and ParamTrigger will return 1 (one) for this single execution (further refreshes will return zero, until the button is pressed again)

IMPORTANT: Parameter names and values must NOT contain non-printable characters (ASCII codes < 32).

EXAMPLE

```
trigger = ParamTrigger("Place Order", "Click here to place order");

if( trigger )
{
// your one-shot code here
}
```

SEE ALSO

References:

The **ParamTrigger** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically
- AFL Timing functions
- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- channel indicator
- Chart Zoom
- Continuous Contract Rollover
- Create a list of functions in your program
- elliott wave manual labelling
- Extract specific lines of code from your program
- For Auto Trading Setup
- Heatmap V1
- How to add IB Option Symbols
- interactively test discretionary trading
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Reconnect TWS
- Send Alerts from Amibroker to Telgram
- TAPE READING

More information:

See updated/extended version on-line.

PDI
- plus directional movement indicator

Indicators
(AmiBroker 3.30)

SYNTAX	**pdi(period = 14)**
RETURNS	ARRAY
FUNCTION	Calculates Plus Directional Movement Indicator (+DI line)
EXAMPLE	decvolume()
SEE ALSO	

References:

The **PDI** function is used in the following formulas in AFL on-line library:

- ADX Indicator - Colored
- ADXbuy
- AJDX system
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- CCI/DI+- COMBO indicator
- Commodity Selection Index (CSI)
- Dave Landry Pullbacks
- DMI Spread Index
- ekeko price chart
- Heatmap V1
- ICHIMOKU SIGNAL TRADER
- Index and ETF trading
- Mndahoo ADX
- Multiple Ribbon Demo
- swing chart
- The Three Day Reversal
- Trend Analysis_Comentary
- Trending Ribbon
- TrendingRibbonArrowsADX
- Vivek Jain

More information:

See updated/extended version on-line.

Peak - peak

Basic price pattern detection
(AmiBroker 3.10)

SYNTAX	**Peak(ARRAY, *change*, *n* = 1)**
RETURNS	ARRAY
FUNCTION	Gives the value of ARRAY *n*-th peak(s) ago. This uses the Zig Zag function (see Zig Zag) to determine the peaks. *n* =1 would return the value of the most recent peak. *n* =2 would return the value of the 2nd most recent peak. **Caveat:** this function is based on Zig-Zag indicator and may look into the future.
EXAMPLE	peak(close,5,1)
SEE ALSO	

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2003-02-13 04:02:04	Zig/Peak/Trough functions work correctly for ARRAYS containing data greater than zero.

References:

The **Peak** function is used in the following formulas in AFL on-line library:

- Advanced Trend Lines with S & R
- Andrews Pitchfork
- Andrews PitchforkV3.3
- Double top detection
- ekeko price chart
- Fund Screener
- Gartley 222 Pattern Indicator
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- MACD commentary
- Pivot Point with S/R Trendlines
- Schiff Lines
- Support Resistance levels
- Tom DeMark Trend Lines
- Wolfe Wave Patterns
- Woodie's CCI Panel Full Stats
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Retracements

More information:

See updated/extended version on-line.

PeakBars
- bars since peak

Basic price pattern detection
(AmiBroker 3.10)

SYNTAX	**PeakBars(ARRAY, *change*, *n* = 1)**
RETURNS	ARRAY
FUNCTION	Gives the number of bars that have passed from the *n*-th peak. This uses the Zig Zag function (see Zig Zag) to determine the peaks. *n* =1 would return the number of bars that have passed since the most recent peak. *n* =2 would return the number of bars that have passed since the 2nd most recent peak **Caveat:** this function is based on Zig-Zag indicator and may look into the future.
EXAMPLE	peakbars(close,5,1)
SEE ALSO	

References:

The **PeakBars** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Advanced Trend Lines with S & R
- DMI Spread Index
- Fibonacci Internal and External Retracements
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gartley 222 Pattern Indicator
- Harmonic Patterns
- Head & Shoulders Pattern
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- LunarPhase
- Modified Head & Shoulder Pattern
- Pattern Recognition Exploration
- Pivot Point with S/R Trendlines
- QP2 Float Analysis
- RSI Trendlines and Wedges
- Stochastics Trendlines
- The Fibonaccian behavior
- Tom DeMark Trend Lines
- Wolfe Wave Patterns
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount
- ZigZag Retracements

More information:

See updated/extended version on-line.

Percentile - calculate percentile

Statistical functions
(AmiBroker 4.50)

SYNTAX **Percentile(** ***array, period, rank*** **)**

RETURNS ARRAY

FUNCTION The **Percentile** function gives *rank* percentile value of the array over last *period* bars.

rank is 0..100 - defines percentile rank in the array

Performance note: the implementation of percentile function involves sorting that is relatively slow process even though that quicksort algorithm is used.

Since version 5.92 Percentile supports variable period.

Note that Percentile is very computation intensive function (it involves re-sorting arrays every bar) and variable-period version (if you call it with period being ARRAY) runs slower than scalar version

EXAMPLE

```
// Example 1:
// show bars when 'current' Day Volume ranks within
// TOP 30% of volumes of last 100 bars (is above 70th Percentile)
Filter = Volume > Percentile( Volume, 100, 70 );

// Example 2:
// show bars when 'current' Day Volume ranks within
// BOTTOM 30% of volumes of last 100 bars (is below 30th percentile)
Filter = Volume < Percentile( Volume, 100, 30 );

// variable period version
bi = BarIndex();
x = Percentile( Close, bi, 50 );
Plot( x, "Cumulative 50% Percentile", colorRed );
Plot( Close, "Price", colorDefault, styleCandle );
```

SEE ALSO Median() function

References:

The **Percentile** function is used in the following formulas in AFL on-line library:

- Market Meanness Index

More information:

See updated/extended version on-line.

PercentRank - calculate percent rank

Indicators
(AmiBroker 5.40)

SYNTAX **PercentRank(array, range)**

RETURNS ARRAY

FUNCTION INPUTS:

- array - input data
- range - lookback range

Returns percent rank (0...100) of the current element of the array within all elements over the specified range.

A value of 100 indicates that the current element of the array is the highest for the given lookback range, while a value of 0 indicates that the current value is the lowest for the given lookback range.

It is equivalent (but 2x faster) to:

```
function PercentRank2( Data, Periods)
{
   Count = 0;
  for ( i = 1; i <= Periods ; i++ )
   {
   Count += Data > Ref( Data, -i );
   }
  return 100 * Count / Periods;
}
```

EXAMPLE

SEE ALSO

References:

The **PercentRank** function is used in the following formulas in AFL on-line library:

- Bollinger Band Squeeze
- ConnorsRSI

More information:

See updated/extended version on-line.

PlaySound - play back specified .WAV file

Miscellaneous functions
(AmiBroker 5.40)

SYNTAX	**PlaySound("filename")**
RETURNS	NUMBER
FUNCTION	The function plays back specified .WAV file. Returns 1 on success, 0 on failure
EXAMPLE	PlaySound("c:\\windows\\media\\ding.wav");
SEE ALSO	

References:

The **PlaySound** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Plot
- plot indicator graph

Exploration / Indicators
(AmiBroker 3.80)

SYNTAX **Plot(** ***array*****,** ***name*****,** ***color/barcolor*****,** ***style*** **= styleLine,** ***minvalue*** **= {empty},** ***maxvalue*** **= {empty},** ***XShift*** **= 0,** ***Zorder*** **= 0,** ***width*** **= 1)**

RETURNS NUMBER

FUNCTION Plots the graph using ***array*** data. Parameters:

- ***array*** - data array to be plotted
- ***name*** - defines graph name used for displaying values in a title bar.
- ***color*** - defines plot color that could be static (if third argument is a number) or dynamic (when third argument is an array). Color indexes are related to the current palette (see Preferences/Color)
- ***style*** is a combination of one or more of following values:

  styleLine = 1 - normal (line) chart (default)
  styleHistogram = 2 - histogram chart
  styleThick =4 - fat (thick)
  styleDots = 8 - include dots
  styleNoLine = 16 - no line
  styleDashed = 32 - dashed line style
  styleCandle = 64 - candlestick chart
  styleBar = 128 - traditional bar chart
  styleNoDraw = 256 - no draw (perform axis scaling only)
  styleStaircase = 512 - staircase (square) chart
  styleSwingDots = 1024 - middle dots for staircase chart
  styleNoRescale = 2048 - no rescale
  styleNoLabel = 4096 - no value label
  stylePointAndFigure = 8192 - point and figure
  (new in 4.20):
  styleArea = 16384 - area chart (extra wide histogram)
  styleOwnScale = 32768 - plot is using independent scaling
  styleLeftAxisScale = 65536 - plot is using left axis scale (independent from right axis)
  styleNoTitle - do not display values of this plot in the chart title
  styleCloud - cloud style (area between high and low arrays) - to be used with PlotOHLC function
  styleClipMinMax - clip (do not paint) area between min and max levels - note this style is incompatible with printers and WMF (metafiles).
  styleGradient - (new in 5.60) - gradient area chart. Upper gradient color is specified by color parameter in Plot() function, bottom gradient color is either background color or can be defined using SetGradientFill function. styleGradient can be combined with styleLine
- ***minvalue*** and ***maxvalue -*** (used by styleOwnScale plots ONLY) define plot minimum and maximum values (lower and upper boundary for Y axis)
- ***XShift*** - allows to visually shift the chart past the last bar.
- ***ZOrder*** - defines the Z-axis position of given plot. The default is zero. Zorder = 0 means also where the "grid" is located. So if you want to plot BEHIND the grid you need to specify negative zorder parameter.Plots are drawn in the following order:

- ♦ zorder parameter takes precedence over the order of calling Plot() functions, so if z-order is set, it determines plotting order. See http://www.amibroker.com/gifs/zorder.gif
- ♦ If multiple plots use the same z-order parameter they are plotted in reverse call order (ones that appear last in the code are plotted first). This rule can be changed by already existing switch graphzorder = 1 which, when specified, reverses this behaviour (so plots are drawn in call order). Please note the above applies to each zorder "layer" separately (so within same zorder "layer" reverse call rule applies) This may sound complicated but is required for backward compatibility.

- ***width*** - defines pixel or percent width of given plot. The default is 1 pixel. Positive values specify PIXEL width, negative values specify width in percent of current bar width. So for example -20 will give you dynamic width that is 20% of bar width. Example: Plot(C, "Close", colorBlack, styleBar, Null, Null, 0, 1, -20 /* line width as percent of bar */);

EXAMPLE

```
// Example 20-bar Moving average shifted 10 bars into the future
past the last bar:
Plot(Close,"Close",colorBlack,styleCandle);
Plot(MA(Close,20), "Shifted MA", colorRed, styleLine, Null, Null, 10
);
// Note that shift occurs during plotting AND does NOT affect source
array
```

SEE ALSO PLOTFOREIGN() function , PLOTGRID() function , PlotText() function , PLOTSHAPES() function , PLOTOHLC() function

References:

The **Plot** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- % B of Bollinger Bands With Adaptive Zones
- %b indicator - related bollinger bands
- 'R' Channel
- 10-20 Indicator
- 2 Timeframes Candlestick Bar Chart
- 3 Price Break
- 3 ways to use RMI in one script
- 30 Week Hi Indicator - Display
- 3TF Candlestick Bar Chart
- 52 Week New High-New Low Index
- Abhimanyu
- AccuTrack
- Adaptave Zones O/B & O/S Oscillator
- Adaptive Centre of Gravity
- Adaptive Cyber Cycle
- Adaptive Laguerre Filter, from John Ehlers
- Adaptive Price Channel
- Adaptive Relative Vigour Index
- Add SL/TGT other params to any strategy
- Advanced MA system
- Advanced Trend Lines with S & R

- Adverse Move Ratio
- Advisory NRx price chart display.
- ADX Indicator - Colored
- ADXR
- ADXVMA
- AFL Example
- AFL Example - Enhanced
- AFL Timing functions
- AFL to Python COM Link
- ALJEHANI
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Andrews Pitchfork
- Andrews PitchforkV3.3
- Another FIb Level
- AO+Momentum
- Application of Ehler filter
- Arnaud Legoux Moving Average (ALMA)
- Aroon The Advisor
- AR_Prediction.afl
- Auto Fib Ext&Retracement
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trend-line
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- Awsome Oscilator
- B-Xtrender
- babaloo chapora
- Bad Tick Trim on 5 sec database
- Balance of Power
- balance of power
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Better Bollinger Bands
- Bman's HaDiffCO
- BMTRIX Intermediate Term Market Trend Indicator
- bolingerbands
- Bollinger - Keltner Bands
- Bollinger Band Squeeze
- Bollinger Band Width
- Bollinger Fibonacci Bands
- bonlinger bands
- Brian Wild
- Bull/Bear Volume
- Bullish Percent Index 2 files combined

- Bullish Percent Index 2004
- Button trading using AB auto trading interface
- Buyer Seller Force
- Caleb Lawrence
- Candle Identification
- Candle Stick Analysis
- Candle Stick Demo
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- CCI(20) Divergence Indicator
- Centre of Gravity
- Chande Momentum Oscillator
- Chande's Trend Score
- Chandelier Exit
- Chandelier Exit
- Chandelier Exit or Advanced Trailing Stop
- Chandelier Plugin
- changing period moving avarage
- channel indicator
- Channel/S&R and trendlines
- CoinToss ver 1
- Color Display.afl
- Color MACD Histogram Changes
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- com-out
- Commodity Selection Index (CSI)
- Composite Index
- Computing Cointegration and ADF Dashboard
- Congestions detection
- Connors TPS
- ConnorsRSI
- Constant Trendline Plot
- Continuous Contract Rollover
- Controlling Height of Volume Bars
- Coppock Histogram
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Coral Trend Indicator
- correlerror
- Cyber Cycle
- Cybernertic Hilbert Sine Wave
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period
- Dahl Oscillator modified
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Darvas Wisestocktrader

- Dave Landry PullBack Scan
- Day Bar No
- Days to Third Friday
- DEBAJ
- Demand Index
- Digital indiactors
- DiNapoli Detrended Oscillator
- Dinapoli MACD (DEMA)
- Dinapoli Perferred Stochastic
- DiNapolis 3x Displaced Moving Averages
- Distance Coefficient Ehlers Filter
- Divergence indicator
- Dominant Cycle Phase
- Donchian Channel
- Double Super Trend System
- DT Oscillator
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- Effective Swing Indicator
- Ehlers Center of Gravity Oscillator
- Ehlers CyberCycle
- Ehlers Dominant Cycle Period
- Ehlers Fisher Transform
- Ehlers Hilbert Transformer Indicator
- Ehlers Instantaneous Trend
- Ehlers Laguerre RSI
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- ekeko price chart
- EKEKO SAR-MF
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Ray - Bull Bear
- Elder Ray Oscillator with MA
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- Elder's SafeZone Stop
- Elliott Wave Oscillator
- Ema bands
- EMA Crossover
- EMA Crossover Price
- Even Better Sinewave Indicator
- Expiry day/days - Last thursday of month
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fib CMO
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements

- Fibonacci Moving averages
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator
- Fisher Relative Vigour Index
- fishnet
- For Auto Trading Setup
- Force index
- Forward/Reverse EMA by John Ehlers
- Fre
- Frequency distribution of returns
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future MA Projection
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Futures - Dollar Move Indicator
- Futures - Dollar Move Today Indicator
- Gabriel Linear Regression Angle Indicator
- Gann level plotter
- Gann Swing Charts in 3 modes with text
- Gartley 222 Pattern Indicator
- Geometric Mean of Volume
- Get Moneycontrol News Snippets into Amiboker
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Guppy Cloud
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Heikin Ashi Delta
- Herman
- HH-LL-PriceBar
- High Low Detection code
- Hilbert Sine Wave
- Hilbert Sine Wave with Hull Moving Average
- Historical Volatility Index
- Historical Volotility Scan - 6/100
- Historical Volotility Scan - 50 Day
- HLspread
- How to add IB Option Symbols
- Hull Moving Average
- Hull Multiple Moving Averages
- Hull Range Indicator
- Hull Rate of Return Indicator
- Hurst "Like" DE
- IB Backfiller
- Ichimoku Kinko Hyo
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Ichimoku with plot mofified to use cloud function
- IchimokuBrianViorelRO

- IFT of RSI - Multiple TimeFrames
- Improved NH-NH scan / indicator
- Index of 30 Wk Highs Vs Lows
- Indicator Explorer (ZigZag)
- Instantaneous Trend Line
- Inter-market Yield Linear Regression Divergence
- interactively test discretionary trading
- Intraday Average Volume
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Intraday Range and Periods Framer
- Intraday Strength
- Intraday Trend Break System
- Intraday Volume EMA
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Jesse Livermore Secret Market Key
- John Ehler
- Kairi Relative Index
- Kiss and Touch with the Modified True Range
- Lagging MA-Xover
- Last Five Trades Result Dashboard – AFL code
- Laugerre PPO Oscillator
- Least Squares Channel Indicator
- Linear Candle
- Linear Regression Line w/ Std Deviation Channels
- Log Time Scale
- LSMA
- Luna Phase
- LunarPhase
- MACD BB Indicator
- MACD Histogram - Change in Direction
- MACD indicator display
- MACD-V
- MAM
- Market Meanness Index
- MAVG
- MCDX (Multi Color Dragon Extended)
- Meu Sistema de Trading - versão 1.0
- mfimacd
- mitalpradip
- Mndahoo ADX
- MOCS
- Modified Head & Shoulder Pattern
- Modified-DVI
- Momentum Volume Price (MVP) Indicator
- Moving Averages NoX
- Moving Trend Bands (MTB)
- MO_CrashZone
- MS Darvas Box with Exploration
- Multi-color Volume At Price (VAP)
- MultiCycle 1.0

- Multiple Ribbon Demo
- Multiple sinus noised
- Murrey Math Price Lines
- N Line Break Chart
- Nadaraya-Watson Envelope
- NASDAQ 100 Volume
- Neural Network Powered Smooth/Predictive RSI V2
- New HL Scanner
- nifty
- nikhil
- Non-repaitning Zigzag line
- Nonlinear Ehlers Filter
- Noor_Doodie
- nth (1 - 8) Order Polynomial Fit
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- Ord Volume
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- Parabolic SAR in native AFL (v.4.31.1 required)
- ParabXO
- Parametric Chande Trendscore
- pattenz
- Percentage Price Oscillator
- Perceptron
- Peter Cooper
- PF Chart - Close - April 2004
- Pivot Calculator
- Pivot End Of Day Trading System
- Pivot Point with S/R Trendlines
- Pivots And Prices
- Pivots for Intraday Forex Charts
- Plot the Equity Curve without Backtesting?
- plot tomorrows pivots on an intraday database
- Plot visual stop / target ratio.
- Point & figure Chart India Securities
- Polarized Fractal Efficiency
- Polyfit Lines
- Position Sizer vers2, stocks and CFDs
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- Positive Bars Percentage
- prakash
- Price Chart - Fundamental
- Price with Woodies Pivots
- Prior Daily OHLC
- Projection Oscillator
- PVT Trend Decider
- R-Squared
- Random Walk
- Random Walk Index
- Random Walk Index, base formula included

- Range Filter - Trading Strategy
- Ranking and sorting stocks
- Ranking Ticker WatchList
- Rea Time Daily Price Levels
- Rebalancing Backtest avoiding leverage
- regavg
- Relative Momentum Index (RMI)
- Relative Strength Multichart of up to 10 tickers
- Relative Vigour Index
- Relative Volume
- Rene Rijnaars
- Renko Chart
- Renko Chart
- Reverse EMA function
- Revised Renko chart
- RI - Auto Trading System
- Robert Antony
- RSI + Avgs
- RSI indicator with Upper & Lower Zone Bars
- RSI styleClipMinMax
- RSI Trendlines and Wedges
- RSIS
- RUTVOL timing signal with BB Scoring routine
- SAR-ForNextBarStop
- Schiff Lines
- Second-order Infinite impulse response filter
- Shares To Buy Price Graph
- Signal to Noise
- Simple Momentum
- Sine Wave Indicator
- Smoothed Adaptive Momentum
- Sony
- Spearman Rank Correlation Coefficient
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Standard Error Bands (Native AFL)
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Stochastic %J - KDJ
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic of Weekly Price Array
- Stochastic Oscillator
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stops on percentages
- Stress with SuperSmoother
- Super Trend Indicator
- Super Trend Indicator
- Support and resistance
- Support Resistance levels
- suresh

- T3 Function
- TD Channel-1
- TD Channel-2
- TD Moving Average 2
- TD Moving Average I
- TD sequential
- TD Sequential
- The D_oscillator
- The Fibonaccian behavior
- The Saturation Indicator D_sat
- The Stochastic CCI
- Three Day Balance Point
- Three Pole Butterworth
- Time Frame Weekly Bars
- Time segment value
- Tom DeMark Trend Lines
- tomy_frenchy
- Tracking Error
- Tracking Error
- Trades per second indicator
- Trading ATR 10-1
- Trailing Stop Loss
- Trend Detection
- Trend Following System
- Trend Lines from 2 points
- Trend Trigger Factor
- TRENDAdvisor
- Trending Ribbon
- TrendingRibbonArrowsADX
- Triangular Moving Average
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- Trix Bars number
- TSV
- TTM Squeeze
- Twiggs Money Flow
- Twiggs money flow weekly
- TWS auto-export Executions-file parser
- Ultimate plus
- Updated Renko Chart
- Using From and To dates from Auto Analysis in Code
- ValueChart
- VAMA
- Vertical Horizontal Filter (VHF)
- Vikram's Floor Pivot Intraday System
- Visi-Trade
- Visible Min and Max Value Demo
- visual turtle trading system
- Visualization of stoploses and profit in chart
- Volatility
- Volatility System
- Volume - compared with Moving Avg (100%)

- Volume Charts
- Volume Color with Dynamic Limit
- Volume Divided Histogram
- Volume Occilator
- Volume Spread for VSA
- Volume Zone Oscillator
- Voss Predictive Filter (A Peek Into the Future)
- VSTOP (2)
- VSTOP (3)
- VWAP versus Average Price
- White Theme
- William's Alligator System II
- WILSON RELATIVE PRICE CHANNEL
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- Woodies CCI
- z-distance from vwap
- Z-Score Indicator
- ZeroLag MACD(p,q,r)
- Zig Zag
- Zig Zag Indicator with Valid Entry and Exit Points
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag filter rewrited from scratch in AFL
- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

PlotForeign - plot foreign security data

Referencing other symbol data
(AmiBroker 4.20)

SYNTAX **PlotForeign(*tickersymbol*, *name*, *color/barcolor*, *style* = *styleCandle* | *styleOwnScale*, *minvalue* = {empty}, *maxvalue* = {empty}, XShift = 0, ZOrder = 0, width = 1)**

RETURNS NUMBER

FUNCTION Plots the foreign-symbol price chart (symbol is defined by *tickersymbol* parameter). Second argument *name* defines graph name used for displaying values in a title bar. Graph color could be static (if third argument is a number) or dynamic (when third argument is an array). Color indexes are related to the current palette (see Preferences/Color)
style defines chart plot style (see Plot() function for possible values)
minvalue and *maxvalue* - (used by styleOwnScale plots ONLY) define plot minimum and maximum values (lower and upper boundary for Y axis)
XShift - allows to visually shift the chart into future (blank) bars.
ZOrder - this parameter takes precedence over the order of calling Plot() functions, so if z-order is set, it determines plotting order. See http://www.amibroker.com/gifs/zorder.gif
If multiple plots use the same z-order parameter they are plotted in reverse call order (ones that appear last in the code are plotted first). This rule can be changed by already existing switch graphzorder = 1 which, when specified, reverses this behaviour (so plots are drawn in call order).

Please note the above applies to each zorder "layer" separately (so within same zorder "layer" reverse call rule applies). This may sound complicated but is required for backward compatibility.

width - defines pixel or percent width of given plot. The default is 1 pixel. Positive values specify PIXEL width, negative values specify width in percent of current bar width. So for example -20 will give you dynamic width that is 20% of bar width. Example: Plot(C, "Close", colorBlack, styleBar, Null, Null, 0, 1, -20 /* line width as percent of bar */);

EXAMPLE PlotForeign("^DJI", "Dow Jones", colorRed);

SEE ALSO

References:

The **PlotForeign** function is used in the following formulas in AFL on-line library:

- Dave Landry PullBack Scan
- Elder Triple Screen Trading System
- Intraday Fibonacii Trend Break System
- Peter Cooper
- qp2 industry charts as a panel in the stocks chart

More information:

See updated/extended version on-line.

PlotGrid - Plot horizontal grid line

Exploration / Indicators
(AmiBroker 4.30)

SYNTAX	**PlotGrid(level, color = colorDefault, pattern = 1, width = 1, Label = True)**
RETURNS	NOTHING

FUNCTION Plots hotizontal grid line using built-in dotted style at given level, color, pattern, width, with or without Label.

New in version 5.90:

- *pattern* - defines line pattern. Available grid patterns 1..10 as shown in the Tools->Preferences, "Axes & Grids", 1-7 are single pixel patterns, 8 is regular windows dot pattern (PS_DOT) , 9 is regular Windows dash pattern (PS_DASH), 10 is solid line
- *width* - grid line width (in pixels). Note that due to Windows GDI limitations only patterns 8 (PS_DOT), 9 (PS_DASH), 10 (PS_SOLID) are available in widths > 1
- *label* - whenever to display value label or not

Use PlotGrid to display horizontal lines that are constant instead of using Plot. PlotGrid offers much better performance in this case.

So instead of

Plot(50, "", colorRed, styleLine | styleThick);

use:

PlotGrid(50, colorRed, 10, 2, False); // solid line 2 pixels thick, no label

EXAMPLE

```
PlotGrid( 25, colorRed );
```

SEE ALSO PLOT() function , PLOTFOREIGN() function , PLOTOHLC() function

Comments:

Tomasz Janeczko
tj --at-- amibroker.com
2003-04-18 07:12:14
	Instead of number you can also use expression but it must be NUMERIC expression, not ARRAY.
	
	Use LastValue to convert:
	
	your_expression = ...
	PlotGrid(LastValue(your_expression));

References:

The **PlotGrid** function is used in the following formulas in AFL on-line library:

- Adverse Move Ratio
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style

- Composite Index
- Dynamic Momentum Index
- Dynamic Momentum Index
- Ehlers Center of Gravity Oscillator
- Ehlers Fisher Transform
- Ehlers Laguerre RSI
- IFT of RSI - Multiple TimeFrames
- Kairi Relative Index
- MultiCycle 1.0
- P&F chart with range box sizes
- Rene Rijnaars
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- z-distance from vwap

More information:

See updated/extended version on-line.

PlotOHLC - plot custom OHLC chart

Exploration / Indicators
(AmiBroker 4.20)

SYNTAX **PlotOHLC(*open, high, low, close*, *name*, *color/barcolor*, *style* = *styleCandle* | *styleOwnScale*, *minvalue* = {empty}, *maxvalue* = {empty}, *XShift* = 0, ZOrder = 0, width = 1)**

RETURNS NUMBER

FUNCTION Plots the price chart using custom *open, high, low, close* arrays supplied as parameters. Fifth argument *name* defines graph name used for displaying values in a title bar. Graph color could be static (if sixth argument is a number) or dynamic (when sixth argument is an array). Color indexes are related to the current palette (see Preferences/Color)
style defines chart plot style (see Plot() function for possible values)
minvalue and *maxvalue* - (used by styleOwnScale plots ONLY) define plot minimum and maximum values (lower and upper boundary for Y axis)

XShift - allows to visually shift the chart into future (blank) bars.

ZOrder - this parameter takes precedence over the order of calling Plot() functions, so if z-order is set, it determines plotting order. See http://www.amibroker.com/gifs/zorder.gif
If multiple plots use the same z-order parameter they are plotted in reverse call order (ones that appear last in the code are plotted first). This rule can be changed by already existing switch graphzorder = 1 which, when specified, reverses this behaviour (so plots are drawn in call order).

Please note the above applies to each zorder "layer" separately (so within same zorder "layer" reverse call rule applies) This may sound complicated but is required for backward compatibility.

width - defines pixel or percent width of given plot. The default is 1 pixel. Positive values specify PIXEL width, negative values specify width in percent of current bar width. So for example -20 will give you dynamic width that is 20% of bar width. Example: Plot(C, "Close", colorBlack, styleBar, Null, Null, 0, 1, -20 /* line width as percent of bar */);

EXAMPLE PlotOHLC(1.1*Open, 1.1* High, 1.1* Low, 1.1* Close, "Price chart shifted 10% up", colorRed, styleCandle);

SEE ALSO PLOT() function , PLOTFOREIGN() function

References:

The **PlotOHLC** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- Alphatrend
- Aroon The Advisor
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- automatic trendlines using fractal patterns
- babaloo chapora
- BBAreacolor&TGLCROSSNEW

- Bollinger Fibonacci Bands
- Brian Wild
- candlestick chart for Volume/RSI/OBV
- channel indicator
- com-out
- Congestions detection
- Constant Trendline Plot
- Cycle Period
- Digital indiactors
- DPO with shading
- Elder Triple Screen Trading System
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Gordon Rose
- Guppy Cloud
- Heatmap V1
- Heikin Ashi Candles
- Heikin Ashi System
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- Hurst "Like" DE
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- IchimokuBrianViorelRO
- Indicator Explorer (ZigZag)
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Intraday Range and Periods Framer
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- Least Squares Channel Indicator
- Linear Candle
- Log Time Scale
- Lunar Phases - original
- MACD BB Indicator
- MACD Histogram - Change in Direction
- Meu Sistema de Trading - versão 1.0
- mfimacd
- Monthly bar chart
- MO_CrashZone
- N-period candlesticks (time compression)
- NASDAQ 100 Volume
- Nick
- nth (1 - 8) Order Polynomial Fit
- Pivot Finder
- Pivots for Intraday Forex Charts
- Plot visual stop / target ratio.
- Point & figure Chart India Securities
- Prashanth
- Price with Woodies Pivots
- Rea Time Daily Price Levels
- Renko Chart
- Revised Renko chart

- RSI + Avgs
- RSI indicator with Upper & Lower Zone Bars
- RSI styleClipMinMax
- shailu lunia
- Stress with SuperSmoother
- Support and resistance
- Three Line Break - TLB
- TSV
- ValueChart
- Vikram's Floor Pivot Intraday System
- Volume Occilator
- Volume Spread for VSA
- Weekly chart
- WILSON RELATIVE PRICE CHANNEL
- Woodie's Heikin-Ashi Panel

More information:

See updated/extended version on-line.

PlotShapes - plots arrows and other shapes

Exploration / Indicators
(AmiBroker 4.30)

SYNTAX **PlotShapes(*shape, color, layer = 0, yposition = graph0, offset = -12, XShift = 0*);**

RETURNS NOTHING

FUNCTION Plots arrows and other shapes on any chart pane.

Parameters:

- shape defines type of the symbol. when shape is zero nothing is plotted values other than zero cause plotting various pre-defined shapes. Odd values plot shape BELOW indicator, even values plot shape ABOVE indicator.
- color defines color of shape
- layer defines layer number on which shapes are plotted
- yposition defines Y-position where shapes are plotted (by default they are plotted 'around' graph0 (first indicator) line)
- offset - (or distance) parameter (by default -12), Offset is expressed in SCREEN pixels. Negative offsets shift symbols down, positive offsets shift symbol up. To place the shape exactly at ypostion, specify 0 as offset
- (new in 5.66) XShift - allows to visually shift the the shapes by the specified number of bars (even past the last bar).

Constants for shapes:

shapeNone, shapeUpArrow, shapeDownArrow, shapeHollowUpArrow, shapeHollowDownArrow, shapeSmallUpTriangle, shapeSmallDownTriangle, shapeHollowSmallUpTriangle, shapeHollowSmallDownTriangle, shapeUpTriangle, shapeDownTriangle, shapeHollowUpTriangle, shapeHollowDownTriangle, shapeSmallSquare, shapeHollowSmallSquare, shapeSquare, shapeHollowSquare, shapeSmallCircle, shapeHollowSmallCircle, shapeCircle, shapeHollowCircle, shapeStar, shapeHollowStar, shapeDigit0, shapeDigit1, shapeDigit2, shapeDigit3, shapeDigit4, shapeDigit5, shapeDigit6, shapeDigit7, shapeDigit8, shapeDigit9, shapePositionAbove

EXAMPLE Example 1:

```
PlotShapes( IIF( buy, shapeDigit9 + shapePositionAbove, shapeNone ),
colorGreen );
```

Example 2:

```
Graph0=MACD();
Graph1=Signal();
Buy=Cross(Graph0, Graph1);
Sell=Cross(Graph1, Graph0);
PlotShapes( ( Buy OR Sell ) * ( 1 + Cum( Buy OR Sell ) % 52 ), IIf(
Buy, colorGreen, colorRed ), 5 );
GraphXSpace = 5;
```

SEE ALSO PLOT() function

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2003-07-01 09:31:04	You can position your arrows relative to High/Low too. See the code below for the example:
	
	Buy=Cross(MACD(), Signal());
	
	Sell=Cross(Signal(), MACD());
	
	shape = Buy * shapeUpArrow + Sell * shapeDownArrow;
	
	Plot(Close, "Price", colorBlack, styleCandle);
	
	PlotShapes(shape, IIf(Buy, colorGreen, colorRed), 0, IIf(Buy, Low, High));
	
	GraphXSpace = 5;

Tomasz Janeczko tj --at-- amibroker.com 2006-06-07 17:14:40	ShapePositionAbove must NOT be used together with shapes that have positions already included (all those which have "Down" or "Up" in the name). All "Down" shapes are positioned ABOVE already and "Up" shapes are positioned BELOW already, so it makes no sense to add those two. So you may only use ShapePosition above to the following shapes: shapeSmallSquare, shapeHollowSmallSquare, shapeSquare, shapeHollowSquare, shapeSmallCircle, shapeHollowSmallCircle, shapeCircle, shapeHollowCircle, shapeStar, shapeHollowStar, shapeDigit0, shapeDigit1, shapeDigit2, shapeDigit3, shapeDigit4, shapeDigit5, shapeDigit6, shapeDigit7, shapeDigit8, shapeDigit9

References:

The **PlotShapes** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- 4% Model - Determine Stock Market Direction
- Abhimanyu
- Adaptive Centre of Gravity
- Adaptive Cyber Cycle
- Adaptive Relative Vigour Index
- Add SL/TGT other params to any strategy
- Advanced MA system
- Advanced Trend Lines with S & R
- Advisory NRx price chart display.
- AFL Example
- AFL Example - Enhanced
- ALJEHANI
- AllinOneAlerts - Module
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF

- An n bar Reversal Indicator
- Aroon The Advisor
- Auto Fib Ext&Retracement
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Awsome Oscilator
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Brian Wild
- Button trading using AB auto trading interface
- Caleb Lawrence
- Candle Stick Analysis
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- Centre of Gravity
- Chandelier Exit
- Chandelier Exit or Advanced Trailing Stop
- channel indicator
- Channel/S&R and trendlines
- com-out
- Congestions detection
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Cyber Cycle
- Darvas Wisestocktrader
- Double Super Trend System
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Effective Swing Indicator
- ekeko price chart
- EKEKO SAR-MF
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder safe Zone Long + short
- Expiry Thursday for Indian markets
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Oscillator
- Fisher Relative Vigour Index
- Fre
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Gordon Rose
- Halftrend
- Harmonic Pattern Detection
- Heikin Ashi Delta
- Hilbert Sine Wave Support & Resistance
- Hull Moving Average

- Hurst "Like" DE
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- IchimokuBrianViorelRO
- IFT of RSI - Multiple TimeFrames
- Indicator Explorer (ZigZag)
- interactively test discretionary trading
- Intraday Fibonacii Trend Break System
- Intraday Range and Periods Framer
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- Lagging MA-Xover
- Last Five Trades Result Dashboard – AFL code
- Least Squares Channel Indicator
- Linear Candle
- Lunar Phases - original
- LunarPhase
- MACD BB Indicator
- Meu Sistema de Trading - versão 1.0
- MFE and MAE and plot trades as indicator
- mitalpradip
- Modified Head & Shoulder Pattern
- Momentum Volume Price (MVP) Indicator
- Moving Averages NoX
- Nadaraya-Watson Envelope
- New HL Scanner
- Open Range Breakout Trading System
- pattenz
- Perceptron
- Pivot End Of Day Trading System
- Pivot Finder
- Polyfit Lines
- Range Filter - Trading Strategy
- Relative Momentum Index (RMI)
- Relative Vigour Index
- Renko Chart
- RI - Auto Trading System
- RUTVOL timing signal with BB Scoring routine
- shailu lunia
- Stan Weinstein strategy
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stops on percentages
- Stress with SuperSmoother
- Super Trend Indicator
- Super Trend Indicator
- Support and resistance
- suresh
- TD Moving Average 2
- TD REI

- TD sequential
- TD Sequential
- Three Day Balance Point
- Trading ATR 10-1
- Trailing Stop Loss
- Trend Detection
- Trend Following System
- TrendingRibbonArrowsADX
- Triangular Moving Average new
- TSV
- TTM Squeeze
- TWS auto-export Executions-file parser
- TWS trade plotter
- Updated Renko Chart
- Using From and To dates from Auto Analysis in Code
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- Visualization of stoploses and profit in chart
- Volatility
- Volatility System
- Volume Occilator
- Volume Zone Oscillator
- Woodie's CCI Panel Full Stats
- Zig Zag Indicator with Valid Entry and Exit Points
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

PlotText
- write text on the chart

Indicators
(AmiBroker 4.80)

SYNTAX **PlotText("text", x, y, color, bkcolor = colorDefault, yoffset = 0)**

RETURNS NOTHING

FUNCTION This function writes text in specified co-ordinates.

where:

- **x** - is x-coordinate in bars (like in LineArray)
- **y** - is y-coordinate in dollars
- **color** is text color
- **bkcolor** is background color
- **yoffset** (new in 5.80) is a Y-axis offset in pixels

If bkcolor is NOT specified (or equal to colorDefault) text is written with TRANSPARENT background, any other value causes solid background with specified background color.

EXAMPLE

```
Plot(C,"Price", colorBlack, styleLine );
Plot(MA(C,20),"MA20", colorRed );

Buy=Cross( C, MA(C,20 ) );
Sell= Cross( MA( C, 20 ), C );

dist = 1.5*ATR(10);

for( i = 0; i < BarCount; i++ )
{
if( Buy[i] ) PlotText( "Buyn@" + C[ i ], i, L[ i ]-dist[i],
colorGreen );
if( Sell[i] ) PlotText( "Selln@" + C[ i ], i, H[ i ]+dist[i],
colorRed, colorYellow );
}

PlotShapes( Buy * shapeUpArrow + Sell * shapeDownArrow, IIf( Buy,
colorGreen, colorRed ) );
```

SEE ALSO PLOT() function , PlotTextSetFont() function

References:

The **PlotText** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Auto Fib Ext&Retracement
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Caleb Lawrence

- Channel/S&R and trendlines
- Congestions detection
- Day Bar No
- Expiry day/days - Last thursday of month
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fre
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann Swing Charts in 3 modes with text
- Halftrend
- Harmonic Patterns
- High Low Detection code
- Hilbert Sine Wave Support & Resistance
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- MFE and MAE and plot trades as indicator
- New HL Scanner
- Ord Volume
- Pivots And Prices
- Point & figure Chart India Securities
- Prior Daily OHLC
- Range Filter - Trading Strategy
- Renko Chart
- Square of Nine Roadmap Charts
- SUPER PIVOT POINTS
- suresh
- Trix Bars number
- Updated Renko Chart
- visual turtle trading system
- Visualization of stoploses and profit in chart
- White Theme
- ZigZag Hi Lo Barcount
- ZigZag Retracements

More information:

See updated/extended version on-line.

PlotTextSetFont - write text on the chart with user-defined font

Basic price pattern detection
(AmiBroker 4.80)

SYNTAX **PlotTextSetFont("text", "fontname", pointsize, x, y, color, bkcolor = colorDefault, yoffset = 0)**

RETURNS NOTHING

FUNCTION This function writes text in specified co-ordinates using specified font

where:

- **'text'** is a text to display
- **'fontname'** is a type face name
- **pointsize** is a font size in points
- **x** - is x-coordinate in bars (like in LineArray)
- **y** - is y-coordinate in dollars
- **color** is text color
- **bkcolor** is background color
- **yoffset** is Y-axis offset (in pixels)

If bkcolor is NOT specified (or equal to colorDefault) text is written with TRANSPARENT background, any other value causes solid background with specified background color.

The function also sets font for all subsequent calls to PlotText().

EXAMPLE

```
Plot(C, "Price", colorDefault );
x = SelectedValue( BarIndex() );
y = Close[ x ];

PlotTextSetFont("E", "Wingdings", 40, x, y, colorGreen,
colorDefault, -30 );
PlotText( "C",  BarCount, Close[ BarCount - 1 ], colorRed,
colorDefault, -20 ); // will use new font too
```

SEE ALSO PlotText() function

References:

The **PlotTextSetFont** function is used in the following formulas in AFL on-line library:

- AllinOneAlerts - Module
- Harmonic Pattern Detection
- Inside Bar
- Inside Bar
- pattenz
- Support and resistance
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

PlotVAPOverlay - plot Volume-At-Price overlay chart

Exploration / Indicators
(AmiBroker 4.40)

SYNTAX **PlotVAPOverlay(lines = 300, width = 5, color = colorGreen, vapstyle = 0);**

RETURNS NOTHING

FUNCTION Plots Volume-At-Price (VAP) overlay chart. Please note that there must be at least one regular Plot function in your formula for this to work, and there can be only one PlotVAPOverlay in one indicator. This function plots single segment for visible bars only. To plot multiple-segment VAP chart use **PlotVAPOverlayA** function.

- vapstyle = 0 - left side, area fill, on top of all plots
- vapstyle = 1 - right side, area fill, on top of all plots
- vapstyle = 2 - left side, lines only, on top of all plots
- vapstyle = 3 - right side, lines only, on top of all plots
- vapstyle = 4 - left side, area fill, behind all plots
- vapstyle = 5 - right side, area fill, behind all plots
- vapstyle = 6 - left side, lines only, behind all plots
- vapstyle = 7 - right side, lines only, behind all plots

EXAMPLE

```
Plot( Close, "Price", colorWhite, styleCandle );
PlotVAPOverlay( Param("lines",300,10,1000,1),
Param("width",10,1,99,1), ParamColor("color", colorDarkBlue),
Param("style",0,0,7,1) );
```

SEE ALSO PLOT() function , PlotVAPOverlayA() function

References:

The **PlotVAPOverlay** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

PlotVAPOverlayA - plot multiple-segment Volume-At-Price chart

Indicators
(AmiBroker 5.20)

SYNTAX **PlotVAPOverlayA(segments, lines = 300, width = 80, color = colorLightGrey, vapstyle = 4);**

RETURNS NOTHING

FUNCTION Plots multiple Volume At Price charts at user-defined points.

Parameters:

segments - is an array which holds 0 and 1 (False/True) values, where 1 indicates starting/ending point of each VAP segment

AmiBroker will draw as many segments as there are '1' in the array. Note that minimum segment length is 2, so if entire array is filled with 1-s only, it won't draw anything. In other words, there must be zeros (at least one) between 1's.

lines - number of vertical lines (resolution)

width - percentage width of the VAP overlay relatative to segment length (0..100)

color - the color of VAP chart

vapstyle

- vapstyle = 0 - left side, area fill, on top of all plots
- vapstyle = 1 - right side, area fill, on top of all plots
- vapstyle = 2 - left side, lines only, on top of all plots
- vapstyle = 3 - right side, lines only, on top of all plots
- vapstyle = 4 - left side, area fill, behind all plots
- vapstyle = 5 - right side, area fill, behind all plots
- vapstyle = 6 - left side, lines only, behind all plots
- vapstyle = 7 - right side, lines only, behind all plots

EXAMPLE Simplest example:

```
Plot(C, "Close", colorBlack, styleCandle );
segments = IIf( Interval() < inDaily, Day(), Month() ); // draw
daily or monthly VAP segments depending on display interval
segments = segments != Ref( segments , -1 );

PlotVAPOverlayA( segments );
```

More complex example:

```
_SECTION_BEGIN("Price");
SetChartOptions(0,chartShowArrows|chartShowDates);
_N(Title = StrFormat("{{NAME}} - {{INTERVAL}} {{DATE}} Open %g, Hi
%g, Lo %g, Close %g (%.1f%%) {{VALUES}}", O, H, L, C, SelectedValue(
ROC( C, 1 ) ) ));
Plot( C, "Close", ParamColor("Color", colorBlack ), styleNoTitle |
ParamStyle("Style") | GetPriceStyle() );
_SECTION_END();

_SECTION_BEGIN("VAP");
segments = IIf( Interval() < inDaily, Day(), Month() );
segments = segments != Ref( segments , -1 );

PlotVAPOverlayA( segments , Param("Lines", 300, 100, 1000, 1 ),
Param("Width", 80, 1, 100, 1 ), ParamColor("Color", colorGold ),
ParamToggle("Side", "Left|Right" ) | 2 * ParamToggle("Style",
"Fill|Lines", 0) | 4*ParamToggle("Z-order", "On top|Behind", 1 ) );
Plot(segments, "", colorLightGrey, styleHistogram | styleOwnScale );
_SECTION_END();
```

SEE ALSO PLOTVAPOVERLAY() function

References:

The **PlotVAPOverlayA** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

PopupWindow - display pop-up window

Miscellaneous functions
(AmiBroker 50)

SYNTAX **PopupWindow(bodytext, captiontext, timeout = 5, left = -1, top = -1, width = -1, height = -1, captureFocus = True);**

RETURNS NOTHING

FUNCTION The function creates and displays pop-up window with specified bodytext, captiontext.

Parameters:

- bodytext - the string containing the text of the window body
- caption - the string containing the text of window caption
- timeout - auto-close time in seconds (default 5 seconds)
- left - top-left corner X co-ordinate (default = -1 -means auto-center)
- top - top-left corner Y co-ordinate (default = -1 - means auto-center)
- width = width in pixels, -1 - use default width
- height = height in pixels, -1 - use default height
- captureFocus - decides whenever popup window captures input focus or not

EXAMPLE

```
if( ParamTrigger("Display Popup Window", "Press here" ) )
{
    PopupWindow("Current time is: " + Now(),"Alert", 2,
640*mtRandom(), 480*mtRandom());
}
```

SEE ALSO ParamColor() function , ParamDate() function , ParamField() function , ParamList() function , PARAMSTR() function , ParamStyle() function , ParamTime() function , ParamToggle() function , ParamTrigger() function

References:

The **PopupWindow** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically
- AFL_Glossary_Converter
- AllinOneAlerts - Module
- For Auto Trading Setup
- Gfx Toolkit
- Heatmap V1
- How to add IB Option Symbols
- MFE and MAE and plot trades as indicator

More information:

See updated/extended version on-line.

Prec
- adjust number of decimal points of floating point number

Math functions

SYNTAX	**Prec(ARRAY, *precision*)**
RETURNS	ARRAY
FUNCTION	Truncates ARRAY to *precision* decimal places.
EXAMPLE	The formula "prec(10.12981, 2)" returns 10.120. The formula "prec(10.12981, 4)" returns 10.12980.
SEE ALSO	

References:

The **Prec** function is used in the following formulas in AFL on-line library:

- Alphatrend
- CCI Woodies Style
- Controlling Height of Volume Bars
- For Auto Trading Setup
- GFX ToolTip
- Graphical sector analysis
- Graphical sector stock amalysis
- Harmonic Pattern Detection
- High Low Detection code
- Open Range Breakout Trading System
- pattenz
- Relative Strength Multichart of up to 10 tickers
- Tracking Error
- Tracking Error
- Triangle exploration using P&F Chart
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

Prefs - retrieve preferences settings

Miscellaneous functions
(AmiBroker 3.40)

SYNTAX	**Prefs(index)**
RETURNS	NUMBER, STRING

FUNCTION	Retrieves preferences setting. Allowed *index* values are:
	0: FatLineChart;
	1: MarkQuotations;
	2: ChartVolumeType;
	3: ShortTimeMA;
	4: STMARange;
	5: MidTimeMA;
	6: MTMARange;
	7: BollingerBands;
	8: Pref.BBFactor;
	9: ROC;
	10: RSI;
	11, 12, 13: MACD;
	14: StochSlow;
	15, 16, 17: Ultimate;
	18: VolumeType;
	19-22: /* reserved amiga only */
	23: AutoArrange;
	24: LogChartScale;
	25: MaxChartQuot;
	26: TRIX;
	27: LongTimeMA;
	28: LTMARange;
	29: VolMARange;
	30: RelativeStrengthBase (string);
	31: LimitSave;
	32: LimitSaveRange;
	33: CCI;
	34: CCIAvg;
	35: Tooltips;
	36: MFI;
	37, 38: Chaikin;
	39: DataPath (string)
	40: DataTooltips;
	41: LoadAllWhenSelect;
	42: PartialLoad;
	43: PartialLoadQty;
	44, 45: TRIN;
	46: STMAType;
	47: MTMAType;
	48: LTMAType;
	49: ADX;
	50, 51: ParabolicSAR;
	52: EnableMainChartSAR;

53: DefaultPriceStyle;
54: StockTreeMode;
55: TickerListMode;

EXAMPLE macd(prefs(11), prefs(12), prefs(13));

SEE ALSO

References:

The **Prefs** function is used in the following formulas in AFL on-line library:

- Dinapoli Guru Commentary
- ekeko price chart
- hassan
- Moving Trend Bands (MTB)

More information:

See updated/extended version on-line.

PriceVolDistribution - general-purpose distribution function

Matrix functions
(AmiBroker 6.20)

SYNTAX **PriceVolDistribution(priceH, priceL, vol, bins, absolute = False, startbar = 0, endbar = -1)**

RETURNS MATRIX

FUNCTION The function is general purpose frequency distribution function. It is typically used to create distribution of volume versus price but this is only one of many possible uses. The function takes priceH, priceL and vol arrays and outputs distribution MATRIX with price levels in first column and relative volume at that level in second column. The bins parameter decides how many price 'bins' the distribution will have. The absolute parameter decides whenever returned volumes are absolute sum or relative (0...1) values. The startbar and endbar decides which bars from input array should be used.

EXAMPLE

```
// a demo showing
// re-implementation of VAP overlay using
// PriceVolDistribution and low-level graphics
bi = BarIndex();
fvb = FirstVisibleValue( bi );
lvb = LastVisibleValue( bi );

mx = PriceVolDistribution( H, L, V, 100, False, fvb, lvb );

GfxSetCoordsMode( 1 );

GfxSelectPen( colorRed );

bins = MxGetSize( mx, 0 );
for( i = 0; i < bins; i++ )
{
price = mx[ i ][ 0 ]; // price level
relvolume = mx[ i ][ 1 ]; // relative volume 0..1
relbar = relvolume * (lvb-fvb+1);
GfxMoveTo( fvb, price );
GfxLineTo( fvb + relbar, price );
}

Plot( C, "Price", colorDefault, styleBar );

if( ParamToggle("BuildinVAP", "No|Yes") ) PlotVAPOverlay( 100, 100,
colorGreen, 2 );
```

SEE ALSO

References:

The **PriceVolDistribution** function is used in the following formulas in AFL on-line library:

- Multi-color Volume At Price (VAP)
- Volume based support resistance price finder

More information:

See updated/extended version on-line.

printf
- Print formatted output to the output window.

String manipulation
(AmiBroker 4.50)

SYNTAX **printf(** ***formatstr*****, ...)**

RETURNS NOTHING

FUNCTION The **printf** function formats and prints a series of characters and values to the output window, which can be either commentary or interpretation window.

If arguments follow the format string, the format string must contain specifications that determine the output format for the arguments.

printf and StrFormat behave identically except that printf writes output to the window, while StrFormat does not write anything to output window but returns resulting string instead.

Note 1: for numbers always use %f, %e or %g formatting, %d or %x will not work because there are no integers in AFL.

Note 2: as of now only numbers and arrays can now be printed. For arrays 'selected value' is printed

Note 3: to print a single percent-sign character, you can not type % alone, you must use %%.

Note 4: read more about various % format specifiers on:
https://www.google.com/search?q=printf+format+specifiers

Starting from version 6.10, printf/StrFormat now implement a check for correct formatting string as sometimes users passed strings with % that is special marker for formatting string instead of %% to print actual percent sign. When check failes, "Error 61. The number of % formatting specifier(s) does not match the number of arguments passed." is displayed

Starting from version 6.20, printf/StrFormat support now "%s" (string specifier)

EXAMPLE

```
for( i = 0; i < 10; i++ )
{
   printf( "Hello world, line %g\n", i );
}
```

SEE ALSO StrFormat() function

References:

The **printf** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_1
- AR_Prediction.afl
- Auto Trade Step by Step
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Backup Data of 1min Interval
- Button trading using AB auto trading interface
- Congestions detection

- DateNum_DateStr
- ekeko price chart
- elliott wave manual labelling
- ICHIMOKU SIGNAL TRADER
- interactively test discretionary trading
- Manual Bracket Order Trader
- MFE and MAE and plot trades as indicator
- Multiple sinus noised
- nth (1 - 8) Order Polynomial Fit
- Polyfit Lines
- prakash
- Profit Table (Color Coded)
- Reconnect TWS
- Robert Antony
- Square of Nine Roadmap Charts
- Stochastic %J - KDJ
- tomy_frenchy
- Trigonometric Fit - TrigFit with AR for cos / sin
- Ultimate plus
- Visi-Trade
- WLBuildProcess

More information:

See updated/extended version on-line.

Prod - cumulative product of array over specified range

Moving averages, summation
(AmiBroker 6.20)

SYNTAX	**Prod(array, range)**
RETURNS	ARRAY
FUNCTION	The function returns cumulative product of array elements over specified number of bars (range). Note: when range is less or equal zero, product is equal 1.
EXAMPLE	
SEE ALSO	ProdSince() function

References:

The **Prod** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

ProdSince - cumulative product since condition is met

Moving averages, summation
(AmiBroker 6.20)

SYNTAX	**ProdSince(condition, array)**
RETURNS	ARRAY
FUNCTION	The function returns cumulative product of array elements since condition is met. On bar when condition is met, initial product value is 1.
EXAMPLE	
SEE ALSO	SumSince() function , Sum() function

References:

The **ProdSince** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

PVI - positive volume index

Indicators

SYNTAX	**pvi()**
RETURNS	ARRAY
FUNCTION	Calculates the Positive Volume Index.
EXAMPLE	
SEE ALSO	The nvi() function

References:

The **PVI** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Random
- random number

Statistical functions
(AmiBroker 3.90)

SYNTAX	**Random(** ***seed = Null*** **)**
RETURNS	ARRAY
FUNCTION	Returns an array of random values in 0..1 range (to get single random value use LastValue(Random())) Seed is defined it initializes the seed of random number generator this allows to produce repetitive series of pseudo-random series. If seed is not specified - random number generator continues generation. To reinitialize the generator, use 1 as the seed argument. Any other value for seed sets the generator to a random starting point.

EXAMPLE	Example 1:
	
	`Graph0 = Random(); // generates different sequence with each refresh`
	
	Example 2:
	
	`Graph0 = Random(1); // generates the same sequence with each refresh`

SEE ALSO	mtRandom() function

Comments:

Tomasz Janeczko
tj --at-- amibroker.com
2007-02-27 09:42:14
	Internally Random() function uses Microsoft C runtime library rand() function scaled to cover 0..1 range:
	(1.0*rand()/RAND_MAX)
	
	Now Microsoft's rand() function (used in all MS languages) is Linear Congruential Pseudo-Random Number Generator coded using 32 bit integer arithmetic as follows:
	
	static long holdrand;
	int rand()
	{
	holdrand = holdrand * 214013 + 2531011;
	return (holdrand >> 16) & 0x7fff;
	}

References:

The **Random** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically
- How to add IB Option Symbols
- MFE and MAE and plot trades as indicator
- Multiple sinus noised
- Random Walk
- Randomize()

More information:

See updated/extended version on-line.

Ref - reference past/future values of the array

Trading system toolbox

SYNTAX	**Ref(ARRAY, *period*)**
RETURNS	ARRAY
FUNCTION	References a previous or subsequent element in a ARRAY. A positive *period* references "n" periods in the future; a negative *period* references "n" periods ago. The function accepts periods parameter that can be constant as well as time-variant (array).
EXAMPLE	The formula "ref(CLOSE, -14)" returns the closing price 14 periods ago. Thus, you could write the 14-day price rate-of-change (expressed in points) as "C - ref(C, -14)." The formula "ref(C, 12)" returns the closing price 12 periods ahead (this means looking up the future)
SEE ALSO	

References:

The **Ref** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 2 Timeframes Candlestick Bar Chart
- 3 Price Break
- 30 Week Hi Indicator - Calculate
- 3TF Candlestick Bar Chart
- 4% Model - Determine Stock Market Direction
- 52 Week New High-New Low Index
- AC+ acceleration
- Adaptave Zones O/B & O/S Oscillator
- Adaptive Centre of Gravity
- Adaptive Cyber Cycle
- Adaptive Price Channel
- Adaptive Relative Vigour Index
- Advanced MA system
- Advanced Trend Lines with S & R
- Advisory NRx price chart display.
- ADX Indicator - Colored
- ADXbuy
- ADXR
- ADXVMA
- AFL Example
- AFL Example - Enhanced
- Against all odds
- AJDX system
- Alert Output As Quick Rewiev
- AllinOneAlerts - Module
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- An n bar Reversal Indicator
- Analytic RSI formula
- Andrews Pitchfork

- Andrews PitchforkV3.3
- AO+ Momentum indicator
- AO+Momentum
- Application of Ehler filter
- Aroon
- AR_Prediction.afl
- ATR Study
- Auto Analysis Closing Price Reversal
- Auto Analysis Hook Reversal
- Auto Analysis Island Reversal
- Auto Analysis Key Reversal
- Auto Analysis Open/Close Reversal
- Auto Analysis Pivot Point Reversal
- Auto Analysis Short-term Reversals Exploration
- Auto Fib Ext&Retracement
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- Average Dollar Price Volatility Exploration
- Awsome Oscilator
- babaloo chapora
- Bad Tick Trim on 5 sec database
- Balance of Power
- balance of power
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bman's HaDiffCO
- Bollinger Band Gap
- Bollinger band normalization
- Bollinger Fibonacci Bands
- Bottom Fisher Exploration
- Bow tie
- Brian Wild
- Buff Volume Weighted Moving Averages
- Bull/Bear Volume
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Buyer Seller Force
- Caleb Lawrence
- CAMSLIM Cup and Handle Pattern AFL
- Candle Identification
- Candle Pattern Function
- Candle Stick Analysis
- candlestick chart for Volume/RSI/OBV
- CandleStick Comentary--Help needed
- Candlestick Commentary
- Candlestick Commentary Modified

- Candlestick Commentary-modified
- Candlestick Volume Bars with Moving Average
- CandleStochastics
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- CCI(20) Divergence Indicator
- CCT Coppock Curve
- CCT Kaleidoscope
- Centre of Gravity
- Chande Momentum Oscillator
- Chande's Trend Score
- changing period moving avarage
- Channel/S&R and trendlines
- CoinToss ver 1
- Cole
- Color Coded Short Term Reversal Signals
- Color MACD Histogram Changes
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- com-out
- Commodity Channel Index
- Commodity Selection Index (CSI)
- Compare Sectors against Tickers
- Composite Index
- Congestions detection
- Connors TPS
- ConnorsRSI
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Coral Trend Indicator
- crBeta
- CVR--severe filter
- Cyber Cycle
- Cybernertic Hilbert Sine Wave
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period
- Dahl Oscillator modified
- Daily High Low in Advance
- danningham penetration
- Darvas Amibroker
- Darvas Johndeo Research
- Darvas Wisestocktrader
- Dave Landry PullBack Scan
- Dave Landry Pullbacks
- Day Bar No
- Demand Index
- Demand Index
- DeMarker
- Detailed Equity Curve

- Digital indiactors
- Dinapoli Guru Commentary
- DiNapolis 3x Displaced Moving Averages
- Distance Coefficient Ehlers Filter
- Divergence indicator
- Divergences
- DMI Spread Index
- Donchian Channel
- Double top detection
- DPO with shading
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Ed Seykota's TSP: EMA Crossover System
- Effective Swing Indicator
- Ehler's filters and indicators
- Ehlers Center of Gravity Oscillator
- Ehlers CyberCycle
- Ehlers Dominant Cycle Period
- Ehlers Fisher Transform
- Ehlers Instantaneous Trend
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- EKEKO SAR-MF
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- Elder's SafeZone Stop
- ElderSafeZoneStopLong
- ElderSafeZoneStopShort
- Ema bands
- EMA Crossover
- End Of Year Trading
- Evaluating Candle Patterns in a trading system
- Even Better Sinewave Indicator
- Fast Refreshed KAGI Swing Charts (Price Swing)
- FastStochK FullStochK-D
- Fib CMO
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fisher Centre of Gravity
- Fisher Cyber Cycle
- Fisher Relative Vigour Index
- Follow the Leader
- For Auto Trading Setup
- Force index
- Forward/Reverse EMA by John Ehlers
- Fre
- Frequency distribution of returns
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY

- Fund Screener
- Future MA Projection
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Futures - Dollar Move Today Indicator
- Gann Five Day pullback
- Gann HiLo Indicator and System
- Gann Swing Chart
- Gann Swing Charts in 3 modes with text
- Gap and Circuit
- garythompson
- garythompson
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Halftrend
- Harmonic Patterns
- hassan
- Heatmap V1
- Heikin Ashi Delta
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- HH-LL-PriceBar
- High Low Detection code
- Hilbert Study
- Historical Volotility Scan - 6/100
- Historical Volotility Scan - 50 Day
- Hook Reversals
- Hull Rate of Return Indicator
- Hurst "Like" DE
- Hurst Constant
- IBD relative strength database ranker
- Ichimoku Chart
- Ichimoku charts
- ICHIMOKU SIGNAL TRADER
- Ichimoku with plot mofified to use cloud function
- IchimokuBrianViorelRO
- IFT of RSI - Multiple TimeFrames
- Improved NH-NH scan / indicator
- Index and ETF trading
- Index of 30 Wk Highs Vs Lows
- Indicator Explorer (ZigZag)
- Intraday Average Volume
- INTRADAY HEIKIN ASHI new
- IntraDay Open Marker
- Intraday Range and Periods Framer
- Intraday Strength
- Intraday Volume EMA
- JEEVAN'S SRI CHAKRA
- Kagi Chart
- Kiss and Touch with the Modified True Range
- Lagging MA-Xover

- Larry William's Volatility Channels
- Least Squares Channel Indicator
- Linear Candle
- Linear Regression Line w/ Std Deviation Channels
- LSMA
- Lunar Phases - original
- MACD BB Indicator
- MACD commentary
- MACD indicator display
- Main price chart with Rainbow & SAR
- Manual Bracket Order Trader
- Market Direction
- Market Facilitation Index VS Volume
- Market Meanness Index
- Market Profile
- MAVG
- Meu Sistema de Trading - versão 1.0
- MFE and MAE and plot trades as indicator
- mfimacd
- mitalpradip
- Modified Momentum Finder DDT-NB
- Modified-DVI
- Momentum
- Momentum
- Momentum Volume Price (MVP) Indicator
- Monthly bar chart
- Monthly Coppock Guide
- Moving Average "Crash" Test
- Moving Averages NoX
- MO_CrashZone
- MS Darvas Box with Exploration
- MultiCycle 1.0
- Multiple Ribbon Demo
- N Line Break Chart
- N-period candlesticks (time compression)
- Neural Network Powered Smooth/Predictive RSI V2
- New HL Scanner
- Nick
- Nonlinear Ehlers Filter
- Noor_Doodie
- Now Send Push Notifications From Amibroker
- NR4 Historical Volatility System
- NRx Exploration
- Open Range Breakout Trading System
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- Parametric Chande Trendscore
- pattenz
- Pattern - Rectangle Base Breakout on High Vol
- Pattern_-_Rectangle_Base_Breakout_on_High_Vol 2
- Perceptron
- Performance Check

- Periodically ReBalance a BUY & HOLD Portfolio
- Peterson
- PF Chart - Close - April 2004
- Pivot End Of Day Trading System
- Pivot Point with S/R Trendlines
- Pivots for Intraday Forex Charts
- Point & figure Chart India Securities
- Polarized Fractal Efficiency
- Polyfit Lines
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- Positive Bars Percentage
- Prashanth
- Price Persistency
- Price with Woodies Pivots
- Price-Volume Rank
- Prior Daily OHLC
- Probability Calculator
- Probability Density & Gaussian Distribution
- Profit Table (Color Coded)
- Projection Oscillator
- Pullback System No. 1
- PVT Trend Decider
- Rainbow Charts
- Rainbow Oscillator
- Random Walk Index
- Random Walk Index, base formula included
- Range Expansion Index
- Range Filter - Trading Strategy
- Rapid Prototyping Method for System Development
- Raw ADX
- Rea Time Daily Price Levels
- Regression Analysis Line
- Relative Momentum Index (RMI)
- Relative Strength Multichart of up to 10 tickers
- Relative Vigour Index
- Relative Vigour Index
- Renko Chart
- Renko Chart
- Reverse EMA function
- Revised Renko chart
- ROC of MACD Weekly
- RSI of Weekly Price Array
- RSIS
- RUTVOL timing signal with BB Scoring routine
- Sainath Sidgiddi
- Scan New High and New Low
- Schiff Lines
- Sector Tracking
- SectorRSI
- shailu lunia
- Simple Momentum

- Sine Wave Indicator
- Smoothed Adaptive Momentum
- Sony
- Stan Weinstein strategy
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Stochastic Centre of Gravity
- Stochastic Cyber Cycle
- Stochastic Divergence, negative
- Stochastic Divergence, positive
- Stochastic Divergences, PDI, NDI
- Stochastic Fast%K and Full
- Stochastic of Weekly Price Array
- Stochastic Relative Vigour Index
- Stochastic RSI
- Stops Implementation in AFS
- Strength and Weakness
- Stress with SuperSmoother
- SUPER PIVOT POINTS
- Support and resistance
- swing chart
- TAPE READING
- TAZ Trading Method Exploration
- TD Moving Average I
- TD REI
- TD sequential
- TD Sequential
- testing multiple system simulataneously
- The Mean RSIt
- The Mean RSIt (variations)
- The Relative Slope
- The Relative Slope Pivots
- The Saturation Indicator D_sat
- The Three Day Reversal
- Three Day Balance Point
- Three Line Break - TLB
- Time Frame Weekly Bars
- Time Left to Current Bar
- Time segment value
- Tom DeMark Trend Lines
- tomy_frenchy
- Tracking Error
- Tracking Error
- Trade day of month
- Trades per second indicator
- Trading ATR 10-1
- Trailing Stop Loss
- Trend Analysis_Comentary
- Trend Continuation Factor
- Trend Detection
- Trend Exploration: Count Number of New Highs
- Trend Following System
- Trend Trigger Factor

- Triangle exploration using P&F Chart
- Triangle search
- Triangle Search Extended
- Triangular Moving Average
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- TRIX
- Trix Bars number
- TSV
- TTM Squeeze
- Tushar Chande's Projected Range
- Twiggs Money Flow
- Twiggs money flow weekly
- Updated Renko Chart
- ValueChart
- Varexlist
- Vertical Horizontal Filter
- Vertical Horizontal Filter (VHF)
- Vic Huebner
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- Vivek Jain
- Volatility Quality Index
- Volatility System
- Volume based support resistance price finder
- Volume Color with Dynamic Limit
- Volume Divided Histogram
- Volume Occilator
- Volume Oscillator
- Volume Zone Oscillator
- VolumeDivAvgV3_Study_BrS
- VSTOP (2)
- VWAP - Volume Weighted Average Price
- VWAP versus Average Price
- VWAP with standard deviation bands
- Weekly chart
- Weekly Trend in Daily Graph
- Weinberg's The Range Indicator
- William's Alligator System II
- Williams Alligator system
- WILSON RELATIVE PRICE CHANNEL
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- Woodie's Price Panel With Woodie's Pivots
- Woodies CCI
- Zig Explorer
- Zig Zag Indicator with Valid Entry and Exit Points
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount

- ZigZag Retracements

More information:

See updated/extended version on-line.

RelStrength - comparative relative strength

Referencing other symbol data
(AmiBroker 3.30)

SYNTAX **RelStrength("tickername", fixup = 1)**

RETURNS ARRAY

FUNCTION Calculates relative strength of currently selected security compared to "tickername" security. When you give an empty string as argument, a standard relative strength base security taken from Stock->Categories will be used.
The last parameter - fixup - with the default value of 1 - causes filling the holes in the data with previous values (behaviour introduced in 3.90.3), if fixup is 0 - the holes are not fixed (the old, pre-3.90.3 behaviour)
Note: you can still use Foreign/RelStrength in the old way:
Foreign("ticker", "field"), RelStrength("ticker") - then the holes will be fixed.

EXAMPLE relstrength("^DJI")

SEE ALSO

Comments:

jayson 2003-06-23 09:20:02	Interpretation Comparative Relative Strength compares a security's price change with that of a "base" security. When the Comparative Relative Strength indicator is moving up, it shows that the security is performing better than the base security. When the indicator is moving sideways, it shows that both securities are performing the same (i.e., rising and falling by the same percentages). When the indicator is moving down, it shows that the security is performing worse than the base security (i.e., not rising as fast or falling faster). Comparative Relative Strength is often used to compare a security's performance with a market index. It is also useful in developing spreads (i.e., buy the best performer and short the weaker issue).

References:

The **RelStrength** function is used in the following formulas in AFL on-line library:

- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Relative Strength
- Relative strength comparison with moving average
- Stan Weinstein strategy

More information:

See updated/extended version on-line.

Remap
- re-maps one range to another

Math functions
(AmiBroker 6.30)

SYNTAX **Remap(value, srcMin, srcMax, dstMin, dstMax)**

RETURNS NUMBER or ARRAY

FUNCTION The function re-maps a value (number or array) from 'source' to 'destination' range. That is, a value of srcMin would get mapped to dstMin, a value of srcMax to dstMax, values in-between to values in-between, etc. Does not constrain values to within the range, because out-of-range values are sometimes intended and useful.

Parameters:

- value - the value to be remapped
- srcMin, srcMax - minimum and maximum value of source range
- dstMin, dstMax - minimum and maximum value of destination range

Depending on input value type (number or array) the result will be also number or array.

Mathematically the function performs the following linear transformation:

```
result = (value - srcMin) * (dstMax - dstMin) / (srcMax - srcMin) +
dstMin;
```

Typical usage of the function is to convert indicator from one range to another. Say you have an indicator that ranges from -1..1 and you want it to be bounded in 0..100 space. Then you can use Remap(value, -1, 1, 0, 100);

Or you want to convert pixels to prices or bars or vice versa as shown in the example below

EXAMPLE

```
height = Status("pxheight");
pxmin = Status("pxchartleft");
pxmax = Status("pxchartright");
barmin = Status("firstvisiblebar");
barmax = Status("lastvisiblebar")+1;

GfxSelectPen( colorRed );

for( bar = barmin; bar < barmax; bar++ )
{
   // re-map bar numbers to pixel co-ords
   x = Remap( bar, barmin, barmax, pxmin, pxmax );
   // draw vertical lines where bars are
   GfxMoveTo( x, 0 );
   GfxLineTo( x, height );
}
```

SEE ALSO Status() function

References:

The **Remap** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

RequestMouseMoveRefresh - request chart to be refreshed when user moves mouse cursor

Indicators
(AmiBroker 6.30)

SYNTAX	**RequestMouseMoveRefresh()**
RETURNS	NOTHING
FUNCTION	The function causes that the chart is refreshed (and formula re-executed) when user moves mouse cursor over the chart area
EXAMPLE	
SEE ALSO	

References:

The **RequestMouseMoveRefresh** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

RequestTimedRefresh
- forces periodical refresh of indicator pane

Indicators
(AmiBroker 4.90)

SYNTAX **RequestTimedRefresh(interval, onlyvisible = True)**

RETURNS NOTHING

FUNCTION The function causes given indicator window to refresh automatically every seconds regardless of data source used or connection state.

interval parameter defines timeout in seconds between refreshes. AmiBroker attempts to align refreshes to second boundaryso if you call it RequestTimedRefresh(5) you should get refreshes at 0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 and 55 second of the minute.Due to the way how regular (low overhead) timers are implemented in Windows they have accurracy of +/-55ms providedthat CPU is not very busy. Don't expect to get first line of your code to execute exactly at .000 milliseconds. This varies depending on machine load, number of quotes, system time slice and tens of other factors.Usually (on my testing machines) the first line of the code executes anywhere in the first 100 ms of the second, provided that other processes do not interfere. Windows is not real-time operating system and it does not guarantee any fixed execution/reaction times.

onlyvisible parameter set to True (default value) means that refreshes are triggered only for visible and not minimised windows. This applies also to main AmiBroker window - when it is minimised charts are NOT refreshed by default. To force refreshes when window is minimised you need to set this parameter to False. Note that this visibility applies to mostly to 'minimised' state or the situation when you move chart outside the boundary of physical screen so it is not visible to an eye but still open. It does not apply to chart windows that are on placed on inactive sheets, as they do not really exist until they are shown (this way AmiBroker conserves memory and CPU) and as non-existing, can not be refreshed.

Hint: to detect whenever given refresh comes from timer or user action you can use Status("redrawaction") function. It returns 0 for regular refresh (user action) and 1 for timer-refresh

Starting from version 5.30.3 RequestTimedRefresh supports sub-second (down to 0.1 sec) resolution, when enabled via registry setting (HKCU/Software/TJP/Broker/Settings/EnableHiresRTR, DWORD value = 1)

EXAMPLE
```
RequestTimedRefresh( 5 );
// automatically refresh this particular chart every 5 seconds
```

SEE ALSO STATUS() function

References:

The **RequestTimedRefresh** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 2 Timeframes Candlestick Bar Chart
- 3 ways to use RMI in one script
- 3TF Candlestick Bar Chart
- AllinOneAlerts - Module
- Animated BackGround

- Animated BackGround 1.1
- Auto Trade Step by Step
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- CoinToss ver 1
- Continuous Contract Rollover
- elliott wave manual labelling
- For Auto Trading Setup
- GFX ToolTip
- Heatmap V1
- High Low Detection code
- How to add IB Option Symbols
- New HL Scanner
- Visi-Trade
- Volume Charts
- White Theme
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

RestorePriceArrays - restore price arrays to original symbol

Referencing other symbol data
(AmiBroker 4.50)

SYNTAX	**RestorePriceArrays(*tradeprices* = False)**
RETURNS	NOTHING
FUNCTION	The **RestorePriceArrays** restores original price and volume arrays after the call to **SetForeign**. *tradeprices* parameter has to match the one used in SetForeign() function. When *tradeprices* argument is set to TRUE, then not only OHLC, V, OI, Avg arrays are restored, but BuyPrice, SellPrice, ShortPrice, CoverPrice, PointValue, TickSize, RoundLotSize, MarginDeposit variables too. new in 5.90 - it can be also used to restore OHLC arrays that were directly overwritten by the user (without calling TimeFrameSet or Foreign).

EXAMPLE

```
// Example 1: Plot the indicator using foreign security data
SetForeign("MSFT");
Plot( Ultimate(), "Ultimate from MSFT", colorRed );
RestorePriceArrays();

// Example 2: Use SetForeign with Equity function
SetForeign("MSFT", True, True );
Buy = Cross( MACD(), Signal());
Sell = Cross( Signal(), MACD());
e = Equity(); // backtest on MSFT
RestorePriceArrays( True ); //
```

SEE ALSO SetForeign() function , TimeFrameSet() function

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2004-07-10 07:03:50	TimeFrameRestore and RestorePriceArrays is essentially the same function. So please note that calling RestorePriceArrays also resets the time interval set by eventual previous call to TimeFrameSet

References:

The **RestorePriceArrays** function is used in the following formulas in AFL on-line library:

- BEANS-Summary of Holdings
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- Graphical sector analysis
- Graphical sector stock amalysis
- Heatmap V1
- IBD relative strength database ranker
- Improved NH-NH scan / indicator

- Ranking and sorting stocks
- Ranking Ticker WatchList
- Stress with SuperSmoother

More information:

See updated/extended version on-line.

Reverse
- reverse the order of the elements in the array

Miscellaneous functions
(AmiBroker 5.90)

SYNTAX **Reverse(array, first = 0, last = -1)**

RETURNS

FUNCTION Returns a new array with the order of the elements in specified range reversed. The range is specified by **first** and **last** arguments. If **last** is not specified or negative then AmiBroker will use BarCount - 1.

EXAMPLE

```
Filter = 1;
AddColumn( BarIndex(), "BI");
AddColumn( Reverse( BarIndex() ), "Reversed BI" );
```

SEE ALSO Sort() function

References:

The **Reverse** function is used in the following formulas in AFL on-line library:

- Harmonic Pattern Detection
- Revised Renko chart

More information:

See updated/extended version on-line.

RMI - Relative Momentum Index

Indicators
(AmiBroker 4.10)

SYNTAX	**rmi(*periods* = 20, *momentum* = 5)**
RETURNS	ARRAY
FUNCTION	Calculates Altman's Relative Momentum Index (S&C Feb 1993)
EXAMPLE	rmi(20, 5)
SEE ALSO	

References:

The **RMI** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- Relative Momentum Index (RMI)

More information:

See updated/extended version on-line.

ROC - percentage rate of change

Indicators

SYNTAX	**roc(ARRAY, *periods* = 12, *absmode* = *False*)**
RETURNS	ARRAY
FUNCTION	Calculates the *periods* rate-of-change of ARRAY expressed as percentage. if absmode = False the value returned is 100*(array - ref(array, -periods))/ref(array, -periods) if absmode = True the value returned is 100*(array - ref(array, -periods))/abs(ref(array, -periods))
EXAMPLE	The formula *roc(CLOSE, 14)* returns the 14-period percent rate-of-change of the closing prices.
SEE ALSO	

References:

The **ROC** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Abhimanyu
- AccuTrack
- Adaptave Zones O/B & O/S Oscillator
- Advanced MA system
- Advanced Trend Lines with S & R
- Advisory NRx price chart display.
- AFL Example - Enhanced
- Against all odds
- Alpha and Beta and R_Squared Indicator
- Alphatrend
- Andrews PitchforkV3.3
- Aroon The Advisor
- Auto Trade Step by Step
- Auto-Optimization Framework
- Automatic Trend-line
- Average Price Crossover
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- BMTRIX Intermediate Term Market Trend Indicator
- Bollinger band normalization
- Button trading using AB auto trading interface
- CCT Coppock Curve
- Chaikin's Volatility
- changing period moving avarage
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- Compare Sectors against Tickers
- ConnorsRSI

- Coppock Curve
- Coppock Histogram
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Customised Avg. Profit %, Avg. Loss % etc
- CVR--severe filter
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Dave Landry PullBack Scan
- Day Bar No
- DEBAJ
- Demand Index
- Detailed Equity Curve
- DiNapolis 3x Displaced Moving Averages
- DMI Spread Index
- Double Super Trend System
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- End Of Year Trading
- Envelope System
- Fib Fan Based on ZZ
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- For Auto Trading Setup
- Fre
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Fund Screener
- Futures - Dollar Move Today Indicator
- Gann level plotter
- Gann Swing Charts in 3 modes with text
- Geometric Mean of Volume
- GFX ToolTip
- Guppy Cloud
- Halftrend
- Harmonic Patterns
- Heikin-Ashi(Koma-Ashi) with Moving Average
- HH-LL-PriceBar
- High Low Detection code
- How to add IB Option Symbols
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Inter-market Yield Linear Regression Divergence
- Intraday Fibonacii Trend Break System
- Intraday Strength
- Intraday Trend Break System
- Know Sure Thing
- Linear Candle
- LunarPhase
- MFE and MAE and plot trades as indicator
- Modified Momentum Finder DDT-NB
- Monthly Coppock Guide

- MS Darvas Box with Exploration
- N Line Break Chart
- Nadaraya-Watson Envelope
- New HL Scanner
- nifty
- OBV with Linear Regression
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- pattenz
- Perceptron
- Pivot End Of Day Trading System
- Pivots And Prices
- plot tomorrows pivots on an intraday database
- Polarized Fractal Efficiency
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- prakash
- Price Chart - Fundamental
- Prior Daily OHLC
- PVT Trend Decider
- Range Filter - Trading Strategy
- Rebalancing Backtest avoiding leverage
- Reverse MFI Crossover
- Robert Antony
- SAR-ForNextBarStop
- Schiff Lines
- SectorRSI
- shailu lunia
- Shares To Buy Price Graph
- SIROC Momentum
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Super Trend Indicator
- Super Trend Indicator
- suresh
- swing chart
- TD Moving Average 2
- TD Sequential
- Trend Detection
- Trend Exploration: Count Number of New Highs
- TrendingRibbonArrowsADX
- TSV
- Varexlist
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- Volume - compared with Moving Avg (100%)
- Volume Occilator
- William's Alligator System II
- WILSON RELATIVE PRICE CHANNEL
- Zig Zag Indicator with Valid Entry and Exit Points
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels

- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

Round
- round number to nearest integer

Math functions

SYNTAX	**Round(NUMBER)**
	round(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Rounds NUMBER or ARRAY to the nearest integer.
EXAMPLE	The formula "round(+11.5)" returns +12. The formula "round(-11.4)" returns -11.
SEE ALSO	The ceil() function; the floor() function; the int() function.

References:

The **Round** function is used in the following formulas in AFL on-line library:

- Adaptive Price Channel
- AR_Prediction.afl
- Bad Tick Trim on 5 sec database
- Bullish Percent Index 2 files combined
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI Woodies Style
- Congestions detection
- Continuous Contract Rollover
- Dave Landry PullBack Scan
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- Elder Triple Screen Trading System
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Gfx Toolkit
- Historical Volotility Scan - 50 Day
- Hurst "Like" DE
- Intraday Average Volume
- JEEVAN'S SRI CHAKRA
- Kagi Chart
- Open Range Breakout Trading System
- Ord Volume
- P&F Chart - High/Low prices Sept2003
- pattenz
- PF Chart - Close - April 2004
- Price with Woodies Pivots
- Random Walk
- Square of Nine Roadmap Charts
- Triangle exploration using P&F Chart
- Triangular Moving Average
- Triangular Moving Average new
- ValueChart
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats

- Woodie's Price Panel With Woodie's Pivots

More information:

See updated/extended version on-line.

RSI - relative strength index

Indicators

SYNTAX	**RSI(*periods* = 14)**
	RSIa(*array, periods* = 14)

RETURNS	ARRAY
FUNCTION	Calculates the RSI indicator using *periods* range Second version RSIa accepts input array so it RSI can be applied to other arrays than close.

EXAMPLE	RSI(12)
	RSIa(High, 12);

SEE ALSO	

Comments:

Tomasz Janeczko
tj -- at -- amibroker.com
2007-07-27 09:34:56
	// Internally RSI is implemented as follows
	//
	function BuiltInRSIEquivalent(period)
	{
	P = N = 0;
	
	result = Null;
	
	for(i = 1; i < BarCount; i++)
	{
	diff = C[i] - C[i - 1];
	W = S = 0;
	if(diff > 0) W = diff;
	if(diff < 0) S = -diff;
	
	P = ((period -1) * P + W) / period;
	N = ((period -1) * N + S) / period;
	
	if(i >= period)
	result[i] = 100 * P / (P + N);
	}
	return result;
	}
	
	Plot(BuiltInRSIEquivalent(14), "RSI 1", colorRed);
	Plot(RSI(14), "RSI 2", colorBlue);

References:

The **RSI** function is used in the following formulas in AFL on-line library:

- Adaptave Zones O/B & O/S Oscillator
- Against all odds
- Alphatrend
- BBAreacolor&TGLCROSSNEW

- Bollinger band normalization
- candlestick chart for Volume/RSI/OBV
- CCT StochasticRSI
- colored CCI
- COMBO
- Compare Sectors against Tickers
- Composite Index
- Connors TPS
- ConnorsRSI
- Derivative Oscillator
- Divergence indicator
- DT Oscillator
- Ehlers Laguerre RSI
- Follow the Leader
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Fund Screener
- ICHIMOKU SIGNAL TRADER
- JEEVAN'S SRI CHAKRA
- MCDX (Multi Color Dragon Extended)
- MultiCycle 1.0
- Overbought issues, Oversold issues
- Perceptron
- Ranking and sorting stocks
- Relative Strength Index
- RSI + Avgs
- RSI Double-Bottom
- RSI indicator with Upper & Lower Zone Bars
- RSI Pointer
- RSI styleClipMinMax
- RSI Trendlines and Wedges
- RSIS
- SectorRSI
- Support Resistance levels
- testing multiple system simulataneously
- Varexlist
- Vivek Jain
- Volatility System
- WILSON RELATIVE PRICE CHANNEL

More information:

See updated/extended version on-line.

RWI
- random walk index

Indicators

SYNTAX	**rwi(*minperiods*, *maxperiods*)**
RETURNS	ARRAY
FUNCTION	Calculates the Random Walk Index indicator as a difference between Random Walk Index from Highs (RWIHI() function) and Random Walk Index from Lows (RWILO() function.
EXAMPLE	rwi(9, 40);
SEE ALSO	

References:

The **RWI** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

RWIHi - random walk index of highs

Indicators

SYNTAX	**RWIHi(** ***minperiods*****,** ***maxperiods*** **)**
RETURNS	ARRAY
FUNCTION	Calculates the Random Walk Index from Highs.
EXAMPLE	rwihi(9, 40);
SEE ALSO	

References:

The **RWIHi** function is used in the following formulas in AFL on-line library:

- Random Walk Index

More information:

See updated/extended version on-line.

RWILo
- random walk index of lows

Indicators

SYNTAX	**RWILo(*minperiods*, *maxperiods*)**
RETURNS	ARRAY
FUNCTION	Calculates the Random Walk Index from Lows.
EXAMPLE	rwilo(9, 40);
SEE ALSO	

References:

The **RWILo** function is used in the following formulas in AFL on-line library:

- Random Walk Index

More information:

See updated/extended version on-line.

SafeDivide
- division with divide-by-zero protection

Math functions
(AmiBroker 6.40)

SYNTAX **SafeDivide(x, y, valueifzerodiv)**

RETURNS NUMBER or ARRAY

FUNCTION Safe division that handles division by zero using special handling (replace result with user-defined value)

Parameters:

- *x* - dividend
- *y* - divisor
- *valyeifzerodiv* - the value that is returned by the function if divisor (y) is equal zero

The function returns the value of *x* / *y* (*x* divided by *y*) as long as *y* != 0. If *y* == 0 the value specified in *valyeifzerodiv* argument is returned. Since AmiBroker 6.90 *valyeifzerodiv* might be an array.

EXAMPLE

```
// normal division by (High - Low) can result in division by zero
error
// but this will safely return 0 on days when high == low
result = SafeDivide( Close - Low, High - Low, 0 );
```

SEE ALSO

References:

The **SafeDivide** function is used in the following formulas in AFL on-line library:

- Revised Renko chart
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

SAR
- parabolic stop-and-reverse

Indicators
(AmiBroker 3.30)

SYNTAX	**sar(accel = 0.02, max = 0.2)**
RETURNS	ARRAY
FUNCTION	Calculates Parabolic SAR indicator. Acceleration is given by accel argument and maximum acceleration level is given by max argument
EXAMPLE	sar()
SEE ALSO	

References:

The **SAR** function is used in the following formulas in AFL on-line library:

- BBAreacolor&TGLCROSSNEW
- EKEKO SAR-MF
- Main price chart with Rainbow & SAR
- Parabolic SAR in VBScript
- RI - Auto Trading System
- Trend Analysis_Comentary
- Volatility Breakout with Bollinger Bands

More information:

See updated/extended version on-line.

Say
- speaks provided text

Text-to-Speech functions
(AmiBroker 4.90)

SYNTAX **Say("text", purge = True)**

RETURNS NUMBER

FUNCTION Say() function speaks user-specified text (Windows XP, on lower-end Windows you need to install Microsoft Speech API, voice settings are in Windows Control Panel)

New in 5.20 - 'purge' parameter

when purge is set to True (the default) - any call to Say() purges all previous speak requests (queued or in-progress) and speaks specified text immediatelly.

when purge is set to False - speak request is queued and will be spoken right after previous requests are done.

Also now Say() function returns the NUMERIC value that indicates how many speak requests are pending
0 - ERROR - speech engine not installed or not working properly
1 - currently requested text is spoken now (queue was empty)
2 or more - queue was not empty and previous request(s) will be completed prior to speaking currently specified text

CAVEAT: Creative SoundBlaster X-Fi cards have a driver problems on Windows 7 and Windows Vista causing memory leak in audiodg.exe process (Windows Audio) when using audio output (esp. Say command). To workaround this X-Fi driver problem, do the following:

1. right click the speaker icon in your taskbar
2. select 'Playback Devices'
3. right click the Speakers output
4. select 'Properties'
5. click on the 'Sound Blaster' tab
6. check 'Disable Sound Blaster Enhancements' box

EXAMPLE

```
// simple example
Say("Testing text to speech engine");


// helpful helper functions

function SayOnce( text )
{
   if( StaticVarGetText("lastsaidtext") != text )
   {
      Say( text );
      StaticVarSetText("lastsaidtext", text );
   }
}
```

```
function SayNotTooOften( text, Minperiod )
{
   elapsed=GetPerformanceCounter()/1000;
   Lastelapsed = Nz( StaticVarGet("lastsaytime") );

  if( elapsed - Lastelapsed > Minperiod )
   {
     StaticVarSet("lastsaytime", elapsed );

     Say( text );
   }

}



SayOnce("Testing "+Name() );
SayNotTooOften( "Say not more often than every 60 seconds", 60 );
```

SEE ALSO

References:

The **Say** function is used in the following formulas in AFL on-line library:

- Add Nifty 50 IB Equity Symbol Automatically
- AllinOneAlerts - Module
- AutoTrade using an Exploration
- Button trading using AB auto trading interface
- channel indicator
- For Auto Trading Setup
- How to add IB Option Symbols
- Least Squares Channel Indicator
- Say Notes
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

Second
- get current bar's second

Date/Time
(AmiBroker 40)

SYNTAX	**Second()**
RETURNS	ARRAY
FUNCTION	Retrieves current bar's second
EXAMPLE	Hour()*10000 + Minute() * 100 + Second()
SEE ALSO	Hour(), Minute(), TimeNum()

References:

The **Second** function is used in the following formulas in AFL on-line library:

- Backup Data of 1min Interval
- Export EOD or Intraday to .csv file
- Export Intraday Data
- For Auto Trading Setup
- High Low Detection code
- How to add IB Option Symbols
- New HL Scanner
- White Theme

More information:

See updated/extended version on-line.

SectorID
- get sector ID / name

Information / Categories
(AmiBroker 3.80)

SYNTAX	**SectorID(mode = 0)**
RETURNS	NUMBER/STRING
FUNCTION	Retrieves current stock sector ID/name When mode = 0 (the default value) this function returns numerical sector ID (consecutive sector number) When mode = 1 this function returns name of the sector.

EXAMPLE	Filter = SectorID() == 7 OR SectorID() == 9;
	AddTextColumn(SectorID(1), "Sector name");

SEE ALSO	

References:

The **SectorID** function is used in the following formulas in AFL on-line library:

- Compare Sectors against Tickers
- Graphical sector stock amalysis
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator

More information:

See updated/extended version on-line.

SelectedValue
- retrieves value of the array at currently selected date/time point

Exploration / Indicators
(AmiBroker 4.10)

SYNTAX	**SelectedValue(ARRAY)**
RETURNS	NUMBER
FUNCTION	Retrieves array value at currently selected bar. Main purpose: commentary and interpreration code.
EXAMPLE	selectedvalue(close)
SEE ALSO	

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2003-05-11 06:02:38	SelectedValue(array) is a function that retrieves 'selected element' of the array. Since 'selection line' is available only for CHARTS. SelectedValue gives the value of array at bar that is currently selected in chart by vertical line. This is how it works in INDICATORS, INTERPRETATION and CHART COMMENTARY (because they are relative to selected bar) In AA window 'selected element' means THE LAST BAR of currently selected analysis range. It is the last available bar for "all quotes" and "last n quotes" range. It is the the bar corresponding to "End Date" when using "From-To" range. So if you choose range: "all quotes" in AA SelectedValue function is equivalent to array[BarCount - 1]

References:

The **SelectedValue** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Abhimanyu
- Advanced MA system
- Advanced Trend Lines with S & R
- Advisory NRx price chart display.
- Alphatrend
- Andrews Pitchfork
- Andrews PitchforkV3.3
- Aroon The Advisor
- AR_Prediction.afl
- Auto Trade Step by Step
- Average Price Crossover
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bull/Bear Volume

- Button trading using AB auto trading interface
- Candle Identification
- Candle Stick Demo
- CCI(20) Divergence Indicator
- Chandelier Exit
- changing period moving avarage
- Color Display.afl
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- Congestions detection
- Coppock Trade Signal on Price Chart
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- DateNum_DateStr
- Dave Landry PullBack Scan
- Day Bar No
- DEBAJ
- DiNapolis 3x Displaced Moving Averages
- Double Super Trend System
- ekeko price chart
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Expiry day/days - Last thursday of month
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- For Auto Trading Setup
- Fre
- Frequency distribution of returns
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann level plotter
- Gann Swing Charts in 3 modes with text
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Guppy Cloud
- Halftrend
- Harmonic Patterns
- Heatmap V1
- Heikin-Ashi(Koma-Ashi) with Moving Average
- HH-LL-PriceBar
- High Low Detection code
- How to add IB Option Symbols
- Hurst "Like" DE
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Intraday Fibonacii Trend Break System
- Intraday Strength

- Intraday Trend Break System
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Luna Phase
- LunarPhase
- MAVG
- MS Darvas Box with Exploration
- Multiple sinus noised
- New HL Scanner
- nifty
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- pattenz
- PF Chart - Close - April 2004
- Pivot End Of Day Trading System
- Pivots And Prices
- plot tomorrows pivots on an intraday database
- Point & figure Chart India Securities
- prakash
- Price Chart - Fundamental
- Prior Daily OHLC
- Probability Density & Gaussian Distribution
- PVT Trend Decider
- Range Filter - Trading Strategy
- Robert Antony
- SAR-ForNextBarStop
- Schiff Lines
- shailu lunia
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Super Trend Indicator
- Super Trend Indicator
- suresh
- TD Sequential
- tomy_frenchy
- TrendingRibbonArrowsADX
- Trigonometric Fit - TrigFit with AR for cos / sin
- Vikram's Floor Pivot Intraday System
- Visible Min and Max Value Demo
- visual turtle trading system
- William's Alligator System II
- WILSON RELATIVE PRICE CHANNEL
- Woodie's CCI Panel Full Stats
- Zig Zag Indicator with Valid Entry and Exit Points
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

SendEmail - send an e-mail message

Miscellaneous functions
(AmiBroker 5.90)

SYNTAX	**SendEmail("subject", "message", ShowUI = False)**
RETURNS	NOTHING
FUNCTION	Send an e-mail to an address defined in the Preferences/Alert page. A direct version of functionality already provided by AlertIF. This function sends e-mail unconditionally and it is easier to use if you don't need state logic provided by AlertIf. Note that "From" and "To" addresses as well as SMTP email configuration should be done in Tools->Preferences, "Alerts" page. Without configuring E-mail settings first this function will not work. E-mails are sent asynchronously (so function returns BEFORE e-mail is actually sent, sending occurs in the background) ShowUI parameter decides whenever user interface of e-mailer program is shown after sending e-mail or not.
EXAMPLE	`SendEmail( "This is subject", "Hello world\n\nHere we are sending the body of the email\n\nBest wishes" );`
SEE ALSO	AlertIf() function

References:

The **SendEmail** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

SetBacktestMode - Sets working mode of the backtester

Trading system toolbox
(AmiBroker 50)

SYNTAX **SetBacktestMode(mode)**

RETURNS NOTHING

FUNCTION Sets working mode of the backtester. A 'mode' parameter is one of the following backtest modes:

Supported backtest modes:

- **backtestRegular** - regular, signal-based backtest, redundant signals are removed as shown in this picture
- **backtestRegularRaw** - signal-based backtest, redundant (raw) entry signals are NOT removed, only one position per symbol allowed
- **backtestRegularRawMulti** - signal-based backtest, redundant (raw) entry signals are NOT removed, MULTIPLE positions per symbol will be open if BUY/SHORT signal is true for more than one bar and there are free funds, Sell/Cover exit all open positions on given symbol, Scale-In/Out work on all open positions of given symbol at once.
- **backtestRegularRaw2** - for custom backtester users only, the same as backtestRegularRaw, but redundant exit signals are also kept. AVOID this mode - it requires lots of memory and slows everything down.
- **backtestRegularRaw2Multi** - for custom backtester users only, same as backtestRegularRawMulti, but redundant exit signals are also kept. AVOID this mode - it requires lots of memory and slows everything down.
- **backtestRotational** - rotational trading system see this.

EXAMPLE

```
// default, as in 4.90, regular, signal-based backtest, redundant
signals are removed
SetBacktestMode( backtestRegular );

// signal-based backtest, redundant (raw) signals are NOT removed,
only one position per symbol allowed
SetBacktestMode( backtestRegularRaw );

// signal-based backtest, redundant (raw) signals are NOT removed,
// MULTIPLE positions per symbol will be open if BUY/SHORT signal is
'true' for more than one bar and there are free funds
// Sell/Cover exit all open positions on given symbol, Scale-In/Out
work on all open positions of given symbol at once.
SetBacktestMode( backtestRegularRawMulti );

// rotational trading mode - equivalent of EnableRotationalTrading()
call
SetBacktestMode( backtestRotational );
```

SEE ALSO EnableRotationalTrading() function

References:

The **SetBacktestMode** function is used in the following formulas in AFL on-line library:

- Envelope System
- OBV with Linear Regression
- Periodically ReBalance a BUY & HOLD Portfolio
- Reverse MFI Crossover

More information:

See updated/extended version on-line.

SetBarFillColor - set bar/candlestick/cloud chart fill color

Indicators
(AmiBroker 5.10)

SYNTAX **SetBarFillColor(colorarray)**

RETURNS NOTHING

FUNCTION SetBarFillColor(colorarray) allows to independently control candlestick, bar, cloud, and area chart fill color

SetBarFillColor must PRECEDE the Plot() function call it applies to.

when applied to:

- styleCandle - SetBarFillColor controls the color of interior of candle body, shadows and outline is controlled by color passed in Plot statement
- styleArea - SetBarFillColor controls the color of interior of histogram bars
- styleBar - SetBarFillColor controls the color of the bar (H-L), open and close ticks color is controlled by Plot statement
- styleCloud - SetBarFillColor controls the color of interior of the cloud, outline is controlled by color passed in Plot statement

Other styles are not affected

EXAMPLE

```
SetBarFillColor( IIf( MACD()>Signal(), colorYellow, colorBlue ) );
Plot( C, "Price", IIf( C > O, colorGreen, colorRed ), styleCandle )
```

SEE ALSO PLOT() function

References:

The **SetBarFillColor** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trendlines using multiple timeframes
- BBAreacolor&TGLCROSSNEW
- Caleb Lawrence
- channel indicator
- Channel/S&R and trendlines
- Colorfull Price
- Day Bar No
- Fib Fan Based on ZZ
- For Auto Trading Setup
- Harmonic Pattern Detection
- High Low Detection code
- How to add IB Option Symbols
- Least Squares Channel Indicator

- Linear Candle
- NASDAQ 100 Volume
- New HL Scanner
- Volume Occilator
- Volume Spread for VSA
- White Theme
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

SetBarsRequired - set number of previous and future bars needed for script/DLL to properly execute

Miscellaneous functions
(AmiBroker 4.10)

SYNTAX	**SetBarsRequired(*backwardref* = -1, *forwardref* = -1)**
RETURNS	nothing
FUNCTION	set number of previous and future bars needed for script/DLL to properly execute. If your formula is pure AFL you don't need to use this function at all, as AmiBroker automatically calculates number of bars required for all its built-in functions. But if you are using script or a DLL you may need to use this function to make sure that your indicators are properly calculated in QuickAFL mode. Specifying -1 means no change. For example if you are using the script that calculates 100 bar moving average you may need to call SetBarsRequired(100, 0); at the very beginning of your formula. Please note that in most cases it is not necessary (even if you are using script or DLL) because AmiBroker always provides at least 30 past data bars more than needed. Starting from AmiBroker version 5.20 you can use sbrAll (-2) constant to tell AmiBroker to use ALL available bars:
EXAMPLE	SetBarsRequired(-2, -2); // require ALL past and future bars - this turns OFF quickAFL (v5.20)
SEE ALSO	

References:

The **SetBarsRequired** function is used in the following formulas in AFL on-line library:

- Adaptive Laguerre Filter, from John Ehlers
- AFL Example
- AFL Example - Enhanced
- Alphatrend
- Andrews Pitchfork
- Andrews PitchforkV3.3
- AR_Prediction.afl
- automatic trendlines using fractal patterns
- Backup Data of 1min Interval
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Chandelier Exit
- channel indicator
- Computing Cointegration and ADF Dashboard
- Congestions detection
- Continuous Contract Rollover
- Cybernertic Hilbert Sine Wave
- Darvas Wisestocktrader
- DateNum_DateStr
- Dominant Cycle Phase
- Ehlers Center of Gravity Oscillator
- Ehlers CyberCycle
- Ehlers Dominant Cycle Period
- Ehlers Fisher Transform

- Ehlers Hilbert Transformer Indicator
- Ehlers Instantaneous Trend
- Ehlers Laguerre RSI
- Export EOD or Intraday to .csv file
- Fre
- High Low Detection code
- Hilbert Sine Wave Support & Resistance
- How to add IB Option Symbols
- IBD relative strength database ranker
- interactively test discretionary trading
- Intraday Range and Periods Framer
- JEEVAN'S SRI CHAKRA
- Kagi Chart
- Least Squares Channel Indicator
- MACD BB Indicator
- Monthly bar chart
- Multiple sinus noised
- N Line Break Chart
- New HL Scanner
- Non-repaitning Zigzag line
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- PF Chart - Close - April 2004
- Prashanth
- PVT Trend Decider
- Random Walk
- Renko Chart
- Renko Chart
- Revised Renko chart
- Schiff Lines
- Signal to Noise
- Sine Wave Indicator
- Sony
- Square of Nine Roadmap Charts
- Super Trend Indicator
- Support and resistance
- TAPE READING
- Three Line Break - TLB
- tomy_frenchy
- Triangle exploration using P&F Chart
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- Updated Renko Chart
- Volatility System
- Volume Color with Dynamic Limit
- Voss Predictive Filter (A Peek Into the Future)
- VWAP versus Average Price
- VWAP with standard deviation bands
- White Theme
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

SetChartBkColor - set background color of a chart

Indicators
(AmiBroker 4.80)

SYNTAX	**SetChartBkColor(color)**
RETURNS	NOTHING
FUNCTION	Sets chart background to user-specified color
EXAMPLE	SetChartBkColor(colorBlue);
SEE ALSO	ColorHSB() function , ColorRGB() function

References:

The **SetChartBkColor** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Animated BackGround
- Animated BackGround 1.1
- Auto Trade Step by Step
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- BBAreacolor&TGLCROSSNEW
- Button trading using AB auto trading interface
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- channel indicator
- DEBAJ
- Fib Fan Based on ZZ
- Fibonacci Internal and External Retracements
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Get Moneycontrol News Snippets into Amiboker
- Harmonic Pattern Detection
- Heatmap V1
- Heinkin-Ashi
- Hull Moving Average
- JEEVAN'S SRI CHAKRA
- Least Squares Channel Indicator
- MACD BB Indicator
- Market Facilitation Index VS Volume
- Point & figure Chart India Securities
- Polyfit Lines
- Rebalancing Backtest avoiding leverage
- SUPER PIVOT POINTS
- Support and resistance

- TAPE READING
- Three Day Balance Point
- VSTOP (2)
- VSTOP (3)
- VWAP versus Average Price
- White Theme
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- Woodie's Price Panel With Woodie's Pivots
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

SetChartBkGradientFill - enables background gradient color fill in indicators

Indicators
(AmiBroker 4.90)

SYNTAX **SetChartBkGradientFill(topcolor, bottomcolor, titlebkcolor = default, miny= Null, maxy = Null)**

RETURNS NOTHING

FUNCTION Enables background gradient color fill in indicators.

Please note that this is independent from chart background color (background color fills entire pane, gradient fill is only for actual chart interior, so axes area is not affected by gradient fill)

- topcolor - specifies top color of the gradient fill
- bottomcolor - specifies bottom color of the gradient fill
- titlebkcolor - (optional) the background color of title text. If not specified then top color is automatically used for title background.
- miny, maxy - (optional, new in 5.30) - allows gradient area charts in combination with cloud style (see example 2 below)

EXAMPLE Example 1: basic background gradient

```
SetChartBkGradientFill( ParamColor("BgTop",
colorWhite),ParamColor("BgBottom", colorLightYellow));
```

Example 2: area gradient chart

```
function PlotGradientArea( array, caption, ColorTop, ColorBottom )
{
bkclr = GetChartBkColor();

HH = HighestVisibleValue( array );
if( NOT IsNull( hh ) ) SetChartBkGradientFill( ColorTop,
ColorBottom, bkclr, Null, HH );
Plot( array, Caption, ColorBlend( ColorBottom, colorBlack ) );
PlotOHLC( HH, HH, array, HH, "", bkclr, styleNoLabel | styleNoTitle
| styleCloud, Null, Null, 0, -10 );
}

_SECTION_BEGIN("Price");
SetChartOptions(0,chartShowArrows|chartShowDates);
_N(Title = StrFormat("{{NAME}} - {{INTERVAL}} {{DATE}} Open %g, Hi
%g, Lo %g, Close %g (%.1f%%) {{VALUES}}", O, H, L, C, SelectedValue(
ROC( C, 1 ) ) ));
PlotGradientArea( C, "Close", ParamColor("Top", colorLightOrange),
ParamColor("Bottom", colorPaleGreen ) );

_SECTION_END();
```

SEE ALSO PLOT() function , PLOTFOREIGN() function , PLOTGRID() function , PLOTOHLC() function , PLOTSHAPES() function , PlotText() function , PLOTVAPOVERLAY() function

References:

The **SetChartBkGradientFill** function is used in the following formulas in AFL on-line library:

- ALJEHANI
- Average Price Crossover
- BBAreacolor&TGLCROSSNEW
- CoinToss ver 1
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- DEBAJ
- Fib Fan Based on ZZ
- Fibonacci Internal and External Retracements
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Heinkin-Ashi
- JEEVAN'S SRI CHAKRA
- Jesse Livermore Secret Market Key
- Linear Candle
- Lunar Phases - original
- LunarPhase
- Plot visual stop / target ratio.
- Probability Density & Gaussian Distribution
- RSI + Avgs
- RSI indicator with Upper & Lower Zone Bars
- Square of Nine Roadmap Charts
- TD Sequential
- Three Day Balance Point
- TRENDAdvisor
- TSV
- Volume Occilator
- White Theme
- WILSON RELATIVE PRICE CHANNEL

More information:

See updated/extended version on-line.

SetChartOptions - set/clear/overwrite defaults for chart pane options

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **SetChartOptions(Mode = 0, Flags = 0, gridFlags = chartGridMiddle, ymin = 0, ymax = 0, blankbars = 0)**

RETURNS NOTHING

FUNCTION Allows to set/clear/overwrite/set defaults for chart pane options

- Mode - specifies how options are set:
  - 0 - set only the DEFAULT values for new chart. Defaults are applied only once when chart is inserted in a new pane, so later you can modify any option using Indicator Builder
  - 1 - overwrite - the values specified in 2nd and 3rd argument overwrite any previously set values
  - 2 - set flag - flags specified in 2nd and 3rd parameter are binary-ORed with the current values, so effectively these options are set while remaining are unchanged
  - 3 - reset flag - flags specified in 2nd and 3rd parameter are cleared while the others remain unchanged.
- Flags - allowable flags are:
  chartShowDates, chartLogarithmic, chartShowArrows, chartWrapTitle (4.75 or higher), chartHideQuoteMarker (v5.06). chartHideQuoteMarker - hides the quote selector line on per-pane basis, the same as Parameter dialog -> Axes & Grid -> Vert. quote marker: Show/Hide, chartDisableYAxisCursor (new in 5.80) - disables changing mouse pointer to up/down arrow when hovering above Y axis, chartDisableTooltips (new in 5.80) - disables displaying tooltips (data tips).
- gridFlags - (for internal AmiBroker use - do not use it in your own coding as this parameter will be eventually removed) allowable values are: chartGridDiv100, chartGridPercent, chartGridDiv1000, chartGridMargins chartGridMiddle, chartGrid0, chartGrid30, chartGrid70, chartGrid10, chartGrid90, chartGrid50,chartGrid100,chartGrid20,chartGrid80,chartGrid1
- ymin, ymax - (new in 5.07) these parameters specify Y-axis minimum and maximum values for custom scaling. If you specify any values that meet the condition ymin < ymax, AmiBroker will turn OFF automatic scaling and use specified min/max values for Y scale. Note that Mode argument controls when these settings are applied (0 - only when new chart is created, 1 - always), when modes 2 and 3 are used - scaling is not changed.
- blankbars - (new in 5.30) defines the minimum number of blank bars for given chart. The default value of zero means no change (use preferences setting).
  If specified value is less than value set in preferences, it is ignored, so you can not decrease the blank bars below value set in preferences.
  If many panes within same chart use this function, then the largest specified blankbars will be used

  Note that you can still extend blank space further using END key. Special feature - if "blankbars" is negative then extra blank bars added are equal to absolute value of blankbars parameter plus chart gets scrolled to rightmost position.

Caveat: forcing custom blankbars via SetChartOptions effectivelly disables HOME key scroll to begin operation.

EXAMPLE

```
//to mark "Show arrows" by default in a new chart use
SetChartOptions( 0, chartShowArrows );
```

Example 2 (works only with version 4.75 or higher):

```
SetChartOptions(2, chartWrapTitle );
Title="this is a test of automatic wrapping of title text that is
too long to fit in single line, for that reason this sample formula
uses very long text. I hope you are enjoying the sample";
```

SEE ALSO

References:

The **SetChartOptions** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Advanced MA system
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Auto Fib Ext&Retracement
- Auto Trade Step by Step
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Button trading using AB auto trading interface
- Caleb Lawrence
- changing period moving avarage
- channel indicator
- Channel/S&R and trendlines
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- Computing Cointegration and ADF Dashboard
- Coppock Trade Signal on Price Chart
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Dave Landry PullBack Scan
- Day Bar No
- DEBAJ

- Dinapoli Perferred Stochastic
- Double Super Trend System
- Elder Triple Screen Trading System
- Fib Fan Based on ZZ
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- For Auto Trading Setup
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann level plotter
- Gann Swing Charts in 3 modes with text
- Get Moneycontrol News Snippets into Amiboker
- Halftrend
- Harmonic Pattern Detection
- Heikin Ashi System
- Heikin-Ashi(Koma-Ashi) with Moving Average
- HH-LL-PriceBar
- High Low Detection code
- How to add IB Option Symbols
- IB Backfiller
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Intraday Fibonacii Trend Break System
- Intraday Strength
- Intraday Trend Break System
- Kairi Relative Index
- Last Five Trades Result Dashboard – AFL code
- Least Squares Channel Indicator
- Linear Candle
- Lunar Phases - original
- LunarPhase
- MACD BB Indicator
- Market Profile
- MS Darvas Box with Exploration
- Multi-color Volume At Price (VAP)
- N Line Break Chart
- New HL Scanner
- nifty
- Non-repaitning Zigzag line
- Open Range Breakout Trading System
- pattenz
- Pivots And Prices
- plot tomorrows pivots on an intraday database
- Polyfit Lines
- prakash
- Price Chart - Fundamental
- Prior Daily OHLC
- Range Filter - Trading Strategy
- Rebalancing Backtest avoiding leverage
- RSI styleClipMinMax
- shailu lunia

- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Super Trend Indicator
- Super Trend Indicator
- Support and resistance
- suresh
- TrendingRibbonArrowsADX
- visual turtle trading system
- VSTOP (2)
- VSTOP (3)
- VWAP versus Average Price
- White Theme
- William's Alligator System II
- Woodie's CCI Panel Full Stats
- ZigZag filter rewrited from scratch in AFL
- ZigZag Hi Lo Barcount
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

SetCustomBacktestProc - define custom backtest procedure formula file

Trading system toolbox
(AmiBroker 4.70)

SYNTAX **SetCustomBacktestProc(filename, enable = True)**

RETURNS NOTHING

FUNCTION This function allows changing custom backtest procedure file from AFL formula level.

To learn more about custom backtester procedures please read this document.

Parameters

- *filename* parameter instructs backtester to use external formula file as custom backtest procedure, if empty - it means use current formula
- *enable* = True (default) - enables custom backtesting procedure (the same as SetOption("UseCustomBacktestProc", True);
  enable = False - disables custom proc

EXAMPLE

```
SetCustomBacktestProc( "Formulas\MyCustomBacktest.afl", True );
```

SEE ALSO

References:

The **SetCustomBacktestProc** function is used in the following formulas in AFL on-line library:

- Customised Avg. Profit %, Avg. Loss % etc
- Detailed Equity Curve
- MFE and MAE and plot trades as indicator
- Rebalancing Backtest avoiding leverage

More information:

See updated/extended version on-line.

SetForeign - replace current price arrays with those of foreign security

Referencing other symbol data
(AmiBroker 4.50)

SYNTAX **SetForeign(*ticker, fixup = True, tradeprices = False*)**

RETURNS NUMBER

FUNCTION The **SetForeign** function replaces current price/volume arrays with those of foreign security, returns True (1) if ticker exists, False (0) otherwise.

If ticker does not exist (and function returns false) price arrays are not changed at all.

fixup parameter controls if data holes are filled with previous bar data or not.

- 0 - the holes are not fixed
- 1 - default value - missing data bar OHLC fields are all filled using previous bar Close and volume is set to zero.
- 2 - (old pre-4.90 behaviour) - causes filling the holes in the data with previous O, H, L, C, V values

tradeprices parameter controls if trade price arrays should be replaced too. If it is set to TRUE, then not only OHLC, V, OI, Avg arrays are set to foreign symbol values, but also BuyPrice, SellPrice, ShortPrice, CoverPrice, PointValue, TickSize, RoundLotSize, MarginDeposit variables are set to correspond to foreign security. This allows Equity() to work well with SetForeign.

Single **SetForeign("ticker")** call is equivalent to the following sequence:

```
C = Foreign( "ticker", "C" );
O = Foreign( "ticker", "O" );
H = Foreign( "ticker", "H" );
L = Foreign( "ticker", "L" );
V = Foreign( "ticker", "V" );
OI = Foreign( "ticker", "I" );
Avg = ( C + H + L )/3;
```

but 6x faster (SetForeign takes about the same time as single foreign). To restore original prices call **RestorePriceArrays()**

EXAMPLE

```
// Example 1: Plot the indicator using foreign security data
SetForeign("MSFT");
Plot( Ultimate(), "Ultimate from MSFT", colorRed );
RestorePriceArrays();

// Example 2: Use SetForeign with Equity function
SetForeign("MSFT", True, True );
Buy = Cross( MACD(), Signal());
Sell = Cross( Signal(), MACD());
e = Equity(); // backtest on MSFT
RestorePriceArrays( True ); //
```

SEE ALSO FOREIGN() function , RestorePriceArrays() function

References:

The **SetForeign** function is used in the following formulas in AFL on-line library:

- Bad Tick Trim on 5 sec database
- BEANS-Summary of Holdings
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- Graphical sector analysis
- Graphical sector stock amalysis
- Heatmap V1
- IBD relative strength database ranker
- Improved NH-NH scan / indicator
- Ranking and sorting stocks
- Ranking Ticker WatchList
- Stress with SuperSmoother
- WLBuildProcess

More information:

See updated/extended version on-line.

SetFormulaName
- set the name of the formula

Trading system toolbox
(AmiBroker 4.50)

SYNTAX **SetFormulaName(string)**

RETURNS NOTHING

FUNCTION Allows to programatically change the name of the formula that is displayed in the backtest result explorer.

EXAMPLE
```
SetFormulaName("My Holy Grail System");
```

SEE ALSO

References:

The **SetFormulaName** function is used in the following formulas in AFL on-line library:

- AFL Example
- AFL Example - Enhanced
- High Low Detection code
- How to add IB Option Symbols
- New HL Scanner
- Periodically ReBalance a BUY & HOLD Portfolio
- Triangular Moving Average new

More information:

See updated/extended version on-line.

SetGradientFill
- set the colors of a gradient fill plot

Indicators
(AmiBroker 5.60)

SYNTAX **SetGradientFill(topcolor, bottomcolor, baseline = Null, baselinecolor = -1)**

RETURNS NOTHING

FUNCTION The function defines colors for gradient area charts. Gradient chart is obtained using styleGradient in the Plot() function call. Upper gradient color is specified by topcolor, bottom gradient color is specified by botttomcolor. Optional parameters (baseline/baselinecolor) allow reverse gradient chart (such as underwater equity) and 3 color gradients top->baseline->bottom. See code for Underwater Equity for example usage of reverse gradient chart (with baseline at the top). Baseline parameter specifies the Y-axis position of chart baseline. The baselinecolor parameter specifies the color of gradient that is to be used at that level. If baselinecolor is not specified, then only 2-color gradient is plotted (topcolor->bottomcolor).

EXAMPLE

```
// Underwater Equity chart
// (C)2009 AmiBroker.com
// Should be used only on ~~~EQUITY or ~~~OSEQUITY symbol


EQ = C;
MaxEQ = Highest( EQ );
DD = 100 * ( Eq - MaxEQ ) / MaxEq;
MaxDD = Lowest( DD );

Title = StrFormat("Drawdown = %.2g%%, Max. drawdown %.2g%%", DD,
LastValue( MaxDD ) );

SetGradientFill( GetChartBkColor(), colorBlue, 0 );

Plot( DD, "Drawdown ", colorBlue, styleGradient | styleLine );

Plot( MaxDD, "Max DD", colorRed, styleNoLabel );

SetChartOptions( 2, 0, chartGridPercent );

if( Name() != "~~~EQUITY" AND Name() != "~~~OSEQUITY"  ) Title =
"Warning: wrong ticker! This chart should be used on ~~~EQUITY or
~~~OSEQUITY only";
```

SEE ALSO Plot() function

References:

The **SetGradientFill** function is used in the following formulas in AFL on-line library:

- B-Xtrender
- Stress with SuperSmoother
- White Theme

More information:

See updated/extended version on-line.

SetOption - sets options in automatic analysis settings

Trading system toolbox
(AmiBroker 4.30)

SYNTAX **SetOption(field, value)**

RETURNS NOTHING

FUNCTION Sets various options in automatic analysis settings. Affects also Equity() function results.

field - is a string that defines the option to change. There are following options available:

- "NoDefaultColumns" - if set to True - exploration does not have default Ticker and Date/Time columns
- "InitialEquity"
- "AllowSameBarExit"
- "ActivateStopsImmediately"
- "AllowPositionShrinking"
- "FuturesMode"
- "InterestRate"
- "MaxOpenPositions" - maximum number of simlutaneously open positions (trades) in portfolio backtest/optimization
- "WorstRankHeld" - the worst rank of symbol to be held in rotational trading mode (see **EnableRotationalTrading** for more details)
- "MinShares" - the minimum number of shares required to open the position in the backtester/optimizer. If you don't have enough funds to purchase that many, trade will NOT be entered
- "MinPosValue" - (4.70.3 and above) the minimum dollar amount required to open the position in the backtester/optimizer. If you don't have enough funds trade will NOT be entered
- "PriceBoundChecking" - if set to False - disables checking and adjusting buyprice/sellprice/coverprice/shortprice arrays to current symbol High-Low range.
- CommissionMode -
  0 - use portfolio manager commission table
  1 - percent of trade
  2 - $ per trade
  3 - $ per share/contract
- CommissionAmount - amount of commission in modes 1..3
- AccountMargin (in old versios it was 'MarginRequirement') - account margin requirement (as in settings), 100 = no margin
- ReverseSignalForcesExit - reverse entry signal forces exit of existing trade (default = True)
- UsePrevBarEquityForPosSizing - Affects how percent of current equity position sizing is performed.
  False (default value) means: use current (intraday) equity to perform position sizing,
  True means: use previous bar closing equity to perform position sizing
- PortfolioReportMode - sets backtester report mode:
  0 - trade list
  1 - detailed log
  2 - summary
  3 - no output (custom only)

- UseCustomBacktestProc - True/False - allows to turn on/off custom backtest procedure
- EveryBarNullCheck - allows to turn on checking for Nulls in arithmetic operations on every bar in the array(by default it is OFF - i.e. AmiBroker checks for nulls that appear in the beginning of the arrayand in the end of the array and once non-null value is detected it assumes no further holes (nulls) in the middle). Turning "EveryBarNullCheck" to True allows to extend these checks to each and every barwhich is the way 4.74.x and earlier versions worked.
  Note however that turning it on gives huge performance penalty (arithmetic operations are performed even 4x slower when this option is ON, so don't use it unless you really have to).
- HoldMinBars - Number - if set to value > 0 - it disables exit during user-specified number of bars even if signals/stops are generated during that period
- EarlyExitBars - Number if set to value > 0 - causes that special early exit (redemption) fee is charged if trade is exited during this period
- EarlyExitFee - defines the % (percent) value of early exit fee
- HoldMinDays - Number - if set to value > 0 - it disables exit during user-specified number of CALENDAR DAYS (not bars) even if signals/stops are generated during that period
- EarlyExitDays - Number if set to value > 0 - causes that special early exit (redemption) fee is charged if trade is exited during the period specified in calendar days (not bars).
- DisableRuinStop - it set to TRUE built-in ruin stop is disabled
- Generate report - allows to suppress/force generation of backtest report. Allowable values: 0, 1, or 2
  By default backtest reports are generated ONLY for portfolio backtests and for individual backtests if individual reporting is turned on in the settings. Reports are disabled for optimization.
  Now with the SetOption() function you can either supress report generation for backtests or enable report generation during certain optimization steps, all from code level.
  SetOption("GenerateReport", 0); // suppress generation of report
  SetOption("GenerateReport", 1); // force generation of full report
  SetOption("GenerateReport", 2); // only one-line report is generated (in results.rlst file) viewable as single line in Report Explorer
- SeparateLongShortRank - True/False
  When separate long/short ranking is enabled, the backtester maintains TWO separate "top-ranked" signal lists, one for long signals and one for short signals. This ensures that long and short candidates are independently even if position score is not symetrical (for example when long candidates have very high positive scores while short candidates have only fractional negative scores). That contrasts with the default mode where only absolute value of position score matters, therefore one side (long/short) may completely dominate ranking if score values are asymetrical.
  When SeparateLongShortRank is enabled, in the second phase of backtest, two separate ranking lists are interleaved to form final signal list by first taking top ranked long, then top ranked short, then 2nd top ranked long, then 2nd top ranked short, then 3rd top ranked long and 3rd top ranked short, and so on... (as long as signals exist in BOTH long/short lists, if there is no more signals of given kind, then remaining signals from either long or short lists are appended)
  For example: Entry signals(score):ESRX=Buy(60.93), GILD=Short(-47.56), CELG=Buy(57.68), MRVL=Short(-10.75), ADBE=Buy(34.75), VRTX=Buy(15.55),

SIRI=Buy(2.79),
As you can see Short signals get interleaved between Long signals even though their absolute values of scores are smaller than corresponding scores of long signals. Also there were only 2 short signals for that particular bar so, the rest of the list shows long signals in order of position score
Although this feature can be used independently, it is intended to be used in combination with MaxOpenLong and MaxOpenShort options.

- MaxOpenLong - limits the number of LONG positions that can be open simultaneously
- MaxOpenShort - limits the number of SHORT positions that can be open simultaneously
  The value of ZERO (default) means NO LIMIT. If both MaxOpenLong and MaxOpenShort are set to zero (or not defined at all) the backtester works old way - there is only global limit active (MaxOpenPositions) regardless of type of trade.
  Note that these limits are independent from global limit (MaxOpenPositions). This means that MaxOpenLong + MaxOpenShort may or may not be equal to MaxOpenPositions.
  If MaxOpenLong + MaxOpenShort is greater than MaxOpenPositions then total number of positions allowed will not exceed MaxOpenPositions, and individual long/short limits will apply too. For example if your system MaxOpenLong is set to 7 and maxOpenShort is set to 7 and MaxOpenPositions is set to 10 and your system generated 20 signals: 9 long (highest ranked) and 11 short, it will open 7 long and 3 shorts.
  If MaxOpenLong + MaxOpenShort is smaller than MaxOpenPositions (but greater than zero), the system won't be able to open more than (MaxOpenLong+MaxOpenShort).
  Please also note that MaxOpenLong and MaxOpenShort only cap the number of open positions of given type (long/short). They do NOT affect the way ranking is made. I.e. by default ranking is performed using ABSOLUTE value of positionscore. If your position score is NOT symetrical, this may mean that you are not getting desired top-ranked signals from one side. Therefore, to fully utilise MaxOpenLong and MaxOpenShort in rotational balanced ("market neutral") long/short systems it is desired to perform SEPARATE ranking for long signals and short signals.
  To enable separate long/short ranking use:
  SetOption("SeparateLongShortRank", True);
- RefreshWhenCompleted - when set to TRUE, it will perform View->Refresh All after Automatic-Analysis operation (scan/exploration/backtest/optimize) is completed.
- RequireDeclarations - when set to TRUE the AFL engine will always require variable declarations (using local/global) on formula-by-formula basis
- ExtraColumnsLocation - allows the user to change the location of custom columns added during backtest/optimization.
  "extra" columns mean:
  a) any custom metrics added using custom backtester
  b) any optimization parameters defined using Optimize() function

  If both custom metrics and optimization parameters are present then custom metrics appear first then optimization parameters

  This function is provided to allow the user to change the default "at the end" location of custom columns/optimization parameters.
  For example:

SetOption("ExtraColumnsLocation", 1);

will cause that custom metrics and opt params will be subsequently added starting from column 1 (as opposed to last column default)

Note that this setting changes "visual" order of columns, not really in-memory order or export order, so exported data files or copy/paste format do not change.

- SettlementDelay - this option describes the number of days (not bars) it takes for sale proceeds to settle and be available for opening new positions.

  SetOption("SettlementDelay", 3); // this will cause that proceeds from sale are only available for trading on 3rd day after sale

  For detailed tracking " Detailed log" report option now shows available and unsettled funds for T+1, T+2 and so on

  Note: when using this option it is recommended to use backtestRegularRaw instead of backtestRegular, otherwise some trades may not be entered because funds are not settled immediately and you need to be able to enter not on first but subsequent buy signals and that is exactly what backtestRegularRaw offers.

  Note2: old backtester (Equity() function) ignores settlement delay
- StaticVarAutoSave - allow periodical auto-saving of persistent static variables (in addition to saving on exit, which is always done).

  The interval is given in seconds. In AmiBroker 6.90 and higher you can also pass -1 as interval for one-shot saving of static variables.
  For example:
  SetOption("StaticVarAutoSave", 60); // auto-save persistent variables every 60 seconds (1-minute)
  It is important to understand that persistent variables are saved ON EXIT automatically, without any user intervention so it should be enough for most cases. If you for some reason want auto-saves when AmiBroker is running, then you can use this function. Please note that writing many static variables into physical disk file takes time and it blocks all static variable access so you should AVOID specifying too small auto-save intervals. Saving every second is bad idea - it will cause overload. Saving every 60 seconds should be fine. Calling function with interval set to zero disables auto-save.
  SetOption("StaticVarAutoSave", 0);
- MCEnable - controls Monte Carlo simulation: 0 - disabled, 1 - enabled in backtests, 2 - enabled in backtests and optimizations
- MCRuns - number of Monte Carlo simulation runs (realizations) default 1000
- MCPosSizeMethod - Monte Carlo position size method: 0 - don't change, 1 - fixed size, 2 - constant amount, 3 - percent of equity
- MCPosSizeShares - number of shares per trade in MC simulation
- MCPosSizeValue - dollar value per trade in MC simulation
- MCPosSizePctEquity - percent of current equity per trade in MC simulation
- MCChartEquityCurves - true/false (1/0) - enables Monte Carlo equity chart
- MCStrawBroomLines - defines number of equity lines drawn in Monte Carlo straw broom chart

- WarningLevel - allows to change warning level. Level 1 is default for all AFL executions with exception of AFL editor and commentary where warning level is set to 4
  Warning Level
  1 - report only level 1 warnings (502- too much plots)
  2 - report level 1 and 2 warnings (above plus assignment within conditional, division by zero, threadsleep period too long)
  3 - report level 1, 2 and 3 warnings (above plus createobject/createstaticobject)
  4- report all warnings (default for the AFL editor)

WARNING: If you change the option on *per-symbol* basis the composite results (%profit for example) will be DISTORTED since calculations assume that OPTIONS are constant for all symbols in one backtest run. 'HoldMinBars', 'EarlyExit..." options are exception from this rule (i.e. can be safely set on per-symbol basis)

EXAMPLE

```
SetOption("InitialEquity", 5000 );
SetOption("AllowPositionShrinking", True );
SetOption("MaxOpenPositions", 5 );
PositionSize = -100/5;
```

SEE ALSO EnableRotationalTrading() function , EQUITY() function

References:

The **SetOption** function is used in the following formulas in AFL on-line library:

- AFL Example
- AFL Example - Enhanced
- Black Scholes Option Pricing
- CoinToss ver 1
- Connors TPS
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- End Of Year Trading
- Envelope System
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- IBD relative strength database ranker
- Kelly criterion
- Last Five Trades Result Dashboard – AFL code
- MFE and MAE and plot trades as indicator
- New HL Scanner
- OBV with Linear Regression
- Perceptron
- Periodically ReBalance a BUY & HOLD Portfolio
- Plot the Equity Curve without Backtesting?
- PVT Trend Decider
- Ranking and sorting stocks
- Rebalancing Backtest avoiding leverage
- Relative Strength
- Reverse MFI Crossover
- RUTVOL timing signal with BB Scoring routine
- Scale Out: Futures
- Simple Candle Exploration

- Sine Wave Indicator
- Stan Weinstein strategy
- suresh
- testing multiple system simulataneously
- Trend Exploration: Count Number of New Highs
- Trend Exploration: Slope Moving Average
- Trend Following System
- Triangular Moving Average new
- Visualization of stoploses and profit in chart
- Volatility System

More information:

See updated/extended version on-line.

SetPositionSize - set trade size

Trading system toolbox
(AmiBroker 4.70)

SYNTAX **SetPositionSize(size, method)**

RETURNS ARRAY

FUNCTION This function allows to control trade (position) size in four different ways, depending on 'method' parameter.

Parameters:

size (ARRAY) defines desired trade size

method (ARRAY) defines how 'size' is interpreted

- spsValue (=1) - dollar value of size (as in previous versions)
- spsPercentOfEquity (=2) - size expressed as percent of portfolio-level equity (size must be from ..100 (for regular accounts) or .1000 for margin accounts)
- spsShares (=4) - size expressed in shares/contracts (size must be > 0)
- spsPercentOfPosition (=3) - size expressed as percent of currently open position (for SCALING IN and SCALING OUT ONLY)
- spsNoChange (=0) - don't change previously set size for given bar

New SetPositionSize function automatically encodes new methods of expressing position size into old "positionsize" variable as follows:

- values below -2000 encode share count,
- values between -2000 and -1000 encode % of current position
- values between -1000 and 0 encode % of portfolio equity
- values above 0 encode dollar value

Although it is possible to assign these values directly to old-style PositionSize variable, new code should use SetPositionSize function for clarity.

EXAMPLE For example to liquidate 50% of position simply use

```
SetPositionSize( 50, spsPercentOfPosition * ( Buy == sigScaleOut )
);
```

Special value spsNoChange (=0) means don't change previously set size for given bar (allows to write constructs like that):

```
SetPositionSize( 100, spsShares ); // 100 shares by default
SetPositionSize( 50, IIf( Buy == sigScaleOut, spsPercentOfPosition,
spsNoChange ) ); // for scale-out use 50% of current position size
```

Example of code that exits 50% on first profit target, 50% on next profit target and everything at trailing stop:

```
Buy = Cross( MA( C, 10 ), MA( C, 50 ) );
```

```
Sell = 0;

// the system will exit
// 50% of position if FIRST PROFIT TARGET stop is hit
// 50% of position is SECOND PROFIT TARGET stop is hit
// 100% of position if TRAILING STOP is hit

FirstProfitTarget = 10; // profit
SecondProfitTarget = 20; // in percent
TrailingStop = 10; // also in percent

priceatbuy=0;
highsincebuy = 0;

exit = 0;

for( i = 0; i < BarCount; i++ )
{
   if( priceatbuy == 0 AND Buy[ i ] )
     {
       priceatbuy = BuyPrice[ i ];
     }

   if( priceatbuy > 0 )
     {
       highsincebuy = Max( High[ i ], highsincebuy );

      if( exit == 0 AND
          High[ i ] >= ( 1 + FirstProfitTarget * 0.01 ) * priceatbuy
)
       {
         // first profit target hit - scale-out
         exit = 1;
         Buy[ i ] = sigScaleOut;
       }

      if( exit == 1 AND
          High[ i ] >= ( 1 + SecondProfitTarget * 0.01 ) *
priceatbuy )
       {
         // second profit target hit - exit
         exit = 2;
         SellPrice[ i ] = Max( Open[ i ], ( 1 + SecondProfitTarget *
0.01 ) * priceatbuy );
       }

      if( Low[ i ] <= ( 1 - TrailingStop * 0.01 ) * highsincebuy )
       {
         // trailing stop hit - exit
         exit = 3;
         SellPrice[ i ] = Min( Open[ i ], ( 1 - TrailingStop * 0.01
) * highsincebuy );
       }

      if( exit >= 2 )
       {
         Buy[ i ] = 0;
         Sell[ i ] = exit + 1; // mark appropriate exit code
         exit = 0;
         priceatbuy = 0; // reset price
         highsincebuy = 0;
       }
     }
}

SetPositionSize( 50, spsPercentOfEquity );
SetPositionSize( 50, spsPercentOfPosition * ( Buy == sigScaleOut )
); // scale out 50% of position
```

SEE ALSO

References:

The **SetPositionSize** function is used in the following formulas in AFL on-line library:

- Connors TPS
- Customised Avg. Profit %, Avg. Loss % etc
- Ed Seykota's TSP: EMA Crossover System
- For Auto Trading Setup
- Halftrend
- Last Five Trades Result Dashboard – AFL code
- Open Range Breakout Trading System
- Plot the Equity Curve without Backtesting?
- Range Filter - Trading Strategy
- Scale Out: Futures
- Stan Weinstein strategy
- testing multiple system simulataneously
- Trend Following System
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

SetSortColumns
- sets the columns which will be used for sorting in AA window

Exploration / Indicators
(AmiBroker 4.90)

SYNTAX **SetSortColumns(col1, col2, ...)**

RETURNS NOTHING

FUNCTION sets the columns which will be used for sorting. col1, col2, ... col10 -Column numbers are ONE-based. Positive number means sort ASCENDING, negative number means sort DESCENDING. Upto 10 columns can be specified for multiple-column sort. Each subsequent call to SetSortColumns overwrites previous one, but multiple SetSortColumns make sense if you want to add multiple rankings by different columns via AddRankColumn

EXAMPLE

```
// sort by 5th column in ascending order
SetSortColumns( 5 )

// sort by 3rd column in descending order
SetSortColumns( -3 )

// sort by 1st column in ascending order AND then by Second column
in descending order (multiple-column sort).
SetSortColumns( 1, -2 );
```

SEE ALSO AddRankColumn() function

References:

The **SetSortColumns** function is used in the following formulas in AFL on-line library:

- Commodity Selection Index (CSI)
- New HL Scanner
- Open Range Breakout Trading System
- Stress with SuperSmoother
- Trend Exploration: Count Number of New Highs

More information:

See updated/extended version on-line.

SetStopPrecedence
- set precedence of built-in stops

Trading system toolbox
(AmiBroker 60)

SYNTAX	**SetStopPrecedence(type1, type2, type3, type4)**
RETURNS	NOTHING
FUNCTION	SetStopPrecedence defines the order in which stops are executed in case many stops trigger on the very same bar, as in example given below. SetStopPrecedence should be called AFTER ApplyStop() functions.
EXAMPLE	see below
SEE ALSO	ApplyStop() function

```
// first execute N-bar stop, then max loss, then trailing, then
profit

SetStopPrecedence( stopTypeNBar, stopTypeLoss, stopTypeTrailing,
stopTypeProfit );
```

References:

The **SetStopPrecedence** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

SetTradeDelays
- allows to control trade delays applied by the backtester

Trading system toolbox
(AmiBroker 4.10)

SYNTAX **SetTradeDelays(*buydelay, selldelay, shortdelay, coverdelay*)**

RETURNS nothing

FUNCTION Sets trade delays applied by the backtester. This function allows you to override trade delays from the "Settings" page. It is important do understand what trade delays really do. They in fact internally apply the following:

```
Buy = Ref( Buy, -buydelay );
Sell = Ref( Sell, -selldelay );
Short = Ref( Short, -shortdelay );
Cover = Ref( Cover, -coverdelay );
```

inside backtester after your formula is executed but before backtester starts trade simulation. It is functionally equivalent to having above 4 lines at the end of your formula. Note that NO OTHER variables are affected by trade delays, therefore for example if your position sizing depends on values found in buy/sell/short/cover variables *and* if you are using non-zero trade delays you need to account for that in your code.

EXAMPLE settradedelays(1, 1, 1, 1)

SEE ALSO

References:

The **SetTradeDelays** function is used in the following formulas in AFL on-line library:

- AFL Example
- AFL Example - Enhanced
- CoinToss ver 1
- Connors TPS
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- Envelope System
- Evaluating Candle Patterns in a trading system
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Last Five Trades Result Dashboard – AFL code
- OBV with Linear Regression
- Perceptron
- PVT Trend Decider
- Rapid Prototyping Method for System Development
- Rebalancing Backtest avoiding leverage
- Reverse MFI Crossover
- RUTVOL timing signal with BB Scoring routine
- Scale Out: Futures
- Stan Weinstein strategy
- Trend Following System
- Triangular Moving Average new
- Volatility System
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

ShellExecute
- execute a file

Basic price pattern detection
(AmiBroker 5.40)

SYNTAX **ShellExecute("filepath", "arguments", "parameters", showcmd = 1)**

RETURNS NUMBER

FUNCTION The function opens a file or runs executable.

It is equivalent of Windows API ShellExecute, with one difference, it always uses "open" verb. This allows running executables, scripts, opening document files using their associated editors, etc.

If the function succeeds, it returns a value greater than 32. If the function fails, it returns an error value that indicates the cause of the failure.

For possible error codes, consult Microsoft documentation:
http://msdn.microsoft.com/en-us/library/bb762153(VS.85).aspx

In AmiBroker version 6.20 this function provides special feature that allows to run batches. If you use "runbatch" in first parameter you can specify the batch (.abb) file in 2nd argument and it will start a batch.

EXAMPLE

```
// run notepad
ShellExecute("notepad.exe", "", "" );

// run batch
ShellExecute("runbatch", "path_to_batch_file.abb", "" );
```

SEE ALSO

References:

The **ShellExecute** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

sign
- returns the sign of the number/array

Math functions
(AmiBroker 4.50)

SYNTAX	**sign(x)**
RETURNS	ARRAY or NUMBER
FUNCTION	Sign function returns 1 if x value is greater than zero, -1 if the x is less than zero and 0 if x equals zero. x can be a number or array.
EXAMPLE	
SEE ALSO	

References:

The **sign** function is used in the following formulas in AFL on-line library:

- Cybernertic Hilbert Sine Wave
- Elder Triple Screen Trading System
- elliott wave manual labelling
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- John Ehler
- Sine Wave Indicator

More information:

See updated/extended version on-line.

Signal
- macd signal line

Indicators

SYNTAX	**Signal(** ***fast*** **= 12,** ***slow*** **= 26,** ***signal*** **= 9)**
RETURNS	ARRAY
FUNCTION	Calculates the Signal line of MACD indicator.
EXAMPLE	signal(14, 28, 10);
SEE ALSO	

References:

The **Signal** function is used in the following formulas in AFL on-line library:

- AFL Example - Enhanced
- ALJEHANI
- BBAreacolor&TGLCROSSNEW
- Bollinger band normalization
- Button trading using AB auto trading interface
- Color MACD Histogram Changes
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Compare Sectors against Tickers
- Customised Avg. Profit %, Avg. Loss % etc
- Detailed Equity Curve
- Dinapoli Guru Commentary
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- ekeko price chart
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Triple Screen Trading System
- Forward/Reverse EMA by John Ehlers
- Fund Screener
- hassan
- ICHIMOKU SIGNAL TRADER
- Indicator Explorer (ZigZag)
- Last Five Trades Result Dashboard – AFL code
- MACD commentary
- MACD Histogram - Change in Direction
- MACD indicator display
- MACD optimize
- Meu Sistema de Trading - versão 1.0
- MFE and MAE and plot trades as indicator
- Rebalancing Backtest avoiding leverage
- ROC of MACD Weekly
- STO & MACD Buy Signals with Money-Management
- swing chart
- The Mean RSIt
- The Mean RSIt (variations)

- Trending or Trading ?
- Trending Ribbon
- TrendingRibbonArrowsADX
- Varexlist
- Vivek Jain

More information:

See updated/extended version on-line.

sin
- sine function

Math functions

SYNTAX	**sin(NUMBER)**
	sin(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Returns the sine of NUMBER or ARRAY. This function assumes that the ARRAY values are in radians.
EXAMPLE	You can plot a sine wave using the formula "sin(cum(0.05))." Increasing the value in this formula (i.e., "0.05") will increase the frequency of the sine wave.
SEE ALSO	The atan() function ; the cos() function.

References:

The **sin** function is used in the following formulas in AFL on-line library:

- AR_Prediction.afl
- Color Display.afl
- Cybernertic Hilbert Sine Wave
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period
- Ehlers Hilbert Transformer Indicator
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- Even Better Sinewave Indicator
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- John Ehler
- Luna Phase
- Moving Average "Crash" Test
- Multiple sinus noised
- Sine Wave Indicator
- Trigonometric Fit - TrigFit with AR for cos / sin

More information:

See updated/extended version on-line.

sinh
- hyperbolic sine function

Math functions
(AmiBroker 4.80)

SYNTAX	**sinh(NUMBER)**
	sinh(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Returns the hyperbolic sine of NUMBER or ARRAY. This function assumes that the ARRAY values are in radians.
EXAMPLE	
SEE ALSO	

References:

The **sinh** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Skewness - calculate skewness

Math functions
(AmiBroker 6.20)

SYNTAX	**Skewness(ARRAY, range, population = True)**
RETURNS	ARRAY
FUNCTION	The function calculates Skewness Skewness(ARRAY, range, False) - works the same as Excel's SKEW function Skewness(ARRAY, range, True) - works the same as Excel's SKEW.P function The Skewness of a data set is a measurement of the asymmetry of the distribution about the mean. A Skewness of zero indicates perfect symmetry: A positive Skewness indicates that more values lie below the mean and the distribution has a 'tail' which extends towards the higher values; A negative Skewness indicates that more values lie above the mean and the distribution has a 'tail' which extends towards the lower values.
EXAMPLE	
SEE ALSO	Kurtosis() function , StDev() function

References:

The **Skewness** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Sort
- performs a quick sort of the array

Miscellaneous functions
(AmiBroker 5.90)

SYNTAX **Sort(array, first = 0, last = -1, indexmode = False)**

RETURNS ARRAY

FUNCTION Sorts a numerical array in ascending order starting from **first** element ending at **last** element. If **last** is not specified or negative then AmiBroker will use BarCount - 1.

If **indexmode = 0** then returned array holds actual sorted values, if **indexmode = 1** the function returns array of indexes to the sorted values instead of values themselves. So if first returned value is 2923 it means that input array[2923] is the smallest element.

EXAMPLE

```
Filter = 1;
AddColumn( BarIndex(), "Bar Index", 1.0 );
AddColumn( Close, "Close" );
// normal mode
AddColumn( Sort( Close ), "Sorted Close" );
// index mode
AddColumn( Sort( Close, 0, -1, True ), "Index of sorted item", 1.0
);
```

SEE ALSO

References:

The **Sort** function is used in the following formulas in AFL on-line library:

- IBD relative strength database Viewer
- JEEVAN'S SRI CHAKRA
- sort function

More information:

See updated/extended version on-line.

SparseCompress - compress sparse array

Miscellaneous functions
(AmiBroker 60)

SYNTAX **SparseCompress(query_points, data)**

RETURNS ARRAY

FUNCTION The function gets values from 'data' array at points defined by non-zero values of 'query_points' array and compresses them so they are squeezed at the end of resulting array

EXAMPLE

```
only_when = ( Month() % 2 ) == 0; // even months only

x = SparseCompress( only_when, Close ); // compact sparse data

y = MA( x, 10 ); // regular calculation

y = SparseExpand( only_when, y ); // expand sparse data

Plot( C, "Price", colorDefault, styleBar );
Plot( y, "Sparse MA from even months", colorRed );


function SparseCompressEquiv( sparse_array, data_array )
{
     result = Null;

     j = BarCount - 1;
     for( i = BarCount - 1; i >= 0; i-- )
     {
       if( sparse_array[ i ] ) result[ j-- ] = data_array[ i ];
     }

     return result;
}

function SparseExpandEquiv( sparse_array, data_array )
{
     result = Null;

     j = BarCount - 1;
     for( i = BarCount - 1; i >= 0; i-- )
     {
       if( sparse_array[ i ] ) result[ i ] = data_array[ j-- ];
     }

     return result;
}
```

SEE ALSO SparseExpand() function

References:

The **SparseCompress** function is used in the following formulas in AFL on-line library:

- Polyfit Lines
- TAPE READING

More information:

See updated/extended version on-line.

SparseExpand - expand compressed array to sparse array

Miscellaneous functions
(AmiBroker 60)

SYNTAX **SparseExpand(query_points, data)**

RETURNS ARRAY

FUNCTION The function expands values from compressed 'data' array at points defined by non-zero values of 'query_points'

EXAMPLE

```
only_when = ( Month() % 2 ) == 0; // even months only

x = SparseCompress( only_when, Close ); // compact sparse data

y = MA( x, 10 ); // regular calculation

y = SparseExpand( only_when, y ); // expand sparse data

Plot( C, "Price", colorDefault, styleBar );
Plot( y, "Sparse MA from even months", colorRed );


function SparseCompressEquiv( sparse_array, data_array )
{
     result = Null;

     j = BarCount - 1;
     for( i = BarCount - 1; i >= 0; i-- )
     {
       if( sparse_array[ i ] ) result[ j-- ] = data_array[ i ];
     }

     return result;
}

function SparseExpandEquiv( sparse_array, data_array )
{
     result = Null;

     j = BarCount - 1;
     for( i = BarCount - 1; i >= 0; i-- )
     {
       if( sparse_array[ i ] ) result[ i ] = data_array[ j-- ];
     }

     return result;
}
```

SEE ALSO SparseCompress() function

References:

The **SparseExpand** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

SparseInterpolate
- interpolate values between sparse points given as input

Miscellaneous functions
(AmiBroker 6.90)

SYNTAX **SparseInterpolate(sparse_points, data, order)**

RETURNS ARRAY

FUNCTION Interpolate values between sparse data points

Parameters:

- sparse_points - non-zero values define sparse points
- data - input data
- order - the interpolator order/algorithm to use (1..5)

EXAMPLE

```
sparse_points = DayOfWeek() == 1;

y = SparseInterpolate( sparse_points, Close, 2 );

Plot( IIf( sparse_points, C, Null ), "Data", colorRed, styleDots );
Plot( y, "interpolated", colorBlue, styleLine );
```

SEE ALSO SparseCompress() function , SparseExpand() function

References:

The **SparseInterpolate** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

sqrt
- square root

Math functions

SYNTAX	**sqrt(NUMBER)**
	sqrt(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Calculates the square root of NUMBER or ARRAY. The square root of a negative number always returns a zero result.
EXAMPLE	The formula "sqrt(16)" returns 4
SEE ALSO	

References:

The **sqrt** function is used in the following formulas in AFL on-line library:

- 'R' Channel
- Basket Trading System T101
- Black Scholes Option Pricing
- channel indicator
- Commodity Selection Index (CSI)
- crMathLib
- Cycle Period
- Dave Landry PullBack Scan
- Ehlers Hilbert Transformer Indicator
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- Elder Triple Screen Trading System
- Even Better Sinewave Indicator
- Gann level plotter
- Heatmap V1
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- Historical Volotility Scan - 6/100
- Historical Volotility Scan - 50 Day
- HLspread
- Hull Moving Average
- Hull with DEMA
- Least Squares Channel Indicator
- Modified-DVI
- Multiple Ribbon Demo
- Option Calls, Puts and days till third friday.
- Polarized Fractal Efficiency
- Probability Calculator
- Probability Density & Gaussian Distribution
- Random Walk Index, base formula included
- RSI Trendlines and Wedges
- Sony
- Square of Nine Roadmap Charts

- Stress with SuperSmoother
- The Fibonaccian behavior
- Tracking Error
- Tracking Error
- Voss Predictive Filter (A Peek Into the Future)
- VWAP with standard deviation bands
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- z-distance from vwap

More information:

See updated/extended version on-line.

StaticVarAdd - an "atomic" addition for static variables

Miscellaneous functions
(AmiBroker 6.10)

SYNTAX **StaticVarAdd("name", value, keepAll = True, persistent = False)**

RETURNS NOTHING

FUNCTION StaticVarAdd implements an atomic addition (interlocked read-add-write) operation for static variables.

It is multithreading safe addition for static variables that are shared by multiple threads. This function is atomic with respect to calls to other static variable functions.

KeepAll flag when it is set to true emulates the behavior of AddToComposite. It keeps all values that are already present, so if data holes exists in current symbol,the bars that are present in static variable but not present in current symbol remain untouched. When *KeepAll* is set to false then only bars that are present in current symbol are kept. Any other bars that were present in static variable but not present in currently processed symbols are removed. That is what normally happens with StaticVarSet(). In fact when *KeepAll* is set to False, StaticVarAdd can be seen as the following pseudo code:

```
EnterCriticalSection
x = Nz( StaticVarGet( "name" ) ); // read exisiting value (and convert Nulls to zero)
x += Nz( value ); // add value to existing
StaticVarSet( "name", x ); // store updated value
LeaveCriticalSection
```

The function can be used to create composites like shown in the example below.

NOTES:

- StaticVarAdd automatically converts all Nulls to zeros (as AddToComposite does).
- If you want to replace AddToComposite with StaticVarAdd, keep in mind that by default AddToComposite skips symbols in group 253. This is done so composite symbols are not added to themselves. If you have composite symbols in your database and want to skip symbols in group 253 you can use
  if(GroupID() != 253) StaticVarAdd("~Composite", values);
- Thanks to extensive code tuning, StaticVarAdd generally offers better performance than AddToComposite which was already blazing fast. Single threaded StaticVarAdd may be twice as fast as ATC. With 8 threads running StaticVarAdd may be 4x as fast (it does not scale as much as naive person may think, because critical section limits performance due to lock contention). To illustrate the amount of fine tuning applied it can be said that first 'straightforward' version of StaticVarAdd was actually 20 times slower than ATC.
- Be careful when using "quickafl" as StaticVarAdd would not increase 'required bars' (as ATC does), so if you want to actually add all bars and quick afl is turned on in analysis, it is better to add
  SetBarsRequired(sbrAll, sbrAll)

EXAMPLE

```
if( status("stocknum") == 0 )
{
    // remove any earier composite values
```

```
   StaticVarRemove("~Composite");
}

StaticVarAdd( "~Composite", MACD() > Signal() );
Buy = 0;
```

SEE ALSO StaticVarSet() function , StaticVarGet() function , StaticVarCompareExchange() function

References:

The **StaticVarAdd** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

StaticVarCompareExchange - atomic interlocked static variable compare-exchange operation

Miscellaneous functions
(AmiBroker 5.50)

SYNTAX **StaticVarCompareExchange("varname", exchange, comperand)**

RETURNS NUMBER

FUNCTION Parameters:

- "varname" - Specifies the name of the destination static variable. Static variable if exists must be scalar numeric type. If static variable is not initialized, the function assumes that it has value of zero.
- exchange - specifies the exchange value. Scalar numeric.
- comperand - specifies the value to compare to the destination static variable. Scalar numeric.

Return Values:
The return value is the initial value of the destination static variable. If variable did not exist, it returns zero.

The StaticVarCompareExchange function performs an atomic comparison of the "varname" static variable value with the Comperand value. If the static variable value is equal to the Comperand value, the Exchange value is stored in the static variable. Otherwise, no operation is performed.

The function StaticVarCompareExchange provides a simple mechanism for synchronizing access to static variables that are shared by multiple threads. The following examples show how to implement semaphore and critical section in AFL using StaticVarCompareExchange function. For more details see Tutorial: Efficient use of multithreading.

EXAMPLE

```
// EXAMPLE 1 : Simple semaphore (no waiting)
if( StaticVarCompareExchange( "semaphore", 1, 0 ) == 0 ) // obtain
semaphore
{
   // protected section here
   // Here you have exclusive access (no other threads that check
for semaphore will enter simultaneously)
   /////////////////////////
   StaticVarSet("semaphore", 0 ); // reset semaphore
}
else
{
  _TRACE("Can not obtain semaphore");
}

///////////////
// EXAMPLE 2 HOW TO IMPLEMENT CRITICAL SECTION IN AFL
///////////////

function _TryEnterCS( secname )
```

```
{
  global _cursec;
   _cursec= "";

  // try obtaining semaphore for 1000 ms
  for( i = 0; i < 1000; i++ )
   if( StaticVarCompareExchange( secname, 1, 0 ) == 0 )
   {
       _cursec = secname;
      break;
   }
   else ThreadSleep( 1 ); //sleep one millisecond

  return _cursec != "";
}

// call it ONLY when _TryEnterCS returned TRUE !
function _LeaveCS()
{
   global _cursec;
   if( _cursec != "" )
     {
      StaticVarSet( _cursec, 0 );
       _cursec = "";
     }
}

function TimeConsumingWork()
{
   // WARNING: the Percentile is CPU hungry as it involves lots of
sorting, the loop below may take > 1 second to complete
   for( i = 0; i< 10; i++ ) Percentile( C, 100, 10 );
}

//_TRACE("Without CS Begin " + GetChartID() );
//TimeConsumingWork(); // some time consuming calculation
//_TRACE("Without CS End" + GetChartID() );

// Example usage (critical section)
if( _TryEnterCS( "mysemaphore" ) )
{
  // you are inside critical section now
   _TRACE("Begin CS " + GetChartID() );
   TimeConsumingWork(); // some time consuming calculation
   _TRACE("End CS " + GetChartID() );
   _LeaveCS();
}
else
{
   _TRACE("Unable to enter CS");
}
```

SEE ALSO ThreadSleep() function

References:

The **StaticVarCompareExchange** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

StaticVarCount - get the total number of static variables in memory

Miscellaneous functions
(AmiBroker 5.30)

SYNTAX	**StaticVarCount()**
RETURNS	NUMBER
FUNCTION	the function returns total number of static variables in memory
EXAMPLE	
SEE ALSO	StaticVarGet() function , StaticVarGetText() function , StaticVarRemove() function , StaticVarSet() function , StaticVarSetText() function

References:

The **StaticVarCount** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

StaticVarGenerateRanks - generate ranking of multiple symbols and store it to static variables

Miscellaneous functions
(AmiBroker 5.70)

SYNTAX **StaticVarGenarateRanks("outputprefix", "inputprefix", topranks, tiemode)**

RETURNS NOTHING

FUNCTION The function implements general-purpose multiple symbol bar-by-bar ranking.

StaticVarGenarateRanks("outputprefix", "inputprefix", topranks, tiemode)

"inputprefix" is a prefix that defines names of static variables that will be used as input for ranking. AmiBroker will search for all static variables that begin with that prefix and assume that remaining part of the variable name is a stock symbol. Say you want to rank stocks by ROC (rate of change). All you need to do is to store values into static variables.

Let us say that we will use static variable names like "ValuesToSortAPPL", "ValuesToSortMSFT", and so on.

To fill input static variables you can use this loop:

```
for( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
SetForeign(sym );
Value = ROC( C, 10 );
RestorePriceArrays();
StaticVarSet( "ValuesToSort" + sym, Value );
}
```

Now you are ready to perform sorting/ranking. There are two modes, normal ranking mode and Top/Bottom Rank mode.

Normal ranking mode is performed when toprank argument is set to zero.

StaticVarGenerateRanks("rank", "ValuesToSort", 0, 1224);

In this case StaticVarGenerateRanks call would generate set of static variables starting with prefix defined by 2nd argument each variable holding the rank of particular symbol, so in this case

RankValuesToSortMSFT will hold ranking of MSFT
RankValuesToSortAAPL will hold ranking of AAPL

Note that in AmiBroker rank count start from ONE.

Third argument (topranks) is zero in normal ranking mode

Fourth argument (tiemode) defines how ties are ranked. Supported modes are 1234 and 1224. In 1224 mode ties are numbered with equal rank.

Top/bottom ranking mode (that generates top/bottom ranking tables that hold indexes to top ranking values. When topranks > 0 top ranked values are used, when topranks < 0 then bottom ranked values are used. The values are stored in variables that have format of: OutputprefixInputprefixN where N is a number 1, 2, 3 representing top/bottom ranks. Let us assume that OutputPrefix parameter is "Top" and Inputprefix parameter is ROC. In such case variable TopROC1 would hold the index of top rated value. TopROC2 would hold second top rated value, and so on.

StaticVarGenerateRanks function uses rank numbering that starts from ONE.

In top ranking mode StaticVarGenerateRanks will also prepare static variable that contains comma separated list of variable names that can be used to find out which index refers to which symbol. So if TopROC1 holds 1 you would lookup first substring in TopROCSymbols variable to find out what variable (symbol) ranked at the top. Additionally StaticVarGetRankedSymbols gives easy-to-use method to retrieve comma separated list of ranked symbols for particular datetime.

EXAMPLE

```
/////////////////////////////////
// Example 1. code for normal ranking mode
// (everything done is done in one pass, can be used in indicator):
/////////////////////////////////

symlist = "C,CAT,DD,GE,IBM,INTC,MSFT";

// delete static variables - DO NOT forget the asterisk (wildcard)
at the end
StaticVarRemove( "ValuesToSort*" );

// fill input static arrays

for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    SetForeign( sym );
     Value = ROC( C, 10 );
    RestorePriceArrays();
    StaticVarSet( "ValuesToSort" + sym, Value );
}

// perform ranking
StaticVarGenerateRanks( "rank", "ValuesToSort", 0, 1224 ); // normal
rank mode

// read ranking
for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    Plot( StaticVarGet( "RankValuesToSort" + sym ), sym,
colorCustom10 + i );
}

/////////////////////////////////
// Example 2. Code for top ranking mode
// (everything done is done in one pass, can be used in indicator):
/////////////////////////////////


symlist = "C,CAT,DD,GE,IBM,INTC,MSFT";

// delete static variables - DO NOT forget the asterisk (wildcard)
at the end
StaticVarRemove( "ValuesToSort*" );

// fill input static arrays

for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    SetForeign( sym );
     Value = ROC( C, 10 );
    RestorePriceArrays();
    StaticVarSet( "ValuesToSort" + sym, Value );
}

// perform ranking
StaticVarGenerateRanks( "rank", "ValuesToSort", 0, 1224 ); // normal
rank mode

StaticVarGenerateRanks( "top", "ValuesToSort", 3, 1224 ); // top-N
mode

StaticVarGenerateRanks( "bot", "ValuesToSort", -3, 1224 ); //
bottom-N mode

// read ranking
for ( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
    Plot( StaticVarGet( "RankValuesToSort" + sym ), sym,
colorCustom10 + i );
}

sdt = SelectedValue( DateTime() );

Title = "{{NAME}} -{{DATE}} - {{VALUES}} TOP: " +
StaticVarGetRankedSymbols( "top", "ValuesToSort", sdt ) +
        " BOT: " + StaticVarGetRankedSymbols( "bot", "ValuesToSort",
sdt ) ;
```

SEE ALSO StaticVarGetRankedSymbols() function

References:

The **StaticVarGenerateRanks** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

StaticVarGet
- gets the value of static variable

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX **StaticVarGet(''varname', align = True')**

RETURNS NUMBER or STRING

FUNCTION Gets the value of static variable.

Static variable - the variable has static duration (it is allocated when the program begins and deallocated when the program ends) and initializes it to Null unless another value is specified. Static variables allow to share values between various formulas. ARRAY static variables are now supported (version 5.30 and above).

Please note that static array variable will consume 8 * (number_of_bars) bytes of memory and it won't be released until program is closed or variable is removed using StaticVarRemove().

Static arrays can be even 100 faster than AddToComposite/Foreign, however these two are not strictly equivalent.

There are following limitations / differences of static arrays as compared to Foreign/AddToComposite:
a) static array variables store only as many bars as there are currently in use by given chart (so they do not affect QuickAFL in any way). This is different that AddToComposite that forces usage and store of all bars.
b) static array variables work best if you read them using the same interval as they were written to. I.e when you create static array variables using 5-minute chart, for best results read them in some other 5-minute chart. Reading in different intervals is possible, but subject to limitations of timestamping (see below)
c) when you read static variable in a different interval that it was originally stored, static variables perform padding/synchronization and time compression/decompression automatically in a similar way as foreign, however Foreign compresses data always from base-time interval, while static variables operate on previously stored interval, hence result may differ. For example, if previously stored data was in daily interval, and you read such static variable in intraday chart, you will see essentially flat lines for each day, representing static data from daily interval.
d) static array variables do not work well for non-time based intervals (tick/n-volume/n-tick) because timestamps in those intervals may not be unique (i.e. several bars may have same time stamp), so time synchronization is not reliable.
e) static array variables are little slower than normal AFL variables, so for best performance, use read-once, write-once paradigm, using temporary normal variable for any processing during formula execution, like this: The new *align* parameter (default = true) decides whenever AmiBroker performs timestamp synchronization/alignment or not.

The default value is True and it means that values stored in static variables are retrieved and aligned to currently selected symbol data/timestamp on each bar basis so data for corresponding date/time stamps match. This is recommended setting and this is the way it worked in previous versions.

When align is switched to False - it means that AmiBroker does not perform any checks nor any alignment and will fill the array with consecutive values stored in static array regardless of their timestamps. If there are less bars in the static array than in the current arrays, the last value of static array will be propagated till BarCount - 1.

It is advised NOT to use align=False, unless you know exactly what you are doing and you are aware that date/time stamps have no meaning in particular variable or in case when date/time stamps are are aligned using your own method.

Note that speed difference between align 'on' and 'off' is usually negligible because alignment algorithm is very fast and has similar complexity as plain memory copy.

EXAMPLE

```
// start of the formula:
temp = StaticVarGet("mystaticarray" );

// now perform all necessary calculations using temp variable

temp = Nz(temp) + C/2;
...

// at the end of the formula store to static
StaticVarSet("mystaticarray", temp );
```

SEE ALSO StaticVarSet() function , StaticVarSetText() function , StaticVarGetText() function

References:

The **StaticVarGet** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- A simple AFL Revision Control System
- AFL Timing functions
- AFL_Glossary_Converter
- AllinOneAlerts - Module
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- channel indicator
- Continuous Contract Rollover
- elliott wave manual labelling
- Fast Refreshed KAGI Swing Charts (Price Swing)
- For Auto Trading Setup
- Gfx Toolkit
- Heatmap V1
- Herman
- interactively test discretionary trading
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Non-repaitning Zigzag line
- Now Send Push Notifications From Amibroker
- Rebalancing Backtest avoiding leverage

- Reconnect TWS
- suresh
- TAPE READING
- Visi-Trade
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

StaticVarGetRankedSymbols - retrieve a list of ranked symbols from static variables

Miscellaneous functions
(AmiBroker 5.70)

SYNTAX **StaticVarGetRankedSymbols("outputprefix", "inputprefix", datetime)**

RETURNS STRING

FUNCTION Retrieves the comma-separated list of symbols from static variables generated using StaticVarGenerateRanks. For more information see StaticVarGenerateRanks documentation.

EXAMPLE

```
symlist = "C,CAT,DD,GE,IBM,INTC,MSFT";

// delete static variables
StaticVarRemove("ValuesToSort*");

// fill input static arrays
for( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
SetForeign(sym );
Value = ROC( C, 10 );
RestorePriceArrays();
StaticVarSet( "ValuesToSort" + sym, Value );
}

// perform ranking
StaticVarGenerateRanks( "rank", "ValuesToSort", 0, 1224 ); // normal
rank mode
StaticVarGenerateRanks( "top", "ValuesToSort", 3, 1224 ); // top-N
mode
StaticVarGenerateRanks( "bot", "ValuesToSort", -3, 1224 ); //
bottom-N mode
// read ranking
for( i = 0; ( sym = StrExtract( symlist, i ) ) != ""; i++ )
{
Plot( StaticVarGet( "RankValuesToSort" + sym ), sym, colorCustom10 +
i );
}

sdt = SelectedValue( DateTime() );
Title = "{{NAME}} -{{DATE}} - {{VALUES}} TOP: " +
StaticVarGetRankedSymbols( "top", "ValuesToSort", sdt ) +
" BOT: " + StaticVarGetRankedSymbols( "bot", "ValuesToSort", sdt ) ;
```

SEE ALSO StaticVarGenerateRanks() function

References:

The **StaticVarGetRankedSymbols** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

StaticVarGetText - gets the value of static variable as string

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX **StaticVarGetText("varname")**

RETURNS STRING

FUNCTION Gets the value of static variable as string.
The only difference between StaticVarGet is that this function always returns string, so if given static variable does not exist it returns empty string "" instead of Null. Numbers are also converted to string.

Static variable - the variable has static duration (it is allocated when the program begins and deallocated when the program ends) and initializes it to Null unless another value is specified. Static variables allow to share values between various formulas. Array static variables are now supported (version 5.30 and higher)

EXAMPLE

```
myvar = StaticVarGetText("MyVariable");

if( myvar  == "" )
{
  printf("Not Set");
}
else
{
  printf("Variable Set: " + myvar);
}
```

SEE ALSO StaticVarGet() function , StaticVarSet() function , StaticVarSetText() function

References:

The **StaticVarGetText** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- Calculate composites for tickers in list files
- channel indicator
- Continuous Contract Rollover
- DateNum_DateStr
- elliott wave manual labelling
- Heatmap V1
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Ranking and sorting stocks
- suresh
- TWS auto-export Executions-file parser
- Visi-Trade
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

StaticVarInfo - get the information about static variable(s)

Miscellaneous functions
(AmiBroker 5.60)

SYNTAX **StaticVarInfo("varname", "field")**

RETURNS STRING or NUMBER

FUNCTION The function provides information about static variables.

Arguments:

- "varname" - is a variable name. It can be also a wildcard template such as "myvariable*" and then it means that AmiBroker will search for all variables beginning with " myvariable". * character matches any string, ? matches any single character
- "field" - defines the information to retrieve. Supported "field" values are:
    - ♦ "list" - returns the list of static variables
    - ♦ "memory" - returns memory usage in bytes (not including memory used for variable name itself)
    - ♦ "totalmemory" - returns memory usage in bytes (including memory used for variable name)
    - ♦ "count" - returns the number of static variables matching the wildcard string (new in AmiBroker 6.90)

EXAMPLE

```
StaticVarSet("my_array1", Close );
StaticVarSet("my_array2", Close );
StaticVarSet("my_scalar", 12 );
StaticVarSetText("my_text", "Text123456" );

"All variables in memory: " + StaticVarInfo( "*", "list" );
" Total static var memory: " + StaticVarInfo( "*", "totalmemory");
" Only my_ variables: " + StaticVarInfo( "my_*", "list" );
" Memory 2 arrays (bytes): " + StaticVarInfo( "my_array*", "memory"
);
" Memory scalar (bytes): " + StaticVarInfo( "my_scalar", "memory" );
" Memory text (bytes): " + StaticVarInfo( "my_text", "memory" );
```

SEE ALSO StaticVarCompareExchange() function , StaticVarCount() function , StaticVarGet() function , StaticVarGetText() function , StaticVarRemove() function , StaticVarSet() function , StaticVarSetText() function

References:

The **StaticVarInfo** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling

More information:

See updated/extended version on-line.

StaticVarRemove - remove static variable

Miscellaneous functions
(AmiBroker 4.80)

SYNTAX **StaticVarRemove("variablename")**

RETURNS NOTHING

FUNCTION This function removes static variable and releases associated memory.

With AmiBroker version 5.30, StaticVarRemove() supports wildcards in the variable name.

"varname" parameter can be either exact variable name or wildcard match string.

The '*' matches any number of characters, including zero characters. The '?' matches exactly one character.

Example 1:

StaticVarRemove("MyVariables*");
// this will remove all static variables beginning with MyVariables prefix.

EXAMPLE

```
StaticVarSet("DifferentName", 1 );

printf( "Total static variables = %g\n\n", StaticVarCount() );

for( i = 1; i <= 5; i++ )
  for( j = 1; j <= 5; j++ )
   {
     VarName = "Test_X=" + i + "_Y=" + j;
     printf("Setting variable " + VarName + "\n" );
     StaticVarSet( Varname, 1 );
   }

printf( "Total static variables = %g\n\n", StaticVarCount() );

printf( "Now wildcard remove *X=1*\n" );

StaticVarRemove( "*X=1*" );

printf( "Total static variables = %g\n\n", StaticVarCount() );

printf( "Now wildcard remove Test*\n" );

StaticVarRemove( "Test*" );

printf( "Total static variables = %g\n\n", StaticVarCount() );

printf("Removing 'differennname' variable\n");

StaticVarRemove("DifferentName" );
```

```
printf( "Total static variables = %g\n\n", StaticVarCount() );
```

SEE ALSO StaticVarGet() function , StaticVarGetText() function , StaticVarSet() function , StaticVarSetText() function

References:

The **StaticVarRemove** function is used in the following formulas in AFL on-line library:

- elliott wave manual labelling
- For Auto Trading Setup
- GFX ToolTip
- Visi-Trade

More information:

See updated/extended version on-line.

StaticVarSet
- sets the value of static variable

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX	**StaticVarSet("varname", value, persistent = False, compressionMode = cmDefault)**
RETURNS	NUMBER
FUNCTION	Sets the value of static variable. Returns 1 on success 0 on failure.

Static variable - the variable has static duration (it is allocated when the program begins and deallocated when the program ends) and initializes it to Null unless another value is specified. Static variables allow to share values between various formulas. ARRAY static variables are supported from version 5.30 and MATRIX static variables are supported from version 6.10

Please note that static array variable will consume 8 * (number_of_bars) bytes of memory and it won't be released until program is closed or variable is removed using StaticVarRemove().

Persistency

Starting from version 5.80 there is a new parameter *persist*. If it is set to True then static variable will be stored in PersistVars.bin file when AmiBroker is closing and reloaded automatically on next startup, preserving the values of static variables between application runs). In addition to saving them automatically on exit, persistent static variables can be auto-saved at user-specified intervals using SetOption("StaticVarAutoSave", interval);

Compression

Starting from version 6.10 there is a new parameter *compressionMode* that decides whenever given variable will be compressed or not.By default only persistent static variables will be compressed (cmDefault). You can turn it off completely compressionMode = cmNever, or turn it on for persitent and non-persistent variables using compressionMode = cmAlways

Compression is done by removing repeated values from the sequence as repeated values are restored when doing StaticVarGet. Compression is NOT compatible with non-aligned mode of StaticVarGet. If compressed array is retrieved by StaticVarGet with align=False, then repeated values found in original array would not be retrieved. Turning compression on slows down StaticVarSet (as it needs to do some extra processing), but does not affect performance of other functions, so StaticVarGet is equally fast with or without compression..

Static variables vs composites

Static arrays can be even 100 faster than AddToComposite/Foreign, however these two are not strictly equivalent.

There are following limitations / differences of static arrays as compared to Foreign/AddToComposite:

- static array variables store only as many bars as there are currently in use by given chart (so they do not affect QuickAFL in any way). This is different that AddToComposite that forces usage and store of all bars. This limitation applies to StaticVarSet but does NOT apply to StaticVarAdd (new in 6.10) which is designed to be functional equivalent of AddToComposite
- static array variables work best if you read them using the same interval as they were written to. I.e when you create static array variables using 5-minute chart, for best results read them in some other 5-minute chart. Reading in different intervals is possible, but subject to limitations of timestamping (see below)
- when you read static variable in a different interval that it was originally stored, static variables perform padding/synchronization and time compression/decompression automatically in a similar way as foreign, however Foreign compresses data always from base-time interval, while static variables operate on previously stored interval, hence result may differ. For example, if previously stored data was in daily interval, and you read such static variable in intraday chart, you will see essentially flat lines for each day, representing static data from daily interval.
- static array variables do not work well for non-time based intervals (tick/n-volume/n-tick) because timestamps in those intervals may not be unique (i.e. several bars may have same time stamp), so time synchronization is not reliable.
- static array variables are little slower than normal AFL variables, so for best performance, use read-once, write-once paradigm, using temporary normal variable for any processing during formula execution, like this:

EXAMPLE

```
// start of the formula:
temp = StaticVarGet("mystaticarray" );

// now perform all necessary calculations using temp variable

temp = Nz(temp) + C/2;
...

// at the end of the formula store to static
StaticVarSet("mystaticarray", temp );
```

SEE ALSO StaticVarSetText() function , StaticVarGet() function

References:

The **StaticVarSet** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- AFL Timing functions
- AFL_Glossary_Converter
- AllinOneAlerts - Module
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- channel indicator
- Continuous Contract Rollover
- elliott wave manual labelling
- Fast Refreshed KAGI Swing Charts (Price Swing)

- For Auto Trading Setup
- Gfx Toolkit
- Heatmap V1
- Herman
- interactively test discretionary trading
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Non-repaitning Zigzag line
- Now Send Push Notifications From Amibroker
- Ranking and sorting stocks
- Rebalancing Backtest avoiding leverage
- Reconnect TWS
- suresh
- TAPE READING
- Visi-Trade
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

StaticVarSetText - Sets the value of static string variable.

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX **StaticVarSetText("varname", "value", persist = False)**

RETURNS

FUNCTION Sets the value of static string variable. Returns 1 on success 0 on failure.

Static variable - the variable has static duration (it is allocated when the program begins and deallocated when the program ends) and initializes it to Null unless another value is specified. Static variables allow to share values between various formulas.

Starting from version 5.80 there is a new parameter 'persist'. If it is set to True then static variable will be stored in PersistVars.bin file when AmiBroker is closing and reloaded automatically on next startup, preserving the values of static variables between application runs). In addition to saving them automatically on exit, persistent static variables can be auto-saved at user-specified intervals using SetOption("StaticVarAutoSave", interval);

EXAMPLE

SEE ALSO StaticVarSet() function , StaticVarGet() function

References:

The **StaticVarSetText** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- Calculate composites for tickers in list files
- channel indicator
- Continuous Contract Rollover
- DateNum_DateStr
- elliott wave manual labelling
- Heatmap V1
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Ranking and sorting stocks
- suresh
- TWS auto-export Executions-file parser
- Visi-Trade
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

Status
- get run-time AFL status information

Miscellaneous functions
(AmiBroker 3.650)

SYNTAX **status(*"statuscode"*)**

RETURNS NUMBER or ARRAY

FUNCTION Returns run-time status of the analysis engine. Supported status codes:

- "stocknum" - gives you the ordinal number of currently analysed symbol
- "action" - gives information in what context given formula is run: 1 - actionIndicator (INDICATOR), 2 - actionCommentary (COMMENTARY), 3 - actionScan (SCAN), 4 - actionExplore (EXPLORATION), 5 - actionBacktest (BACKTEST / OPTIMIZE), 6 - actionPortfolio (portfolio backtest). The value of actionBacktest (5) (backtest) is used also in some other contexts (like code check and profile). Therefore you can use ActionEx to get more detailed/precise information
- "ActionEx" (new in 5.20) - more detailed information about action that triggered AFL execution. Note that 5 first codes are the same as Status("action") but scope is limited to 'core' meaning (see notes below).
  Possible values
  1. actionIndicator - when indicator is being repainted (NOTE: 5.32 - indicator can also give actionExInterpret when updating both chart and interpretation)
  2. actionCommentary (NOTE: commentary only, not interpretaion nor tooltip)
  3. actionScan - when AA Scan is performed
  4. actionExplore - when AA exploration is performed
  5. actionBacktest (NOTE backtest only, no optimization)
  6. actionPortfolio (2nd phase of portfolio backtest (custom backtest)
  7. reserved for future use
  8. reserved for future use
  9. reserved for future use
  10. - actionExAAShowArrows - when AA "Show arrows" command is used
  11. actionExAAParameters - when AA "Parameters" dialog is displayed/updated
  12. actionExEditVerifyFormula - when AFL editor verifies syntax
  13. actionExOptimizeSetup - when Optimize() parameters are read (to setup optimization engine)
  14. actionExOptimizeBacktest - when Backtest is performed as a part of optimization process
  15. actionExOptimizePortfolio - when portfolio-backtest phase (CUSTOM backtester) is performed as a part of optimization process
  16. actionExTooltip - when tooltip for given chart is being displayed/updated
  17. actionExInterpret - when the Interpretation window is being updated (can also mean indicator + interpretation in 5.32 above, see note below)
  18. (not used, reserved for future) actionExAAInit - when AA needs to initialize QuickAFL bars required information and/or formula contains functions changing general AA options

  NOTE: for backward compatiblity with all formulas you have written already, the codes for Status("action") did NOT change.

  NOTE ABOUT 5.32.x CHANGE: Since introduction of multi-threading, there is only

ONE pass/execution that updates both indicator and interpretation when current chart pane has focus. Status("action") will always return actionIndicator and Status("actionex") will either return actionExInterpret (when chart pane has focus and intepretation window is visible) or actionIndicator otherwise (when pane does not have focus or interpretation is NOT visible). Be careful NOT to disable Plot() depending on Status("actionex") code. If you really think you need to execute Plot() conditionally you should only check for Status("action")==actionIndicator.

- "rangefromdate", "rangetodate" - return current auto-analysis From-To range as DateNums
- "rangefromtime", "rangetotime" - return current auto-analysis From-To range as DateNums
- "barinrange" - returns 1 when current bar is within current auto-analysis From-To range
- "barvisible" - (custom indicators only) returns 1 when current bar is visible in current view
- "firstbarinrange" and "lastbarinrange". They return 1 (or True) on the first/last bar of analysis range.
- "buydelay", "selldelay", "shortdelay", "coverdelay" - return delays set in the Settings window
- "firstbarintest" and "lastbarintest" - similar to "firstbarinrange" and "lastbarinrange" but they return the settings of last BACKTEST/OPTIMIZATION and intermediate scans/explorations do not affect them
- "firstvisiblebar", "lastvisiblebar", "firstvisiblebarindex", "lastvisiblebarindex" - return bar number or bar index of first/last visible bar. Available in indicator mode only. Visible bar may potentially include "blank" future bars (past the last bar in the array) as defined in preferences
- "redrawaction" - returns 0 (zero) for regular refreshes, and 1 for refreshes triggered via RequestTimedRefresh().
- "pxwidth" - returns pixel width of chart window pane (indicators only, low-level gfx) (AmiBroker 4.94 or higher)
- "pxheight" - returns pixel height of chart window pane (indicators only, low-level gfx) (AmiBroker 4.94 or higher)
- "axisminy" - retrieves the minimum (bottom) value of Y axis (indicators only, low-level gfx)
- "axismaxy" - retrieves the maximum (top) value of Y axis (indicators only, low-level gfx)
- "pxchartleft" - returns x-coordinate of top-left corner of chart area
- "pxcharttop" - returns y-coordinate of top-left corner of chart area
- "pxchartright" - returns x-coordinate of bottom-right corner of chart area
- "pxchartbottom" - returns y-coordinate of bottom-right corner of chart area
- "pxchartwidth" - returns width chart area (right-left)
- "pxchartheight" - returns width chart area (bottom-top)
- "quickaflfirstdatabar", "quickafllastdatabar" - This feature is for internal use only. These are bar indexes of actual underlying compressed quotation array that make up AFL's array[0] and array[BarCount - 1]
- "timeshift" - returns database timeshift expressed in seconds (v5.60)
- "lastbarend" - returns DateTime of the end of last bar. For example 5 -minute bar at 9:00 will have end time of 9:04:59 (works for time-based bars only) (v5.60)
- "lastbartimeleft" - returns number of seconds to the completion of current last bar. Works for time-based bars only. Note that for proper operation this requires database timeshift to be set properly (so dates displayed on chart match your local computer

time zone). (v5.60)

- "lastbartimeleftrt" - it works like "lastbartimeleft" but uses the most recent RT stream update time instead of Now(). Also added Status("lastrtupdate") - time of last RT stream update Depends on RT plugin to deliver correct DateUpdate / TimeUpdate data. If plugin or date source sends incorrect datetimestamps or does not send DateUpdate/TimeUpdate correctly this function will not operate properly. Note that most data sources send weird (not current) datetime stamps on weekends. Also IQFeed plugin sends DateUpdate/TimeUpdate only inside regular trading hours. (v5.60)
- "lastrtupdate" - returns date time of last update sent by RT plugin (see remarks above) (v5.60)
- "ThreadID" - returns current thread ID under which formula is executed.

EXAMPLE Example 1:

```
if( Status("redrawaction") ==1 )
{
_TRACE("nTIMED REFRESH"+Now());
}
RequestTimedRefresh(1);
```

Example 2 (low-level graphic overlay + pixel co-ordinate conversion):

```
_SECTION_BEGIN("GfxOverlaySampleNew");

function GetVisibleBarCount()
{
 lvb = Status("lastvisiblebar");
 fvb = Status("firstvisiblebar");

 return Min( Lvb - fvb, BarCount - fvb );
}

function GfxConvertBarToPixelX( bar )
{
 lvb = Status("lastvisiblebar");
 fvb = Status("firstvisiblebar");
 pxchartleft = Status("pxchartleft");
 pxchartwidth = Status("pxchartwidth");

 return pxchartleft + bar  * pxchartwidth / ( Lvb - fvb + 1 );
}

function GfxConvertValueToPixelY( Value )
{
 local Miny, Maxy, pxchartbottom, pxchartheight;

 Miny = Status("axisminy");
 Maxy = Status("axismaxy");

 pxchartbottom = Status("pxchartbottom");
 pxchartheight = Status("pxchartheight");

 return pxchartbottom - floor( 0.5 + ( Value - Miny ) *
pxchartheight/ ( Maxy - Miny ) );
}



Plot(C, "Price", colorBlack, styleHistogram );

GfxSetOverlayMode(0);
GfxSelectSolidBrush( colorRed );
GfxSelectPen( colorRed );

AllVisibleBars = GetVisibleBarCount();
fvb = Status("firstvisiblebar");

for( i = 0; i < AllVisibleBars ; i++ )
{
   x = GfxConvertBarToPixelX( i );
   y = GfxConvertValueToPixelY( C[ i + fvb ] );

  GfxRectangle( x-1, y-1, x + 2, y+1 );
}

//SetChartBkGradientFill( ColorRGB(200,200,200), ColorRGB(
255,255,255) );
_SECTION_END();
```

SEE ALSO RequestTimedRefresh() function

References:

The **Status** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- A simple AFL Revision Control System
- AFL Example - Enhanced
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Auto Trade Step by Step
- Auto-Optimization Framework
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- babaloo chapora
- Backup Data of 1min Interval
- Basket Trading System T101

- BBAreacolor&TGLCROSSNEW
- Bid Vs Ask Dashboard
- Button trading using AB auto trading interface
- Calculate composites for tickers in list files
- Caleb Lawrence
- Candle Identification
- Candle Stick Analysis
- Candle Stick Demo
- CCI(20) Divergence Indicator
- channel indicator
- Channel/S&R and trendlines
- Color Display.afl
- Congestions detection
- Customised Avg. Profit %, Avg. Loss % etc
- Detailed Equity Curve
- DPO with shading
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- elliott wave manual labelling
- End Of Year Trading
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fibonacci Internal and External Retracements
- For Auto Trading Setup
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Halftrend
- Heatmap V1
- Herman
- High Low Detection code
- ICHIMOKU SIGNAL TRADER
- interactively test discretionary trading
- Intraday Fibonacii Trend Break System
- Inverted Plotted Volume Overlay Indicator
- Least Squares Channel Indicator
- MACD indicator display
- Manual Bracket Order Trader
- Market Breadth Chart-In-Chart
- Market Profile
- MFE and MAE and plot trades as indicator
- Multiple Ribbon Demo
- Murrey Math Price Lines
- Nadaraya-Watson Envelope
- Open Range Breakout Trading System
- Ord Volume
- pattenz
- Perceptron
- Periodically ReBalance a BUY & HOLD Portfolio
- Peter Cooper
- Pivot Finder
- Pivots And Prices

- prakash
- Probability Density & Gaussian Distribution
- Range Filter - Trading Strategy
- Ranking and sorting stocks
- Rebalancing Backtest avoiding leverage
- Relative Strength Multichart of up to 10 tickers
- Rene Rijnaars
- RUTVOL timing signal with BB Scoring routine
- shailu lunia
- Simple Candle Exploration
- Stan Weinstein strategy
- TAPE READING
- Trend Exploration: Count Number of New Highs
- Trend Exploration: Slope Moving Average
- Trix Bars number
- TWS auto-export Executions-file parser
- TWS trade plotter
- Ultimate plus
- Using From and To dates from Auto Analysis in Code
- Visible Min and Max Value Demo
- visual turtle trading system
- Volume Charts
- White Theme
- Wolfe Wave Patterns
- Woodie's CCI Panel Full Stats
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount

More information:

See updated/extended version on-line.

StdErr
- standard error

Statistical functions
(AmiBroker 3.40)

SYNTAX	**StdErr(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates standard error function (standard error of linear regression estimate) of the ARRAY over *periods* bars The function accepts periods parameter that can be constant as well as time-variant (array).
EXAMPLE	StdErr(close, 10);
SEE ALSO	

References:

The **StdErr** function is used in the following formulas in AFL on-line library:

- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Linear Candle
- Standard Error Bands (Native AFL)
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

StDev - standard deviation

Statistical functions
(AmiBroker 3.40)

SYNTAX	**StDev(ARRAY, *periods*, *Population* = True)**
RETURNS	ARRAY
FUNCTION	Calculates moving standard deviation of the ARRAY over *periods* bars AmiBroker 6.20 adds 3rd argument "Population = True". When Population is True it calculates population based stdev, otherwise sample based StDev(Array, range, False) - works the same as Excel's STDEV StDev(Array, range, True) - works the same as Excel's STDEV.P
EXAMPLE	stdev(close, 10);
SEE ALSO	Skewness() function

Comments:

Tomasz Janeczko 2006-04-04 16:26:27	Note that if you are trying to compare results of StDev function to Excel output you should use STDEVP function in Excel (not StDev).

References:

The **StDev** function is used in the following formulas in AFL on-line library:

- % B of Bollinger Bands With Adaptive Zones
- Adaptive Price Channel
- AR_Prediction.afl
- BB squeeze
- Bollinger Band Width
- Bollinger oscillator
- CCT Bollinger Band Oscillator
- Congestions detection
- correlerror
- CVR--severe filter
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Dave Landry PullBack Scan
- DMI Spread Index
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- Effective Swing Indicator
- Elder Triple Screen Trading System
- Elder's SafeZone Stop
- Follow the Leader

- Historical Volotility Scan - 6/100
- Historical Volotility Scan - 50 Day
- HLspread
- Inter-market Yield Linear Regression Divergence
- Kelly criterion
- Kiss and Touch with the Modified True Range
- Linear Regression Line w/ Std Deviation Channels
- MACD BB Indicator
- Moving Trend Bands (MTB)
- MultiCycle 1.0
- nikhil
- NR4 Historical Volatility System
- NRx Exploration
- Probability Calculator
- Probability Density & Gaussian Distribution
- Relative Volume
- Trend Exploration: Count Number of New Highs
- Trigonometric Fit - TrigFit with AR for cos / sin
- Volatility
- Volume Occilator
- Z-Score Indicator

More information:

See updated/extended version on-line.

StochD - stochastic slow %D

Indicators

SYNTAX	**StochD(*periods* = 14, *Ksmooth*=3, *Dsmooth*=3)**
RETURNS	ARRAY
FUNCTION	Calculates the %D line of Stochastic Oscillator (with internal slowing KSmooth, DSmooth).
EXAMPLE	The formula "stochd(5)" returns the value of a 5-period %D double smoothed by 3 periods
SEE ALSO	STOCHK() function

References:

The **StochD** function is used in the following formulas in AFL on-line library:

- Adaptave Zones O/B & O/S Oscillator
- Auto-Optimization Framework
- BBAreacolor&TGLCROSSNEW
- Bollinger Band Gap
- COMBO
- Dinapoli Guru Commentary
- Dinapoli Perferred Stochastic
- Divergences
- Fund Screener
- hassan
- ICHIMOKU SIGNAL TRADER
- Index and ETF trading
- Stochastic Divergence, negative
- Stochastic Divergence, positive
- Stochastic Divergences, PDI, NDI
- Stochastic optimize
- Stochastic Oscillator
- Stochastic OSI & OBI
- Stochastics Trendlines
- swing chart
- TrendingRibbonArrowsADX

More information:

See updated/extended version on-line.

StochK
- stochastic slow %K

Indicators

SYNTAX	**StochK(*periods* = 14, *ksmooth*=3)**
RETURNS	ARRAY
FUNCTION	Calculates the %K line of Stochastic Oscillator (with internal slowing KSmooth).
EXAMPLE	The formula "stochk(5)" returns the value of a 5-period %K slowed down 3 periods.
SEE ALSO	STOCHD() function

References:

The **StochK** function is used in the following formulas in AFL on-line library:

- Adaptave Zones O/B & O/S Oscillator
- Against all odds
- Auto-Optimization Framework
- BBAreacolor&TGLCROSSNEW
- CandleStochastics
- COMBO
- Dinapoli Guru Commentary
- Dinapoli Perferred Stochastic
- hassan
- ICHIMOKU SIGNAL TRADER
- Index and ETF trading
- Multiple Ribbon Demo
- Stochastic OBV and Price Filter
- Stochastic optimize
- Stochastic Oscillator
- Stochastics Trendlines
- swing chart
- TrendingRibbonArrowsADX

More information:

See updated/extended version on-line.

StrCount
- count the occurrences of substring within a string

String manipulation
(AmiBroker 5.20)

SYNTAX **StrCount("string", "substring")**

RETURNS NUMBER

FUNCTION Function returns integer which is number of times substring was found in string. It is case sensitive.

The function can be used for example to count the number of commas in comma-separated list

EXAMPLE

```
tickers = "AAPL,MSFT,INTC";

numtickers = 1 + StrCount( tickers, "," );
```

SEE ALSO StrExtract() function , StrFind() function , StrFormat() function , StrLeft() function , StrLen() function , StrMid() function , StrReplace() function , StrRight() function

References:

The **StrCount** function is used in the following formulas in AFL on-line library:

- Gfx Toolkit
- Optimal Weights

More information:

See updated/extended version on-line.

StrExtract
- extracts given item (substring) from comma-separated string

String manipulation
(AmiBroker 4.40)

SYNTAX **StrExtract(list, item, separator = ',')**

RETURNS STRING

FUNCTION Extracts given item (substring) from comma-separated list of items. item is a zero-based index of the item in the list (see also note below).

If no substring at given index is found then empty string is returned ("").

Useful to retrive symbols from the list obtained via GetCategorySymbols function.

New in AmiBroker version 5.20:
StrExtract("string", item) now accepts negative item values allowing to address items counting from the END of the list

New in AmiBroker version 5.90:
separator parameter allows to define separator other than comma

EXAMPLE

```
StrExtract( "MSFT,AAPL,AMD,INTC", 2 );// will return AMD

StrExtract( "MSFT,AAPL,AMD,INTC", 0 );// will return MSFT

StrExtract( "MSFT,AAPL,AMD,INTC", 200 );// will return empty string
""


//
// The example below shows how to use negative item
// references (Version 5.20 AND up only!)

tickers = "AAPL,MSFT,INTC";

"The last item is " + StrExtract( tickers, -1 );
printf("listing from the end of the list:n");

for( item = -1; ( sym = StrExtract( tickers, item ) ) != ""; item--
)
{
  printf( sym + "n" );
}
```

SEE ALSO GETCATEGORYSYMBOLS() function

References:

The **StrExtract** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- Add Nifty 50 IB Equity Symbol Automatically
- AFL_Glossary_1

- AFL_Glossary_Converter
- Backup Data of 1min Interval
- Baseline Relative Performance Watchlist charts V2
- Basket Trading System T101
- BEANS-Summary of Holdings
- Calculate composites for tickers in list files
- Candle Identification
- Count Tickers in Watchlist
- elliott wave manual labelling
- Expiry day/days - Last thursday of month
- Gfx Toolkit
- Graphical sector analysis
- Graphical sector stock amalysis
- Heatmap V1
- Herman
- IB Backfiller
- IBD relative strength database ranker
- IBD relative strength database Viewer
- MFE and MAE and plot trades as indicator
- Optimal Weights
- Peter Cooper
- Profit Table (Color Coded)
- Ranking and sorting stocks
- Ranking Ticker WatchList
- Relative Strength
- Rene Rijnaars
- Renko Chart
- suresh
- TWS auto-export Executions-file parser
- TWS trade plotter
- Updated Renko Chart
- White Theme
- WLBuildProcess

More information:

See updated/extended version on-line.

StrFind - find substring in a string

String manipulation
(AmiBroker 4.50)

SYNTAX **StrFind(string, substring)**

RETURNS NUMBER

FUNCTION The **StrFind** function finds first occurrence of substring in string.

Returns 0 if not found, otherwise returns character index (one-based) of first occurrence.

EXAMPLE

```
if( StrFind( Name(), ".L" ) )
{
   printf( "The " + Name() + " has .L suffix ");
}
else
{
   printf( "The " + Name() + " does not have .L suffix ");
}
```

SEE ALSO StrExtract() function

References:

The **StrFind** function is used in the following formulas in AFL on-line library:

- Advanced Search and Find
- AFL_Glossary_1
- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Basket Trading System T101
- BEANS-Summary of Holdings
- Extract specific lines of code from your program
- Get Moneycontrol News Snippets into Amiboker
- Graphical sector stock amalysis
- Heatmap V1
- INTRADAY HEIKIN ASHI new
- Pivots for Intraday Forex Charts
- Visi-Trade

More information:

See updated/extended version on-line.

StrFormat
- Write formatted output to the string

String manipulation
(AmiBroker 4.50)

SYNTAX **StrFormat(formatstr, ...)**

RETURNS STRING

FUNCTION The **StrFormat** function formats and returns a series of characters and values in the result string.

If arguments follow the format string, the format string must contain specifications that determine the output format for the arguments.

StrFormat and printf behave identically except that printf writes output to the window, while StrFormat does not write anything to output window but returns resulting string instead.

StrFormat function is useful with conjunction with **fputs** function that allows to write string to a file.

Note 1: for numbers always use %f, %e or %g formatting, %d or %x will not work because there are no integers in AFL.

Note 2: as of now only numbers and arrays can now be used. For arrays 'selected value' is used

Note 3: to print a single percent-sign character, you can not type % alone, you must use %%.

Starting from version 6.10, printf/StrFormat now implement a check for correct formatting string as sometimes users passed strings with % that is special marker for formatting string instead of %% to print actual percent sign. When check failes, "Error 61. The number of % formatting specifier(s) does not match the number of arguments passed." is displayed

Starting from version 6.20, printf/StrFormat support now "%s" (string specifier)

EXAMPLE

```
fh = fopen("Test.csv", "w" );
for( i = 0; fh && i < 10; i++ )
{
    text = StrFormat( "Hello world, line %g ", i );
   fputs( text, fh );
}

fclose( fh );
```

SEE ALSO printf() function , fputs() function

References:

The **StrFormat** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Abhimanyu
- Advanced MA system
- Advanced Trend Lines with S & R

- Advisory NRx price chart display.
- AFL_Glossary_1
- ALJEHANI
- Alphatrend
- Andrews PitchforkV3.3
- Aroon The Advisor
- Auto Trade Step by Step
- Automatic Linear Trend Channel
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- babaloo chapora
- Backup Data of 1min Interval
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Button trading using AB auto trading interface
- changing period moving avarage
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- Continuous Contract Rollover
- Coppock Trade Signal on Price Chart
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- DateNum_DateStr
- Dave Landry PullBack Scan
- Day Bar No
- DEBAJ
- DiNapolis 3x Displaced Moving Averages
- Double Super Trend System
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- Fast Refreshed KAGI Swing Charts (Price Swing)
- FastStochK FullStochK-D
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- For Auto Trading Setup
- Fre
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Gann level plotter
- Gann Swing Charts in 3 modes with text
- Gfx Toolkit
- Guppy Cloud
- Halftrend
- Harmonic Patterns

- Heikin Ashi System
- Heikin-Ashi(Koma-Ashi) with Moving Average
- HH-LL-PriceBar
- High Low Detection code
- How to add IB Option Symbols
- IB Backfiller
- IBD relative strength database ranker
- IBD relative strength database Viewer
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- Intraday Fibonacii Trend Break System
- Intraday Strength
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- LunarPhase
- Manual Bracket Order Trader
- MFE and MAE and plot trades as indicator
- MS Darvas Box with Exploration
- New HL Scanner
- Next Date Format
- nifty
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- pattenz
- Pivot End Of Day Trading System
- Pivots And Prices
- plot tomorrows pivots on an intraday database
- prakash
- Price Chart - Fundamental
- Prior Daily OHLC
- PVT Trend Decider
- Range Filter - Trading Strategy
- Renko Chart
- Robert Antony
- SAR-ForNextBarStop
- Schiff Lines
- shailu lunia
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Stochastic Oscillator
- Super Trend Indicator
- Super Trend Indicator
- suresh
- TD Sequential
- TrendingRibbonArrowsADX
- Updated Renko Chart
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- William's Alligator System II
- WILSON RELATIVE PRICE CHANNEL
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels

- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

StrLeft
- extracts the leftmost part

String manipulation
(AmiBroker 40)

SYNTAX	**strleft(STRING, *count*)**
RETURNS	STRING
FUNCTION	Extracts the first (that is, leftmost) *count* characters from STRING and returns a copy of the extracted substring. If *count* exceeds the string length, then the entire string is extracted.
EXAMPLE	newstring = strleft(string, 4);
SEE ALSO	

References:

The **StrLeft** function is used in the following formulas in AFL on-line library:

- Advanced Search and Find
- AFL_Glossary_1
- AFL_Glossary_Converter
- Auto Trade Step by Step
- Auto-Optimization Framework
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- babaloo chapora
- Basket Trading System T101
- BEANS-Summary of Holdings
- Binary to Decimal Converter
- Button trading using AB auto trading interface
- Create a list of functions in your program
- DateNum_DateStr
- Date_To_Num(), Time_To_Num()
- Futures - Dollar Move Indicator
- Futures - Dollar Move Today Indicator
- Gordon Rose
- Graphical sector stock amalysis
- Heatmap V1
- Herman
- How to add IB Option Symbols
- IB Backfiller
- IBD relative strength database ranker
- IBD relative strength database Viewer
- Improved NH-NH scan / indicator
- Manual Bracket Order Trader
- MFE and MAE and plot trades as indicator
- pattenz
- Pivot Finder
- Ranking and sorting stocks
- shailu lunia
- TAPE READING
- Time Left in Bar
- TWS auto-export Executions-file parser

- TWS trade plotter
- White Theme

More information:

See updated/extended version on-line.

StrLen
- string length

String manipulation
(AmiBroker 3.50)

SYNTAX	**strlen(STRING)**
RETURNS	NUMBER
FUNCTION	calculates the length of the string
EXAMPLE	This function could be used for (for example) filtering out only 3 letter stock codes: buy = something AND **strlen**(name()) == 3;
SEE ALSO	

References:

The **StrLen** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- Advanced Search and Find
- AFL_Glossary_1
- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Dollar Price Volatility Exploration
- Basket Trading System T101
- BEANS-Summary of Holdings
- Binary to Decimal Converter
- Button trading using AB auto trading interface
- Calculate composites for tickers in list files
- channel indicator
- Graphical sector stock amalysis
- Heatmap V1
- Herman
- IB Backfiller
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- pattenz
- Profit Table (Color Coded)
- Ranking and sorting stocks
- TAPE READING
- Time Left in Bar
- TWS auto-export Executions-file parser

More information:

See updated/extended version on-line.

StrMatch - string pattern/wildcard matching

String manipulation
(AmiBroker 60)

SYNTAX **StrMatch("string", "searchstring")**

RETURNS NUMBER

FUNCTION The function returns TRUE (1) or FALSE (0) whenever string matches searchstring or not.

Searchstring is can contain wild-card characters such as:
* - matches any string, including empty strings
? - matches any single character

This function is case sensitive (of course except wildcard characters).

If you want case insensitive matching - convert both string and searchstring to lowercase or uppercase prior to matching (StrToLower/StrToUpper)

EXAMPLE

```
x = StrMatch("Every breath you take", "Every * you *"); // x will be
TRUE
x = StrMatch("Every step you make", "Every * you *"); // x will be
TRUE
```

SEE ALSO StrFind() function , StrToLower() function , StrToUpper() function

References:

The **StrMatch** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

StrMid
- extracts part of the string

String manipulation
(AmiBroker 40)

SYNTAX **StrMid(STRING, *start*, *count*)**

RETURNS STRING

FUNCTION Extracts a substring of length *count* characters from STRING, starting at position *start* (zero-based). The function returns a copy of the extracted substring.

New in version 5.90 - *count* parameter can be skipped. If you skip *count* then a substring starting from *start* to the end of the string will be returned.

EXAMPLE newstring = strmid(string, 1, 2);

SEE ALSO

References:

The **StrMid** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_1
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Basket Trading System T101
- channel indicator
- DateNum_DateStr
- Date_To_Num(), Time_To_Num()
- Futures - Dollar Move Indicator
- Futures - Dollar Move Today Indicator
- Heatmap V1
- How to add IB Option Symbols
- IBD relative strength database ranker
- IBD relative strength database Viewer
- Least Squares Channel Indicator
- MFE and MAE and plot trades as indicator
- Profit Table (Color Coded)
- TWS auto-export Executions-file parser
- TWS trade plotter

More information:

See updated/extended version on-line.

StrReplace - string replace

String manipulation
(AmiBroker 4.90)

SYNTAX	**StrReplace(srcstring, oldsubstring, newsubstring)**
RETURNS	STRING
FUNCTION	This function returns a string with all occurrences of *oldsubstring* in *srcstring* replaced with the given *newsubstring* value. The string may grow or shrink as a result of the replacement, that is *oldsubstring* and *newsubstring* do not have to be equal in length. The function performs case-sensitive matches.
EXAMPLE	`// the expression below will` `// result in string in which 'red' is replaced with 'brown'` `StrReplace("This fox is red", "red", "brown" );`
SEE ALSO	StrExtract() function , StrFind() function , StrFormat() function , StrLeft() function , StrLen() function , StrMid() function , StrRight() function , StrToDateTime() function , StrToLower() function , StrToNum() function , StrToUpper() function

References:

The **StrReplace** function is used in the following formulas in AFL on-line library:

- A simple AFL Revision Control System
- AFL_Glossary_1
- AFL_Glossary_Converter
- Calculate composites for tickers in list files
- Herman
- White Theme
- WLBuildProcess

More information:

See updated/extended version on-line.

StrRight
- extracts the rightmost part of the string

String manipulation
(AmiBroker 40)

SYNTAX	**StrRight(STRING, *count*)**
RETURNS	STRING
FUNCTION	Extracts the last (that is, rightmost) *count* characters from STRING and returns a copy of the extracted substring. If *count* exceeds the string length, then the entire string is extracted.
EXAMPLE	newstring = strright(string, 4);
SEE ALSO	

References:

The **StrRight** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Advanced Search and Find
- AFL_Glossary_1
- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- BEANS-Summary of Holdings
- Binary to Decimal Converter
- DateNum_DateStr
- Date_To_Num(), Time_To_Num()
- Fibonacci Internal and External Retracements
- Graphical sector stock amalysis
- Heatmap V1
- How to add IB Option Symbols
- IBD relative strength database ranker
- IBD relative strength database Viewer
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- MFE and MAE and plot trades as indicator
- Pivots And Prices
- Relative Strength Multichart of up to 10 tickers
- Time Left in Bar
- TWS auto-export Executions-file parser
- TWS trade plotter

More information:

See updated/extended version on-line.

StrSort
- sort comma-separated item list

String manipulation
(AmiBroker 5.90)

SYNTAX **Sort("tem,list,to,be,sorted", caseSensitive = True, separator = ',')**

RETURNS STRING

FUNCTION Perform sorting of comma-separated item list given in a string. The comma is default separator, but it can be any separator that user chooses in *separator* argument. The sort is either case sensitive (caseSensitive=True) or not (caseSensitive=False). The sort order is ascending. The function returns string with items sorted.

EXAMPLE

```
EnableTextOutput( 0 );

str = "MSFT,INTC,AAPL,GOOG";
printf( "Before sort: '" + str + "' " );

str = StrSort( str );

printf( "After sort: '" + str + "'" );
```

SEE ALSO Sort() function

References:

The **StrSort** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

StrToDateTime - convert string to datetime

String manipulation
(AmiBroker 4.80)

SYNTAX **StrToDateTime("string")**

RETURNS NUMBER

FUNCTION Converts string representing date/time value to the corresponding DateTime number (that can be later compared to output of DateTime() function for example).

This function has shorter synonym: _DT function.

VERSION 5.27 and above: It is important to understand that DateTime is not a simple number but rather bitset and two datetime values can only be reliably compared for equlity or inequality using == or != operators. Any other comparisions (less than/greater then) using normal operators > < can lead to wrong results, therefore to compare two datetime numbers you should use DateTimeDiff(arg1, arg2) which will return positive values if arg1 > arg2 and negative values if arg1 < arg2.

EXAMPLE
```
Buy = DateTime() == StrToDateTime("2005-Mar-05");
```

SEE ALSO DATETIME() function , DateTimeToStr() function , _DT() function

References:

The **StrToDateTime** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_1
- BEANS-Summary of Holdings
- High Low Detection code
- Improved NH-NH scan / indicator
- TWS trade plotter

More information:

See updated/extended version on-line.

StrToLower - convert to lowercase

String manipulation
(AmiBroker 4.80)

SYNTAX	**StrToLower("string")**
RETURNS	STRING
FUNCTION	This function converts input string to all lower case.
EXAMPLE	`Title = StrToLower( "MiXeD CaSe" );`
SEE ALSO	StrToUpper() function

References:

The **StrToLower** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_1
- AFL_Glossary_Converter

More information:

See updated/extended version on-line.

StrToNum - convert string to number

String manipulation
(AmiBroker 4.50)

SYNTAX **StrToNum(string)**

RETURNS NUMBER

FUNCTION Converts string to number.

EXAMPLE

```
List = "123,456,789";

for( i = 0; ( Item = StrExtract( List, i ) ) != ""; i++ )
{
   printf( "%gn", StrToNum( Item ) );
}
```

SEE ALSO WRITEVAL() function

References:

The **StrToNum** function is used in the following formulas in AFL on-line library:

- Advanced Trend Lines with S & R
- AFL_Glossary_1
- ALJEHANI
- AutoTrade using an Exploration
- BEANS-Summary of Holdings
- channel indicator
- Daily High Low in Advance
- DateNum_DateStr
- Date_To_Num(), Time_To_Num()
- Expiry day/days - Last thursday of month
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Graphical sector stock amalysis
- Heatmap V1
- How to add IB Option Symbols
- IBD relative strength database ranker
- IBD relative strength database Viewer
- Least Squares Channel Indicator
- Market Profile
- MFE and MAE and plot trades as indicator
- Square of Nine Roadmap Charts
- suresh
- TD Sequential
- TWS auto-export Executions-file parser
- TWS trade plotter
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots

More information:

See updated/extended version on-line.

StrToUpper - convert to uppercase

String manipulation
(AmiBroker 4.80)

SYNTAX	**StrToUpper("string")**
RETURNS	STRING
FUNCTION	This function converts input string to all upper case.
EXAMPLE	`Title = StrToUpper( "MiXeD CaSe" );`
SEE ALSO	StrToLower() function

References:

The **StrToUpper** function is used in the following formulas in AFL on-line library:

- Advanced Search and Find
- AFL_Glossary_1
- AFL_Glossary_Converter

More information:

See updated/extended version on-line.

StrTrim
- trim whitespaces from the string

String manipulation
(AmiBroker 5.90)

SYNTAX **StrTrim("string", "targets", side = 3)**

RETURNS STRING

FUNCTION The function trims extra characters (specified in "targets") from either left (1), right(2) or both(3) sides of the string. If "targets" parameter is an empty string, then function trims whitespaces (i.e. space, tab, newline)

EXAMPLE

```
EnableTextOutput( 0 );

str = "== ==string with extra chars== ==";

printf( "Before trim: '" + str + "'\n" );

str = StrTrim( str, " =" );

printf( "After trim: '" + str + "'" );
```

SEE ALSO

References:

The **StrTrim** function is used in the following formulas in AFL on-line library:

- Get Moneycontrol News Snippets into Amiboker

More information:

See updated/extended version on-line.

Study
- reference hand-drawn study

Miscellaneous functions
(AmiBroker 3.50)

SYNTAX **Study(STUDYID, CHARTID = 1, scale = -1)**

RETURNS ARRAY

FUNCTION generates an array equivalent to a trendline study drawn by the user - allows detecting trendline breakouts from AFL.
STUDYID is a two-character identifier of the study. identifiers are: "UP" - uptrend, "DN" - downtrend, "SU" - support, "RE" - resistance, "ST" - stop loss, however you can use ANY identifiers (there are no limitations except that AmiBroker accepts only 2 letter codes).
CHARTID - identifies the chart pane where the study was drawn - you can find out what is the chart ID for given chart by looking in Parameters dialog, Axes & Grid, Miscellaneous: Chart ID or using GetChartID() AFL function.
Scale - this parameter specifies which scale should be used:

- **scale = -1 : automatic (default value) - either linear or logarithmic depending on actual chart setting, chart is specified by chartID**
- **scale = 0 : linear scale**
- **scale = 1 : logarithmic scale**

More information about this function is included in the Tutorial: Using Studies in AFL formulas

EXAMPLE
```
// this example plots filled area between
// support (SU) and resistance (RE) lines

Plot(C, "Price", colorBlack, styleCandle );
su = Study("SU", GetChartID() );
re = Study("RE", GetChartID() );
PlotOHLC( re,  re,  su, su, "", colorYellow,styleCloud );
```

SEE ALSO GETCHARTID() function

References:

The **Study** function is used in the following formulas in AFL on-line library:

- Plot visual stop / target ratio.

More information:

See updated/extended version on-line.

Sum - sum data over specified number of bars

Moving averages, summation

SYNTAX	**Sum(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates a cumulative sum of the ARRAY for the specified number of lookback *periods* (including today). The function accepts periods parameter that can be constant as well as time-variant (array).
EXAMPLE	The formula "sum(CLOSE, 14)" returns the sum of the preceding 14 closing prices. A 14-period simple moving average could be written "sum(C,14) / 14."
SEE ALSO	CUM() function

Comments:

Graham Kavanagh gkavanagh [at] e-wire.net.au 2004-08-09 07:52:41	Sum adds up the last "n" number of bars. It sums whatever you put into the first part of the sum formula.
	
	Cum(1) adds 1 to the previous value of Cum, so the first bar is 1 and it just keeps adding one to the last bar value of cum(1).
	You can use Cum to add anything, like how many times you get rising days in the entire chart:
	
	Rise = C>O; //this gives results of 0 or 1
	TotalRise = Cum(Rise);
	
	You could limit this as well to time periods, or any other condition Example would be one for total rise days since 1995:
	
	RecentRise = C>O and Year()>=1995; //this gives results of 0 or 1
	TotalRise = Cum(RecentRise);
	
	If you wanted to know how many rising days in the last 12 bars you would use:
	
	LastRises = Sum(Rise,12);
	
	Hope this helps

References:

The **Sum** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Adaptive Relative Vigour Index
- Against all odds
- AJDX system
- Alpha and Beta and R_Squared Indicator
- Alternative ZIG function
- Alternative ZIG type function, multi TF

- AR_Prediction.afl
- Auto-Optimization Framework
- Automatic Trendlines using multiple timeframes
- Bollinger band normalization
- Buff Volume Weighted Moving Averages
- Buyer Seller Force
- Caleb Lawrence
- CandleStick Comentary--Help needed
- CandleStochastics
- Chaikin Money Flow
- Chande Momentum Oscillator
- Channel/S&R and trendlines
- Cole
- Connors TPS
- crBeta
- crMathLib
- DeMarker
- Dynamic Momentum Index
- Dynamic Momentum Index
- Dynamtic Momentum Index
- ekeko price chart
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's SafeZone Stop
- ElderSafeZoneStopLong
- ElderSafeZoneStopShort
- Fib CMO
- Fisher Relative Vigour Index
- Fre
- Frequency distribution of returns
- Heatmap V1
- Hilbert Study
- Hurst Constant
- Intraday Average Volume
- Kiss and Touch with the Modified True Range
- Linear Regression Line & Bands
- MACD commentary
- Market Meanness Index
- Momentum Volume Price (MVP) Indicator
- MultiCycle 1.0
- Open Range Breakout Trading System
- Perceptron
- Performance Check
- Peterson
- Polarized Fractal Efficiency
- Projection Oscillator
- QP2 Float Analysis
- Range Expansion Index
- Regression Analysis Line
- Relative Vigour Index
- Relative Vigour Index
- RSI of Weekly Price Array

- RSIS
- Sector Tracking
- Sony
- Stochastic of Weekly Price Array
- Stochastic Relative Vigour Index
- TD REI
- TD sequential
- TD Sequential
- The Saturation Indicator D_sat
- Time Frame Weekly Bars
- Time segment value
- Tracking Error
- Tracking Error
- Trailing Stop Loss
- Trend Analysis_Comentary
- Trend Continuation Factor
- Trend Exploration: Count Number of New Highs
- Trend Following System
- Trigonometric Fit - TrigFit with AR for cos / sin
- TSV
- Vertical Horizontal Filter
- Vertical Horizontal Filter (VHF)
- Visualization of stoploses and profit in chart
- Volume Weighted Moving Average
- Volume wieghted moving average
- VWAP - Volume Weighted Average Price
- VWAP versus Average Price
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- z-distance from vwap
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

SumSince
- sum of array elements since condition was tru

Moving averages, summation
(AmiBroker 6.10)

SYNTAX **SumSince(condition, array, incFirst = False)**

RETURNS Array

FUNCTION The function calculates running sum of *array* elements since *condition* was true. It works like:

```
x = Cum( array ) - ValueWhen( condition, Cum( array ) );
```

or like:

```
x = Sum( array, BarsSince( condition ) );
```

but much faster.

When incFirst is set to True, the sum includes the very first value at the bar when condition was true.

EXAMPLE

SEE ALSO Cum() function , Sum() function , BarsSince() function , ValueWhen() function

References:

The **SumSince** function is used in the following formulas in AFL on-line library:

- TAPE READING

More information:

See updated/extended version on-line.

tan
- tangent function

Math functions
(AmiBroker 30)

SYNTAX	**tan(NUMBER)**
	tan(ARRAY)

RETURNS	NUMBER,ARRAY
FUNCTION	Returns the tangent of NUMBER or ARRAY. This function assumes that the ARRAY values are in radians
EXAMPLE	
SEE ALSO	atan() function

References:

The **tan** function is used in the following formulas in AFL on-line library:

- Andrews Pitchfork
- Andrews PitchforkV3.3
- AR_Prediction.afl
- Cybernertic Hilbert Sine Wave
- DMI Spread Index
- Dominant Cycle Phase
- Ehlers Hilbert Transformer Indicator
- Even Better Sinewave Indicator
- Gabriel Linear Regression Angle Indicator
- Heatmap V1
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- Hilbert Study
- John Ehler
- Moving Average "Crash" Test
- Multiple sinus noised
- Schiff Lines
- Signal to Noise
- Sine Wave Indicator
- Smoothed Adaptive Momentum
- Three Pole Butterworth
- Trigonometric Fit - TrigFit with AR for cos / sin
- Voss Predictive Filter (A Peek Into the Future)
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots

More information:

See updated/extended version on-line.

tanh
- hyperbolic tangent function

Math functions
(AmiBroker 4.80)

SYNTAX	**tanh(NUMBER)**
	tanh(ARRAY)

RETURNS	NUMBER, ARRAY
FUNCTION	Returns the hyperbolic tangent of NUMBER or ARRAY. This function assumes that the ARRAY values are in radians.
EXAMPLE	
SEE ALSO	

References:

The **tanh** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

TEMA
- triple exponential moving average

Moving averages, summation
(AmiBroker 40)

SYNTAX	**tema(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates triple exponentially smoothed average - TEMA. The function accepts time-variable *periods.*
EXAMPLE	TEMA(Close, 5)
SEE ALSO	MA(), EMA(), WMA(), DEMA()

Comments:

Graham Walker
helpman [at]
dodo.com.au
2005-02-18 04:51:33
	//TEMA can be implemented via EMA:
	
	Len=10;
	MyTEMA = 3 * EMA(Close,len) - 3 * EMA(EMA(Close,len),Len) + EMA(EMA(EMA(Close,len),len),len);
	
	Plot(MyTEMA,"MyTEMA",colorBlue);
	
	// for comparison only
	Plot(TEMA(Close, Len), "Built-in TEMA", colorRed);

References:

The **TEMA** function is used in the following formulas in AFL on-line library:

- Auto-Optimization Framework
- Average Price Crossover
- babaloo chapora
- Bad Tick Trim on 5 sec database
- Balance of Power
- balance of power
- BBAreacolor&TGLCROSSNEW
- BMTRIX Intermediate Term Market Trend Indicator
- Bull/Bear Volume
- Dahl Oscillator modified
- DEBAJ
- Elder Impulse Indicator V2
- Heatmap V1
- Modified-DVI
- Volume Occilator

More information:

See updated/extended version on-line.

ThreadSleep
- suspend thread for specified number of milliseconds

Miscellaneous functions
(AmiBroker 5.50)

SYNTAX	**ThreadSleep(milliseconds)**
RETURNS	NOTHING
FUNCTION	ThreadSleep(milliseconds) suspends current thread for specified number of milliseconds (maximum is 100 ms). Works only from NON-UI threads. When called from UI thread the function does NOTHING and returns immediatelly. Please do NOT abuse this function. Using it may negatively impact performance. The function is provided for advanced users to implement inter-thread synchronization. For more details see Tutorial: Effective use of multi-threading
EXAMPLE	
SEE ALSO	StaticVarCompareExchange() function

References:

The **ThreadSleep** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

TimeFrameCompress - compress single array to given time frame

Time Frame functions
(AmiBroker 4.50)

SYNTAX **TimeFrameCompress(** ***array, interval, mode = compressLast*** **)**

RETURNS ARRAY

FUNCTION The **TimeFrameCompress** function compresses single array to given interval using given compression mode available modes:

- compressLast - last (close) value of the array within interval
- compressOpen - open value of the array within interval
- compressHigh - highest value of the array within interval
- compressLow - lowest value of the array within interval
- compressVolume - sum of values of the array within interval

To expand compressed array you should use the **TimeFrameExpand** function.

The **TimeFrameCompress** function is provided for completeness and it can be used when you want to compress single array without affecting built-in OHLC,V arrays. If you call TimeFrameCompress it does not affect results of other functions (opposite to **TimeFrameSet**).

For more information check Tutorial: Multiple time frame support

EXAMPLE

```
wc = TimeFrameCompress( Close, inWeekly );

/* now the time frame is still unchanged (say daily) and our MA will
operate on daily data */
dailyma = MA( C, 14 );

/* but if we call MA on compressed array, it will give MA from other
time frame */
weeklyma = MA( wc, 14 ); // note that argument is time-compressed
array

Plot( dailyma, "DailyMA", colorRed );

weeklyma = TimeFrameExpand( weeklyma, inWeekly ); // expand for
display

Plot( weeklyma, "WeeklyMA", colorBlue );
```

SEE ALSO TimeFrameExpand() function

References:

The **TimeFrameCompress** function is used in the following formulas in AFL on-line library:

- Intraday Range and Periods Framer
- Market Breadth Chart-In-Chart
- Volume Color with Dynamic Limit

More information:

See updated/extended version on-line.

TimeFrameExpand - expand time frame compressed array

Time Frame functions
(AmiBroker 4.50)

SYNTAX **TimeFrameExpand(*array, interval, mode = expandLast*)**

RETURNS ARRAY

FUNCTION The **TimeFrameExpand** function expands time-compressed *array* from *interval* time frame to base time frame (*interval* parameter must match the value used in **TimeFrameCompress** or **TimeFrameSet**)

The **TimeFrameExpand** is used to decompress array variables that were created in different time frame. Decompressing is required to properly display the array created in different time frame. For example if you want to display weekly moving average it must be 'expanded' so the data of one weekly bar covers five daily bars (Monday-Friday) of corresponding week.

Available *mode*s:

- expandLast - the compressed value is expanded starting from last bar within given period (so for example weekly close/high/low is available on Friday's bar)
- expandFirst - the compressed value is expanded starting from first bar within given period (so for example weekly open is available from Monday's bar)
- expandPoint - the resulting array gets not empty values only for the last bar within given period (all remaining bars are Null (empty)).

Caveat: expandFirst used on price different than open may look into the future. For example if you create weekly HIGH series, expanding it to daily interval using expandFirst will enable you to know on MONDAY what was the high for entire week.

For more information check Tutorial: Multiple time frame support

EXAMPLE

```
wc = TimeFrameCompress( Close, inWeekly );

/* now the time frame is still unchanged (say daily) and our MA will
operate on daily data */
dailyma = MA( C, 14 );

/* but if we call MA on compressed array, it will give MA from other
time frame */
weeklyma = MA( wc, 14 ); // note that argument is time-compressed
array

Plot( dailyma, "DailyMA", colorRed );

weeklyma = TimeFrameExpand( weeklyma, inWeekly ); // expand for
display

Plot( weeklyma, "WeeklyMA", colorBlue );
```

SEE ALSO TimeFrameSet() function , TimeFrameRestore() function

References:

The **TimeFrameExpand** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Alternative ZIG type function, multi TF
- Automatic Trendlines using multiple timeframes
- Caleb Lawrence
- Channel/S&R and trendlines
- Daily High Low in Advance
- EKEKO SAR-MF
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Triple Screen Trading System
- Harmonic Pattern Detection
- IFT of RSI - Multiple TimeFrames
- Index and ETF trading
- Intraday Range and Periods Framer
- Pivot Point with S/R Trendlines
- Rea Time Daily Price Levels
- TAPE READING
- Twiggs money flow weekly
- Volume Color with Dynamic Limit

More information:

See updated/extended version on-line.

TimeFrameGetPrice - retrieve O, H, L, C, V values from other time frame

Time Frame functions
(AmiBroker 4.50)

SYNTAX **TimeFrameGetPrice(*pricefield, interval, shift = 0, mode = expandFirst*)**

RETURNS ARRAY

FUNCTION The **TimeFrameGetPrice** - retrieves OHLCV fields from other time frames. This works immediatelly without need to call **TimeFrameSet** at all.

First parameter - *pricefield* - is one of the following: "O", "H", "L", "C", "V", "I" (open interest).

Interval is bar interval in seconds. You can use pre-defined interval constants: in1Minute, in5Minute, in15Minute, inHourly, inDaily, inWeekly, inMonthly. Or integer multiples like (3*in1Minute) for 3 minute bars

shift allows to reference past (negative values) and future (positive values) data in higher time frame. For example -1 gives previous bar's data (like in Ref function but this works in higher time frame).

mode - one of available modes:

- expandLast - the compressed value is expanded starting from last bar within given period (so for example weekly close/high/low is available on Friday's bar)
- expandFirst - the compressed value is expanded starting from first bar within given period (so for example weekly open is available from Monday's bar)
- expandPoint - the resulting array gets not empty values only for the last bar within given period (all remaining bars are Null (empty)).

Note these functions work like these 3 nested functions:

```
TimeFrameExpand( Ref( TimeFrameCompress( array, interval,
compress(depending on field used) ), shift ), interval, expandFirst
)
```

therefore, if shift = 0 compressed data may look into the future (weekly high can be known on monday). If you want to write a trading system using this function please make sure to reference PAST data by using negative shift value.

The only difference is that TimeFrameGetPrice is 2x faster than nested Expand/Compress.

For more information check Tutorial: Multiple time frame support

EXAMPLE

```
// Example 1. get previous week Open price
TimeFrameGetPrice( "O", inWeekly, -1 )

// Example 2. get weekly Close price 3 weeks ago
TimeFrameGetPrice( "C", inWeekly, -3 )

// Example 3. get weekly High price 2 weeks ago
TimeFrameGetPrice( "H", inWeekly, -2 )
```

```
// Example 4. get this week Open price.
TimeFrameGetPrice( "O", inWeekly, 0 )

// Example 5. get previous Day High when working on intraday data
TimeFrameGetPrice( "H", inDaily, -1 )
```

SEE ALSO TimeFrameSet() function

References:

The **TimeFrameGetPrice** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- BBAreacolor&TGLCROSSNEW
- For Auto Trading Setup
- Heatmap V1
- High Low Detection code
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- Market Profile
- Open Range Breakout Trading System
- Pivots for Intraday Forex Charts
- plot tomorrows pivots on an intraday database
- Prior Daily OHLC
- Robert Antony
- Vikram's Floor Pivot Intraday System
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

TimeFrameMode - switch time frame compression mode

Time Frame functions
(AmiBroker 4.80)

SYNTAX **TimeFrameMode(mode)**

RETURNS NOTHING

FUNCTION Switches time frame functions to different operating modes. Where mode is one of 0, 1, 2, 3, or 4.

- TimeFrameMode(0);
  - switches time frame functions to time-based operation (the default)
- TimeFrameMode(1);
  - switches time frame functions to N-tick operation (positive values passed to TimeFrameSet are treated now as N-tick)
- TimeFrameMode(2);
  - switches time frame functions to N-volume bar operation (positive values passed to TimeFrameSet are treated nowas N-volme bars)
- TimeFrameMode(3);
  - switches time frame functions to N-Range bar operation (positive values passed to TimeFrameSet are treated now as N-range bars) where N is expressed in DOLLARS (This mode is left in 5.14 and above for backward compatiblity only, see next)
- TimeFrameMode(4);
  - switches time frame functions to N-Range bar operation (positive values passed to TimeFrameSet are treated now as N-range bars) where N is expressed in TickSize units (this is default mode of operation of range bars in 5.14 and above).

Note: N-volume bars are very different from time-based bars(compression of data to N-volume bar may actually deliver MORE output bars - for example if one tick is 1000 shares and you have specified 100V bars then single tick will be expanded to TEN 100V bars - ten times original size)

TimeFrame functions are protected against array overrun and will not decompress beyond original array size (you will get an "Error 47. N-volume bar compressed data longer than base time frame"). Also switching main time frame to some weird N-volume bar value will result in limiting the output to maximum twice original data size(without error message).

You should keep that in mind and avoid using too small N-volume bar intervals that could lead to such condition. Also due to the nature of N-volume bars the only TimeFrameSet() function will yield correct N-volume bar values, TimeFrameGetPrice() may give slightly distorted results.

It is also possible to use n-volume bars in TimeFrame functions without calling TimeFrameMode() - it is then necessary to specify n-volume bars as negative number offset by -1000000 (minus one million):

TimeFrameSet(-1000000 - 2000);

EXAMPLE

```
TimeFrameMode( 2 );
TimeFrameSet( 50000 ); // 50'000 share bars..
//...do something ...
```

```
TimeFrameRestore();
```

SEE ALSO TimeFrameSet() function , TimeFrameRestore() function

References:

The **TimeFrameMode** function is used in the following formulas in AFL on-line library:

- Bad Tick Trim on 5 sec database

More information:

See updated/extended version on-line.

TimeFrameRestore
- restores price arrays to original time frame

Time Frame functions
(AmiBroker 4.50)

SYNTAX **TimeFrameRestore(tradeprices = False)**

RETURNS NOTHING

FUNCTION The **TimeFrameRestore** function restores price arrays replaced by **TimeFrameSet**.

Note that only OHLC, V, OI and Avg built-in variables are restored to original time frame when you call **TimeFrameRestore()**.

All other variables created when being in different time frame remain compressed.

To de-compress them to original interval you have to use **TimeFrameExpand**.

Tradeprice argument should be set to false.

EXAMPLE

```
TimeFrameSet( in5Minute ); // switch to 5 minute frame

/* MA now operates on 5 minute data, ma5_13 holds time-compressed 13
bar MA of 5min bars */

ma5_13 = MA( C, 13 );

TimeFrameRestore(); // restore time frame to original

TimeFrameSet( inHourly ); // switch now to hourly

mah_9 = EMA( C, 9 ); // 9 bar moving average from hourly data

TimeFrameRestore(); // restore time frame to original

Plot( Close, "Price", colorWhite, styleCandle );

// plot expanded average

Plot( TimeFrameExpand( ma5_13, in5Minute), "13 bar moving average
from 5 min bars", colorRed );
Plot( TimeFrameExpand( mah_9, inHourly), "9 bar moving average from
hourly bars", colorRed );
```

SEE ALSO TimeFrameSet() function

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2004-07-10 06:19:47	TimeFrameRestore and RestorePriceArrays is essentially the same function. So please note that calling TimeFrameRestore also resets the ticker set by eventual previous call to SetForeign()

References:

The **TimeFrameRestore** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Alternative ZIG type function, multi TF
- Automatic Trendlines using multiple timeframes
- Caleb Lawrence
- Channel/S&R and trendlines
- Daily High Low in Advance
- EKEKO SAR-MF
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Triple Screen Trading System
- Harmonic Pattern Detection
- Hull Multiple Moving Averages
- IFT of RSI - Multiple TimeFrames
- Improved NH-NH scan / indicator
- Lagging MA-Xover
- Market Breadth Chart-In-Chart
- Pivot Point with S/R Trendlines
- Price with Woodies Pivots
- Rea Time Daily Price Levels
- SUPER PIVOT POINTS
- Twiggs money flow weekly

More information:

See updated/extended version on-line.

TimeFrameSet
- switch price arrays to a different time frame

Time Frame functions
(AmiBroker 4.50)

SYNTAX **TimeFrameSet(*interval*)**

RETURNS NOTHING

FUNCTION The **TimeFrameSet** replaces current price/volume arrays: open, high, low, close, volume, openint, avg with time-compressed bars of specified interval once you switched to a different time frame all calculations and built-in indicators operate on selected time frame. To get back to original interval call **TimeFrameRestore()** function. Before calling **TimeFrameSet** again in the same formula with different interval you have to restore original time frame first using **TimeFrameRestore**.

interval defines time frame interval in seconds. So 60 means 1-minute. For the convenience the following interval constants are pre-defined:

- in1Minute = 60
- in5Minute = 5 * 60
- in15Minute = 15 * 60
- inHourly = 3600
- inDaily = 24 * 3600
- inWeekly = 5 * 24 * 3600 + 1 = 432001
- inMonthly = 25 * 24 * 3600 + 1 = 2160001
- inQuarterly (new in 5.20)
- inYearly (new in 5.20)

To get other intervals you can use multiple of pre-defined intervals, for example: (3*in1Minute) gives 3 minute bars. Or you can use 3 * inDaily for 3-day bars.

New in version 4.70 and above: You can also use NEGATIVE values for N-tick charts: -5 fives 5-tick chart. Note that N-tick compression works correct only if you have 1-tick base time interval selected in database settings.

You can also use TimeFrameSet to create N-volume bars as well as Range bars. See TimeFrameMode() function for more details.

VERY IMPORTANT:
inWeekly constant is now 432001 (5*inDaily + 1) - in previous version it was 432000
inMonthly constant is now 2160001 (25*inDaily + 1) - in previous version it was 2160000
It is changed because now N-day custom intervals are supported and they will interfere with weekly/monthly.
Note that 5*inDaily is now DIFFERENT than inWeekly. 5*inDaily creates 5-day bars that DO NOT necesarily cover Monday-Friday while inWeekly ALWAYS creates bars that begin on Monday and end on Friday. Also 25*inDaily creates 25-day bars that DO NOT necesarily represent full month, while inMonthly always begins with first day of the month and ends at the last day of the month

Once you switch the time frame using **TimeFrameSet** , all AFL functions operate on this time frame until you switch back the time frame to original interval using **TimeFrameRestore** or set to different interval again using **TimeFrameSet**. It is good idea to ALWAYS call

TimeFrameRestore when you are done with processing in other time frames.

When time frame is switched to other than original interval the results of all functions called since **TimeFrameSet** are time-compressed too. If you want to display them in original time frame you would need to 'expand' them as described later. Variables created and assigned before call to **TimeFrameSet()** remain in the time frame they were created. This behaviour allows mixing unlimited different time frames in single formula.

Please note that you can only compress data from shorter interval to longer interval. So when working with 1-minute data you can compress to 2, 3, 4, 5, 6, ...N-minute data. But when working with 15 minute data you can not get 1-minute data bars. In a similar way if you have only EOD data you can not access intraday time frames.

For more information check: Tutorial: Multiple time frame support in AFL

EXAMPLE

```
TimeFrameSet( in5Minute ); // switch to 5 minute frame

/* MA now operates on 5 minute data, ma5_13 holds time-compressed 13
bar MA of 5min bars */

ma5_13 = MA( C, 13 );

TimeFrameRestore(); // restore time frame to original

TimeFrameSet( inHourly ); // switch now to hourly

mah_9 = EMA( C, 9 ); // 9 bar moving average from hourly data

TimeFrameRestore(); // restore time frame to original

Plot( Close, "Price", colorWhite, styleCandle );

// plot expanded average

Plot( TimeFrameExpand( ma5_13, in5Minute), "13 bar moving average
from 5 min bars", colorRed );
Plot( TimeFrameExpand( mah_9, inHourly), "9 bar moving average from
hourly bars", colorRed );
```

SEE ALSO TimeFrameRestore() function , TimeFrameExpand() function , TimeFrameGetPrice() function

Comments:

Tomasz Janeczko
tj --at-- amibroker.com
2004-06-03 04:35:37
	TimeFrameSet(in15Minute);
	MA10_15Min=MA(Close,10);
	TimeFrameRestore();
	
	Buy=Cross(MA(Close,5), TimeFrameExpand(MA10_15Min, in15Minute));

References:

The **TimeFrameSet** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- Alternative ZIG type function, multi TF
- Automatic Trendlines using multiple timeframes
- Bad Tick Trim on 5 sec database
- Caleb Lawrence
- Channel/S&R and trendlines
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Daily High Low in Advance
- EKEKO SAR-MF
- Elder Impulse Indicator
- Elder Impulse Indicator V2
- Elder Triple Screen Trading System
- Harmonic Pattern Detection
- Hull Multiple Moving Averages
- IFT of RSI - Multiple TimeFrames
- Improved NH-NH scan / indicator
- Lagging MA-Xover
- Market Breadth Chart-In-Chart
- Pivot Point with S/R Trendlines
- Price with Woodies Pivots
- Rea Time Daily Price Levels
- SUPER PIVOT POINTS
- Twiggs money flow weekly

More information:

See updated/extended version on-line.

TimeNum - get current bar time

Date/Time
(AmiBroker 40)

SYNTAX	**TimeNum()**
RETURNS	ARRAY
FUNCTION	Returns the array with numbers that represent quotation time coded as follows: 10000 * hour + 100 * minute + second, so 12:37:15 becomes 123715
EXAMPLE	TimeNum()
SEE ALSO	Hour(), Minute(), Second(), TimeNum()

References:

The **TimeNum** function is used in the following formulas in AFL on-line library:

- Add SL/TGT other params to any strategy
- AFL_Glossary_1
- AR_Prediction.afl
- Auto Trade Step by Step
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Bad Tick Trim on 5 sec database
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- Continuous Contract Rollover
- Date_To_Num(), Time_To_Num()
- Export EOD or Intraday to .csv file
- For Auto Trading Setup
- Intraday Range and Periods Framer
- Lagging MA-Xover
- MFE and MAE and plot trades as indicator
- Moving Averages NoX
- Now Send Push Notifications From Amibroker
- Open Range Breakout Trading System
- Rea Time Daily Price Levels
- Reconnect TWS
- Renko Chart
- suresh
- Time Left in Bar
- Time Left to Current Bar
- Trigonometric Fit - TrigFit with AR for cos / sin
- TWS auto-export Executions-file parser
- Updated Renko Chart
- VWAP - Volume Weighted Average Price
- VWAP versus Average Price
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots

More information:

See updated/extended version on-line.

TrimResultRows
- trims Analysis result list to specified number of rows

Exploration / Indicators
(AmiBroker 6.90)

SYNTAX	**TrimResultRows(row_count)**
RETURNS	NOTHING
FUNCTION	Trims the Analysis result list to specified number of rows, positive row count means counting from top, negative from bottom
EXAMPLE	Filter = 1; AddColumn(C, "Price"); TrimResultRows(5);
SEE ALSO	

References:

The **TrimResultRows** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Trin
- traders (Arms) index

Composites
(AmiBroker 3.20)

SYNTAX **Trin()**

RETURNS ARRAY

FUNCTION Calculates TRIN (Arms Index) indicator.

NOTE: All built-in a/d indicators (AdLine/Trin) work only with composites calculated inside AmiBroker http://www.amibroker.com/newsletter/04-2000.html

If you are using QP2 database for example you should use QP2's own symbols for advances/declines.
!NY-A, !NY-D, !NY-AV, !NY-DV

The formula for NYSE TRIN using QP2 database is:

```
ArmsIndex = ( Foreign("!NY-A", "C") / Foreign("!NY-D", "C") ) / (
Foreign("!NY-AV", "C") / Foreign("!NY-DV","C" ) );
Plot( ArmsIndex, "TRIN", colorRed );
```

EXAMPLE trin()

SEE ALSO

References:

The **Trin** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

TRIX - triple exponential smoothed price

Indicators

SYNTAX	**trix(** ***periods*** **= 9)**
RETURNS	ARRAY
FUNCTION	Calculates the TRIX indicator (with averaging range of *periods*).
EXAMPLE	trix(12)
SEE ALSO	

References:

The **TRIX** function is used in the following formulas in AFL on-line library:

- channel indicator
- elliott wave manual labelling
- Gfx Toolkit
- Least Squares Channel Indicator
- Polyfit Lines
- Spearman Rank Correlation Coefficient
- TRIX
- Trix Bars number
- TRIXXX

More information:

See updated/extended version on-line.

Trough
- trough

Basic price pattern detection
(AmiBroker 3.10)

SYNTAX	**Trough(ARRAY, *change*, *n* = 1)**
RETURNS	ARRAY
FUNCTION	Gives the value of ARRAY *n*-th trough(s) ago. This uses the Zig Zag function (see Zig Zag) to determine the troughs. **Caveat:** this function is based on Zig-Zag indicator and may look into the future.
EXAMPLE	trough(close,5,1)
SEE ALSO	

Comments:

Tomasz Janeczko tj --at --- amibroker.com 2007-09-24 03:31:47	Zig/Peak/Trough functions work correctly for ARRAYS containing data greater than zero.

References:

The **Trough** function is used in the following formulas in AFL on-line library:

- Advanced Trend Lines with S & R
- Andrews Pitchfork
- Andrews PitchforkV3.3
- Automatic Trend-line
- Constant Trendline Plot
- Gartley 222 Pattern Indicator
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- MACD commentary
- Pivot Point with S/R Trendlines
- Schiff Lines
- Support Resistance levels
- Tom DeMark Trend Lines
- Wolfe Wave Patterns
- Woodie's CCI Panel Full Stats
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Retracements

More information:

See updated/extended version on-line.

TroughBars - bars since trough

Basic price pattern detection
(AmiBroker 3.10)

SYNTAX **TroughBars(ARRAY, *change*, *n* = 1)**

RETURNS ARRAY

FUNCTION Plots the number of bars that have passed from the *n*-th trough. This uses the Zig Zag function (see Zig Zag) to determine the troughs. **Caveat:** this function is based on Zig-Zag indicator and may look into the future.

EXAMPLE troughbars(close,5,1)

SEE ALSO

References:

The **TroughBars** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Advanced Trend Lines with S & R
- Constant Trendline Plot
- Fibonacci Internal and External Retracements
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gartley 222 Pattern Indicator
- Harmonic Patterns
- Head & Shoulders Pattern
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- Modified Head & Shoulder Pattern
- Pattern Recognition Exploration
- Pivot Point with S/R Trendlines
- QP2 Float Analysis
- RSI Trendlines and Wedges
- Stochastics Trendlines
- The Fibonaccian behavior
- Tom DeMark Trend Lines
- Wolfe Wave Patterns
- Woodie's CCI Panel Full Stats
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount
- ZigZag Retracements

More information:

See updated/extended version on-line.

TSF - time series forecast

Statistical functions
(AmiBroker 4.20)

SYNTAX	**TSF(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates time series forecast indicator (similar to LinearReg but differs by the value of lin reg slope) The function accepts periods parameter that can be constant as well as time-variant (array).

EXAMPLE	Plot(Close, "Price", colorBlue, styleCandle);
	Plot(TSF(close,5), "Time Series Forecast", colorRed);

SEE ALSO	

Comments:

Nigel Rowe
 	
2003-04-30
06:03:00
	TSF is exactly the estimate of LinearReg for the NEXT DAY.
	
	(it is calculated as LinearReg PLUS LinRegSlope * 1 (bar))
	
	Plot(LinearReg(Close, 10)+LinRegSlope(Close, 10), "Forecast for tommorrow", colorRed);
	
	Plot(TSF(Close, 10), "Forecast for tommorrow 2", colorBlue);

References:

The **TSF** function is used in the following formulas in AFL on-line library:

- Advanced MA system
- DEBAJ
- Moving Trend Bands (MTB)

More information:

See updated/extended version on-line.

Ultimate - ultimate oscillator

Indicators

SYNTAX **Ultimate(** ***fast*** **= 7,** ***med*** **= 14,** ***slow*** **= 28)**

RETURNS ARRAY

FUNCTION Calculates the Ultimate Oscillator indicator using the three cycle lengths supplied as parameters. Note that each of the three parameters must be greater than the preceding parameter.

EXAMPLE The formula "ultimate(7, 14, 21)" returns the default Ultimate Oscillator.

SEE ALSO

References:

The **Ultimate** function is used in the following formulas in AFL on-line library:

- Adaptave Zones O/B & O/S Oscillator
- Ultimate plus
- Varexlist

More information:

See updated/extended version on-line.

UncIssues
- unchanged issues

Composites
(AmiBroker 3.20)

SYNTAX	**UncIssues()**
RETURNS	ARRAY
FUNCTION	Returns the number of unchanged issues for a given market (the one that currently analysed stock belongs to)
EXAMPLE	uncissues()
SEE ALSO	

References:

The **UncIssues** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

UncVolume
- unchaged issues volume

Composites
(AmiBroker 3.20)

SYNTAX	**UncVolume()**
RETURNS	ARRAY
FUNCTION	Returns the volume of unchanged issues for a given market (the one that currently analysed stock belongs to)
EXAMPLE	uncvolume()
SEE ALSO	

References:

The **UncVolume** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

ValueWhen
- get value of the array when condition met

Trading system toolbox
(AmiBroker 3.10)

SYNTAX	**ValueWhen(EXPRESSION, ARRAY, *n* = 1)**
RETURNS	ARRAY
FUNCTION	Returns the value of the ARRAY when the EXPRESSION was true on the *n* -th most recent occurrence. Note: this function allows also 0 and negative values for *n* - this enables referencing future
EXAMPLE	valuewhen(cross(close, ma(close,5)) ,macd(), 1)
SEE ALSO	

References:

The **ValueWhen** function is used in the following formulas in AFL on-line library:

- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Andrews PitchforkV3.3
- AR_Prediction.afl
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trend-line
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- babaloo chapora
- Baseline Relative Performance Watchlist charts V2
- Basket Trading System T101
- BEANS-Summary of Holdings
- Bollinger band normalization
- Bullish Percent Index 2004
- Caleb Lawrence
- Candle Stick Analysis
- Candle Stick Demo
- CCI(20) Divergence Indicator
- Chandelier Exit
- Chandelier Exit
- Channel/S&R and trendlines
- Chart Zoom
- Cole
- Colorfull Price
- Congestions detection
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Darvas Amibroker

- Date_To_Num(), Time_To_Num()
- Divergence indicator
- Divergences
- Double top detection
- Fast Refreshed KAGI Swing Charts (Price Swing)
- Fib Fan Based on ZZ
- Fibonacci Internal and External Retracements
- FirstBarIndex(), LastBarIndex()
- Fund Screener
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gann HiLo Indicator and System
- Gann Swing Chart
- Gann Swing chart v41212
- Gann Swing Charts in 3 modes with text
- Graphical sector analysis
- Graphical sector stock amalysis
- Halftrend
- Harmonic Pattern Detection
- Head & Shoulders Pattern
- HH-LL-PriceBar
- High Low Detection code
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Last Five Trades Result Dashboard – AFL code
- MACD commentary
- Market Profile
- MFE and MAE and plot trades as indicator
- mitalpradip
- Modified Head & Shoulder Pattern
- Monthly bar chart
- MS Darvas Box with Exploration
- Nadaraya-Watson Envelope
- Now Send Push Notifications From Amibroker
- Open Range Breakout Trading System
- pattenz
- Pattern Recognition Exploration
- Peterson
- PF Chart - Close - April 2004
- Pivot Finder
- Prashanth
- Prior Daily OHLC
- Pullback System No. 1
- Rainbow Oscillator
- Range Filter - Trading Strategy
- Relative Strength Multichart of up to 10 tickers
- RSI of Weekly Price Array
- RSI Trendlines and Wedges
- Schiff Lines
- shailu lunia

- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Stochastic Divergence, negative
- Stochastic Divergence, positive
- Stochastic Divergences, PDI, NDI
- Stochastic of Weekly Price Array
- Stochastics Trendlines
- Super Trend Indicator
- Support and Resistance
- Support and resistance
- Support Resistance levels
- TAPE READING
- TD sequential
- TD Sequential
- The Fibonaccian behavior
- Time Frame Weekly Bars
- Trend Detection
- Trend exploration with multiple timeframes
- Trend Exploration: Slope Moving Average
- Trigonometric Fit - TrigFit with AR for cos / sin
- Using From and To dates from Auto Analysis in Code
- Visible Min and Max Value Demo
- Visualization of stoploses and profit in chart
- Volatility Quality Index
- VWAP - Volume Weighted Average Price
- Weekly chart
- Weekly Trend in Daily Graph
- White Theme
- William's Alligator System II
- Williams Alligator system
- Zig Zag Indicator with Valid Entry and Exit Points
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

VarGet - gets the value of dynamic variable

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX **VarGet("varname")**

RETURNS ARRAY or NUMBER

FUNCTION Gets the value of dynamic variable.
Returns the NUMBER or ARRAY depending on type of underlying variable.

Dynamic variables are variables that are named dynamically, typically by creating a variable name from a static part and a variable part. For example, the following example dynamically constructs the variable name from a variable prefix and a static suffix. Dynamic variables are always global.

EXAMPLE

```
for( i = 1; i < 10; i++ )
{
Plot( VarGet( "C"+i ), "C"+i, colorRed );
}
```

SEE ALSO VarSet() function , VarGetText() function , VarSetText() function

References:

The **VarGet** function is used in the following formulas in AFL on-line library:

- 3 ways to use RMI in one script
- A simple AFL Revision Control System
- AFL Timing functions
- AFL_Glossary_Converter
- AllinOneAlerts - Module
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Basket Trading System T101
- Button trading using AB auto trading interface
- Candle Identification
- channel indicator
- Continuous Contract Rollover
- Coral Trend Indicator
- Cycle Period
- Detailed Equity Curve
- elliott wave manual labelling
- Fast Refreshed KAGI Swing Charts (Price Swing)
- For Auto Trading Setup
- Gfx Toolkit
- Heatmap V1
- Herman
- interactively test discretionary trading
- Intraday Average Volume
- Intraday Volume EMA
- Least Squares Channel Indicator

- Manual Bracket Order Trader
- Non-repaitning Zigzag line
- Now Send Push Notifications From Amibroker
- Polyfit Lines
- Profit Table (Color Coded)
- Ranking Ticker WatchList
- Rebalancing Backtest avoiding leverage
- Reconnect TWS
- suresh
- TAPE READING
- TWS auto-export Executions-file parser
- Visi-Trade
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

VarGetText - gets the text value of dynamic variable

Miscellaneous functions
(AmiBroker 4.80)

SYNTAX **VarGetText("varname")**

RETURNS STRING

FUNCTION Gets the text (string) value of dynamic variable.
Similar to VarGet but always returns always string values (if underlying variable has different type it is converted to string) Allows for example appending to text variable no matter if it is defined earlier or not as shown in the example below

Dynamic variables are variables that are named dynamically, typically by creating a variable name from a static part and a variable part. For example, the following example dynamically constructs the variable name from a variable prefix and a static suffix. Dynamic variables are always global.

EXAMPLE

```
Title = VarGetText("Title") + "something";
// above will work correctly regardless of whenever title was
defined earlier or not
```

SEE ALSO VarGet() function , VarSet() function

References:

The **VarGetText** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_Converter
- Auto Trade Step by Step
- Automatic Linear Trend Channel
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- BEANS-Summary of Holdings
- Button trading using AB auto trading interface
- Calculate composites for tickers in list files
- channel indicator
- Continuous Contract Rollover
- DateNum_DateStr
- elliott wave manual labelling
- Heatmap V1
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Ranking and sorting stocks
- suresh
- TWS auto-export Executions-file parser
- Visi-Trade
- Volume Divided Histogram
- VolumeDivAvgV3_Study_BrS
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

VarSet - sets the value of dynamic variable

Miscellaneous functions
(AmiBroker 4.60)

SYNTAX **VarSet("varname", value)**

RETURNS NUMBER

FUNCTION Sets the value of dynamic variable. Returns 1 on success, 0 on failure.

Dynamic variables are variables that are named dynamically, typically by creating a variable name from a static part and a variable part. The following example dynamically constructs the variable name from a variable prefix and a static suffix. Dynamic variables are always global. Starting from version 6.10 the function accept matrix variables in addition to numbers and arrays.

EXAMPLE

```
for( i = 1; i < 10; i++ )
{
VarSet( "C"+i, Ref( C, -i ) );
}

// creates variables C1, C2, C3, C4, ...., C10 equal to Ref( C, -1
), Ref( C, -2 ),   ..., Ref( C, -10 )
// respectively
```

SEE ALSO VarGet() function , VarGetText() function , VarSetText() function

References:

The **VarSet** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3 ways to use RMI in one script
- AFL Timing functions
- AFL_Glossary_Converter
- AllinOneAlerts - Module
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Basket Trading System T101
- Button trading using AB auto trading interface
- Candle Identification
- channel indicator
- Continuous Contract Rollover
- Coral Trend Indicator
- Cycle Period
- Detailed Equity Curve
- elliott wave manual labelling
- Fast Refreshed KAGI Swing Charts (Price Swing)
- For Auto Trading Setup
- Gfx Toolkit
- Harmonic Pattern Detection
- Heatmap V1

- Herman
- interactively test discretionary trading
- Intraday Average Volume
- Intraday Volume EMA
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Non-repaitning Zigzag line
- Now Send Push Notifications From Amibroker
- Polyfit Lines
- Profit Table (Color Coded)
- Ranking and sorting stocks
- Ranking Ticker WatchList
- Rebalancing Backtest avoiding leverage
- Reconnect TWS
- Stress with SuperSmoother
- Support and resistance
- suresh
- TAPE READING
- TWS auto-export Executions-file parser
- Visi-Trade
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

VarSetText - sets dynamic variable of string type

Miscellaneous functions
(AmiBroker 4.80)

SYNTAX	**VarSetText("varname", "valuetext")**
RETURNS	STRING
FUNCTION	Sets the text (string) value of dynamic variable. Similar to VarSet but allows to assign string (text) instead of number/array. *Dynamic variables* are variables that are named dynamically, typically by creating a variable name from a static part and a variable part. For example, the following example dynamically constructs the variable name from a variable prefix and a static suffix. Dynamic variables are always global.
EXAMPLE	
SEE ALSO	VarGetText() function , VarGet() function , VarSet() function

References:

The **VarSetText** function is used in the following formulas in AFL on-line library:

- AFL_Glossary_Converter
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Button trading using AB auto trading interface
- Calculate composites for tickers in list files
- channel indicator
- Continuous Contract Rollover
- DateNum_DateStr
- elliott wave manual labelling
- Heatmap V1
- Least Squares Channel Indicator
- Manual Bracket Order Trader
- Ranking and sorting stocks
- Say Notes
- suresh
- TWS auto-export Executions-file parser
- Visi-Trade
- Woodie's CCI Panel Full Stats

More information:

See updated/extended version on-line.

Version
- get version info

Miscellaneous functions
(AmiBroker 3.90)

SYNTAX **Version(*minrequired* = 0)**

RETURNS NUMBER

FUNCTION Returns the AmiBroker version number as float (3.90 for example). Additionally when you specify Version(4.0) AmiBroker will issue an error message when running the formula on AB earlier than 4.0 :)

EXAMPLE version(3.90);

SEE ALSO

References:

The **Version** function is used in the following formulas in AFL on-line library:

- 2 Timeframes Candlestick Bar Chart
- 3TF Candlestick Bar Chart
- AllinOneAlerts - Module
- Automatic Trendlines using multiple timeframes
- Bad Tick Trim on 5 sec database
- channel indicator
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- elliott wave manual labelling
- Gann Swing Charts in 3 modes with text
- Get Moneycontrol News Snippets into Amiboker
- GFX ToolTip
- Harmonic Pattern Detection
- High Low Detection code
- JEEVAN'S SRI CHAKRA
- Least Squares Channel Indicator
- P&F Chart - High/Low prices Sept2003
- PF Chart - Close - April 2004
- Revised Renko chart
- RSI + Avgs
- RSI indicator with Upper & Lower Zone Bars
- Send Alerts from Amibroker to Telgram
- Visi-Trade
- WILSON RELATIVE PRICE CHANNEL
- Wolfe Wave Patterns

More information:

See updated/extended version on-line.

VoiceCount - get number of SAPI voices

Text-to-Speech functions
(AmiBroker 6.20)

SYNTAX	**VoiceCount()**
RETURNS	NUMBER
FUNCTION	The function returns number of available SAPI (Speech API) voices - to be used with Say() command.
EXAMPLE	
SEE ALSO	

References:

The **VoiceCount** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

VoiceSelect - select SAPI voice

Text-to-Speech functions
(AmiBroker 6.20)

SYNTAX	**VoiceSelect(num)**
RETURNS	NOTHING
FUNCTION	The function selects SAPI (Speech API) voice to be used with Say() function, num is a numerical index of installed voices. By default Windows comes with just one voice, but one can add countless 3rd party voices.
EXAMPLE	
SEE ALSO	VoiceCount() function , Say() function

References:

The **VoiceSelect** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

VoiceSetRate - sets voice speech rate

Text-to-Speech functions
(AmiBroker 6.30)

SYNTAX	**VoiceSetRate(rate)**
RETURNS	NOTHING
FUNCTION	The function sets SAPI voice (speech synthesis) rate, i.e. how fast words are spoken. Rate of 0 (zero) is "normal", negative is slower, positive is faster (allowable range -10..+10)
EXAMPLE	
SEE ALSO	Say() function , VoiceCount() function , VoiceSelect() function , VoiceSetRate() function , VoiceSetVolume() function

References:

The **VoiceSetRate** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

VoiceSetVolume - set the volume of speech

Text-to-Speech functions
(AmiBroker 6.30)

SYNTAX	**VoiceSetVolume(volume)**
RETURNS	NOTHING
FUNCTION	The function sets SAPI voice (speech synthesis) volume (0..100)
EXAMPLE	
SEE ALSO	Say() function , VoiceCount() function , VoiceSelect() function , VoiceSetRate() function

References:

The **VoiceSetVolume** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

VoiceWaitUntilDone
- waits until TTS voice has finished speaking

Text-to-Speech functions
(AmiBroker 6.30)

SYNTAX **VoiceWaitUntilDone(timeout)**

RETURNS NUMBER

FUNCTION Waits until voice has finished speaking or specified timeout (1..100ms) has elapsed. Returns True (1) if voice finished, False (0) if the timeout elapsed.

Parameters:

- timeout - time in milliseconds on how long to wait for speech synthesizer to finish

EXAMPLE

SEE ALSO Say() function , VoiceCount() function , VoiceSelect() function , VoiceSetRate() function , VoiceSetVolume() function

References:

The **VoiceWaitUntilDone** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Wilders
- Wilder's smoothing

Moving averages, summation
(AmiBroker 3.40)

SYNTAX	**Wilders(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates Wilder's average of the ARRAY using *periods* averaging range
EXAMPLE	wilders(close, 10);
SEE ALSO	

References:

The **Wilders** function is used in the following formulas in AFL on-line library:

- AC+ acceleration
- Adaptave Zones O/B & O/S Oscillator
- Advanced MA system
- Analytic RSI formula
- AO+ Momentum indicator
- Average Price Crossover
- babaloo chapora
- Bollinger Fibonacci Bands
- DEBAJ
- DMI Spread Index
- Fund Screener
- garythompson
- garythompson
- Raw ADX
- RSI of Weekly Price Array
- The Mean RSIt
- The Mean RSIt (variations)
- Twiggs Money Flow
- Twiggs money flow weekly
- Volume Occilator
- Volume Oscillator
- William's Alligator System II
- Williams Alligator system

More information:

See updated/extended version on-line.

WMA
- weighted moving average

Moving averages, summation
(AmiBroker 40)

SYNTAX	**wma(ARRAY, *periods*)**
RETURNS	ARRAY
FUNCTION	Calculates weighted average. 5 day weighted average gives weight of 5 to the most recent quote, 4 to the previous quote, downto 1 for the 5-bar back quote. The function accepts time-variable *periods.*
EXAMPLE	WMA(Close, 5)
SEE ALSO	MA(), EMA(), WMA(), DEMA()

References:

The **WMA** function is used in the following formulas in AFL on-line library:

- Adverse Move Ratio
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- babaloo chapora
- Bman's HaDiffCO
- Chande's Trend Score
- Chandelier Exit or Advanced Trailing Stop
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- DEBAJ
- Dominant Cycle Phase
- Gabriel Linear Regression Angle Indicator
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- Hull Moving Average
- John Ehler
- MAVG
- MultiCycle 1.0
- Nadaraya-Watson Envelope
- Neural Network Powered Smooth/Predictive RSI V2
- Parametric Chande Trendscore
- Relative strength comparison with moving average
- Signal to Noise
- Stochastic RSI
- Volume Occilator
- Volume Spread for VSA

More information:

See updated/extended version on-line.

WriteIf - commentary conditional text output

Exploration / Indicators

SYNTAX	**WriteIf(EXPRESSION, "TRUE TEXT", "FALSE TEXT")**
RETURNS	STRING
FUNCTION	If EXPRESSION evaluates to "true", then the TRUE TEXT string is displayed within the commentary. If EXPRESSION evaluates to "false", then the FALSE TEXT string is displayed.
EXAMPLE	writeif(c > mov(c,200,s), "The close is above the 200-period moving average.","The close is below the 200-period moving average.")
SEE ALSO	

Comments:

Tomasz Janeczko
tj --at-- amibroker.com
2004-06-12 05:56:01
	WriteIf in fact does not "write" anything. The name is misleading but it is left
	for easy translation of MS formulas to AmiBroker.
	WriteIf is just "TextIIF" it RETURNS string value depending on condition.
	
	In commentary window, statements evaluating to STRINGS on global level
	are displayed in the output window. However if you do the same inside the
	FUNCTION
	it is no longer in global level (it is on LOCAL, FUNCTION level).
	
	To display actual string in this case use PRINTF function:
	http://www.amibroker.com/f?printf
	
	function comment(indicator)
	{
	printf("\nComment...\n");
	
	printf(WriteIf(1, "TrueText", "FalseText"));
	printf(WriteVal(indicator) + "\n");
	}

Tomasz Janeczko 2005-08-10 06:37:55	Please note that WriteIf returns just single string representing current SelectedValue of the EXPRESSION

References:

The **WriteIf** function is used in the following formulas in AFL on-line library:

- 3 Price Break
- Adaptave Zones O/B & O/S Oscillator
- Alphatrend
- AR_Prediction.afl
- Auto Trade Step by Step
- AutoTrade using an Exploration

- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- BMTRIX Intermediate Term Market Trend Indicator
- CandleStick Comentary--Help needed
- Candlestick Commentary
- Candlestick Commentary Modified
- Candlestick Commentary-modified
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI(20) Divergence Indicator
- Color Display.afl
- Commodity Channel Index
- Coppock Trade Signal on Price Chart
- COT REPORT
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Darvas Amibroker
- Double top detection
- ekeko price chart
- Elder Impulse Indicator V2
- For Auto Trading Setup
- Fre
- Geometric Mean of Volume
- Gfx Toolkit
- Graphical sector analysis
- Graphical sector stock amalysis
- Halftrend
- High Low Detection code
- How to add IB Option Symbols
- IBD relative strength database Viewer
- ICHIMOKU SIGNAL TRADER
- Improved NH-NH scan / indicator
- Intraday Fibonacii Trend Break System
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- Linear Candle
- MA Difference 20 Period
- MACD BB Indicator
- MACD commentary
- Main price chart with Rainbow & SAR
- Market Facilitation Index VS Volume
- mitalpradip
- MS Darvas Box with Exploration
- New HL Scanner
- NRx Exploration
- Open Range Breakout Trading System
- Performance Check
- PF Chart - Close - April 2004
- Polarized Fractal Efficiency
- prakash
- R-Squared
- Range Filter - Trading Strategy

- Relative Strength Index
- Relative Strength Multichart of up to 10 tickers
- ROC of MACD Weekly
- RSI of Weekly Price Array
- Scan New High and New Low
- Square of Nine Roadmap Charts
- STD_STK Multi
- Stochastics Trendlines
- StochD_StochK Single.afl
- Stress with SuperSmoother
- Sun&Cloud
- Super Trend Indicator
- TD Sequential
- Time Left in Bar
- Time Left to Current Bar
- Trend Analysis_Comentary
- Trend exploration with multiple timeframes
- Trend Exploration: Slope Moving Average
- Triangular Moving Average
- Triangular Moving Average new
- TSV
- Vertical Horizontal Filter (VHF)
- Volume Occilator
- Volume Spread for VSA
- Weekly Trend in Daily Graph
- White Theme
- William's Alligator System II
- Williams Alligator system
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Price Panel With Woodie's Pivots
- Zig Zag Indicator with Valid Entry and Exit Points
- ZigZag Retracements

More information:

See updated/extended version on-line.

WriteVal
- converts number to string

Exploration / Indicators

SYNTAX **WriteVal(NUMBER, *format* = 1.3, *separator*=True, roundAndPad = False)**
WriteVal(ARRAY, *format* = 1.3, *separator*=True, roundAndPad = False)

RETURNS STRING

FUNCTION THIS FUNCTION IS OBSOLETE. Use functionally identical NumToStr() instead.

It is used to convert numeric value of NUMBER or ARRAY to string. It does NOT write anything, it simply returns a string that can be used in other operations or as a text column in exploration or displayed using printf().

The second parameter - *format* - allows you to control output formatting (decimal places and leading spaces). The integer part of the number controls minimum number of characters used to display the number (if you specify high number the output will be space-padded). The fractional part defines how many decimal places to display, for example 1.0 - will give you a number without fractional part at all, and 1.2 - will give two digits past the decimal point

There is also a special format constant formatDateTime that allows to convert date/time returned by DateTime() function formatted according to Windows regional settings.

Third parameter *separator* (true by default) controls if thousand separator is added or not. Thousands separator is definable in Tools->Preferences->Misc.

Fourth parameter *roundAndPad* controls whenever function rounds output beyond 7th significant digit (and pads the rest with zeros), By default rounding is OFF now because it was off in 5.90 and earlier and rounding introduced in 5.91 could confuse old time users

Note: **NumToStr** is a synonym for **WriteVal** function and **NumToStr** is preferred in new coding.

EXAMPLE printf(WriteVal(Close, 1.3, True));

For more examples see NumToStr function

SEE ALSO WRITEIF() function , DATETIME() function , NumToStr() function

Comments:

Tomasz Janeczko tj --at-- amibroker.com 2004-06-12 05:53:06	WriteVal always returns *one* value of the array (not arrays of values). In almost all cases this is LastValue of the array but in indicators it is "selected value" - the one that is selected by the vertical line.
Tomasz Janeczko tj --at-- amibroker.com 2004-06-12 05:54:45	The name WriteVal() is here because people coming from Metastock wanted easy translation from MS that has WriteVal function too. Better name for it is Num2Str.

References:

The **WriteVal** function is used in the following formulas in AFL on-line library:

- Abhimanyu
- AccuTrack
- Advisory NRx price chart display.
- ADX Indicator - Colored
- AFL Example
- AFL Example - Enhanced
- AFL_Glossary_1
- Alert Output As Quick Rewiev
- Alpha and Beta and R_Squared Indicator
- Aroon Indicators
- AR_Prediction.afl
- babaloo chapora
- Baseline Relative Performance Watchlist charts V2
- BB squeeze
- BBAreacolor&TGLCROSSNEW
- Bollinger - Keltner Bands
- Bull Fear / Bear Fear
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- CandleStick Comentary--Help needed
- CCI(20) Divergence Indicator
- Chande Momentum Oscillator
- Cole
- Color Coded Short Term Reversal Signals
- colored CCI
- COMBO
- Commodity Channel Index
- Coppock Trade Signal on Price Chart
- Count Tickers in Watchlist
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Darvas Amibroker
- Darvas Wisestocktrader
- Day Bar No
- De Mark's Range Projection
- Dinapoli Guru Commentary
- DiNapolis 3x Displaced Moving Averages
- Elder safe Zone Long + short
- Elder's SafeZone Stop
- ElderSafeZoneStopLong
- ElderSafeZoneStopShort
- Expiry day/days - Last thursday of month
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- Fre
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Futures - Dollar Move Indicator

- Futures - Dollar Move Today Indicator
- Gann level plotter
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Halftrend
- hassan
- ICHIMOKU SIGNAL TRADER
- Improved NH-NH scan / indicator
- Intraday Fibonacii Trend Break System
- IntraDay Open Marker
- Intraday Trend Break System
- JEEVAN'S SRI CHAKRA
- Larry William's Volatility Channels
- Last Five Trades Result Dashboard – AFL code
- Linear Candle
- MA Difference 20 Period
- MACD BB Indicator
- MACD commentary
- MACD Histogram - Change in Direction
- MACD indicator display
- Main price chart with Rainbow & SAR
- Meu Sistema de Trading - versão 1.0
- MFE and MAE and plot trades as indicator
- mitalpradip
- Monthly bar chart
- Monthly Coppock Guide
- MS Darvas Box with Exploration
- Multiple sinus noised
- N-period candlesticks (time compression)
- Option Calls, Puts and days till third friday.
- P&F chart with range box sizes
- Parabolic SAR in JScript
- Parabolic SAR in VBScript
- Performance Check
- Peterson
- PF Chart - Close - April 2004
- Pivot End Of Day Trading System
- Pivot Finder
- Pivot Point and Support and Resistance Points
- plot tomorrows pivots on an intraday database
- Point & figure Chart India Securities
- Position Sizing and Risk Price Graph
- Position Sizing and Risk Price Graph - 2
- prakash
- Prashanth
- Probability Calculator
- QP2 Float Analysis
- Rainbow Charts
- Rainbow Oscillator
- Range Filter - Trading Strategy
- Rea Time Daily Price Levels

- Relative Momentum Index (RMI)
- Relative Strength Index
- Relative Strength Multichart of up to 10 tickers
- ROC of MACD Weekly
- RSI of Weekly Price Array
- RSIS
- shailu lunia
- Shares To Buy Price Graph
- STD_STK Multi
- Steve Woods' Float Channel Lines
- Stochastics Trendlines
- StochD_StochK Single.afl
- Super Trend Indicator
- Support Resistance levels
- TD Sequential
- The D_oscillator
- tomy_frenchy
- Trend Analysis_Comentary
- Triangular Moving Average
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- TRIX
- TSV
- Tushar Chande's Projected Range
- Using From and To dates from Auto Analysis in Code
- ValueChart
- Volume Occilator
- Volume Oscillator
- Volume Spread for VSA
- Weinberg's The Range Indicator
- White Theme
- William's Alligator System II
- Williams Alligator system
- WILSON RELATIVE PRICE CHANNEL
- Zig-Hi Zap-Lo

More information:

See updated/extended version on-line.

XYChartAddPoint
- add point to exploration scatter (XY) chart

Exploration / Indicators
(AmiBroker 5.60)

SYNTAX **XYChartAddPoint("chartname", "text", x, y, color, linecolor = colorDefault)**

RETURNS NUMBER

FUNCTION The function adds a data point with specified x,y co-ordinates, text description and point color to the exploration (XY) scatter chart defined by "chartname" parameter.

If chart with given name does not exists already it creates a scatter chart tab in the New Analysis Exploration output window.

Returns 1 (true) on success (when point was added), or 0 (false) on failure - for example when x or y arguments were Null

More on XY charts in explorations can be read in the Tutorial: How to create your own exploration

EXAMPLE

```
// XY chart coding example
// This formula generates 2 X-Y scatter charts that display
relationship between
// final trade profit and MFE and MAE

Buy = Cross( MACD(), Signal());
Sell = Cross( Signal(), MACD() );
Short = False;
Cover = False;

Eq = Equity( 1 ); // single-security equity  this evaluates stops
(if you use them) and removes extra signals

Entry = Buy OR Short;
EqAtEntry = ValueWhen( Entry, Eq );
Profit = 100 * ( Eq - EqAtEntry ) / EqAtEntry; // percent profit

// MAE and MFE below use CLOSING equity, MAE and MFE in the report
use high/low price
MAE = 100 * ( LowestSince( Entry, Eq ) - EqAtEntry ) / EqAtEntry; //
percent MAE
MFE = 100 * ( HighestSince( Entry, Eq ) - EqAtEntry ) / EqAtEntry;
// percent MAE

bi = BarIndex();

Len = bi - ValueWhen( Entry, bi );

EntryPrice = ValueWhen( Entry, BuyPrice );
// if you prefer MAE/MFE using high/low price
// uncomment the lines below (long only Version)
MAE = 100 * ( LowestSince( Entry, Low ) - EntryPrice ) / EntryPrice
```

```
; // percent MAE using low
MFE = 100 * ( HighestSince( Entry, High ) - EntryPrice ) /
EntryPrice ; // percent MAE using high

Exit = Sell OR Cover;
dt = DateTime();
Clr = ColorHSB( Status("stocknum"), 255, 255 );
for( bar = 0; bar < BarCount; bar++ )
{
    if( Exit[ bar ] )
     {
       // item text consists of two parts
       // first part (before t) is a item name displayed
immediatelly on XY chart
       // second part (after t) is a tooltip hint text that is
displayed in the tooltip
       // when you hover on given item
       // here we will only use hint text
        HintText = "t" + Name()+"@"+ DateTimeToStr( dt[ bar ] );

        XYChartAddPoint( "Profit vs MAE", HintText, MAE[ bar ],
Profit[ bar ],Clr );
        XYChartAddPoint( "Profit vs MFE", HintText, MFE[ bar ],
Profit[ bar ], Clr );
        XYChartAddPoint( "Profit vs trade length", HintText, Len[
bar ], Profit[ bar ], Clr );
     }
}

XYChartSetAxis( "Profit vs MAE", "[MAE]", "[Profit]" );
XYChartSetAxis( "Profit vs MFE", "[MFE]", "[Profit]" );
XYChartSetAxis( "Profit vs trade length", "[Length]", "[Profit]"  );


Filter = Exit;
AddColumn( Eq, "Equity" );
AddColumn( Profit, "Profit" );
AddColumn( MAE, "MAE" );
AddColumn( MFE, "MFE" );
AddColumn( Len, "trade length" );
```

SEE ALSO XYChartSetAxis() function

References:

The **XYChartAddPoint** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

XYChartSetAxis
- set the names of X and Y axes in exploration scatter charts

Exploration / Indicators
(AmiBroker 5.60)

SYNTAX **XYChartSetAxis("chartname", "x-axis name", "y-axis name", style = 0)**

RETURNS NOTHING

FUNCTION The function sets the name of X and Y axis for the Exploration scatter (XY) charts.

Since version 6.0 it allows to define chart style. Supported styles are styleLine, styleDots, styleHistogram, styleThick and combinations of those styles.

EXAMPLE

```
chartname="example";

XYChartSetAxis(chartname, "[x]", "[sinx/x]", styleLine | styleDots
); // bar style
for( x = -10; x < 10; x += 0.2 )
{
   y = sin(x ) / x;
   XYChartAddPoint( chartname, "", x, y, colorGreen, colorRed );
}

XYChartAddPoint( chartname, "", Null, Null ); // add a NULL point to
begin new line

for( x = -10; x < 10; x += 0.2 )
{
   y = sin( 2 * x ) / x;
   XYChartAddPoint( chartname, "", x, y, colorOrange, colorBlue );
}
```

SEE ALSO XYChartAddPoint() function

References:

The **XYChartSetAxis** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

Year - year

Date/Time
(AmiBroker 3.40)

SYNTAX	**Year()**
RETURNS	ARRAY
FUNCTION	Returns the array with years (full four digits 1900-....)
EXAMPLE	writeval(year());
SEE ALSO	

References:

The **Year** function is used in the following formulas in AFL on-line library:

- Auto Trade Step by Step
- Backup Data of 1min Interval
- Binance Data Download
- Coinbase Data Download
- Days to Third Friday
- End Of Year Trading
- Expiry day/days - Last thursday of month
- Expiry Thursday for Indian markets
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- For Auto Trading Setup
- FTX Data Download
- High Low Detection code
- How to add IB Option Symbols
- Luna Phase
- Lunar Phases - original
- LunarPhase
- Monthly bar chart
- N-period candlesticks (time compression)
- New HL Scanner
- Next Date Format
- Periodically ReBalance a BUY & HOLD Portfolio
- Prashanth
- Profit Table (Color Coded)
- Relative Strength Multichart of up to 10 tickers
- White Theme

More information:

See updated/extended version on-line.

ZIG
- zig-zag indicator

Basic price pattern detection
(AmiBroker 3.10)

SYNTAX	**zig(ARRAY, *change*)**
RETURNS	ARRAY
FUNCTION	Calculates the minimum % *change* Zig Zag indicator. **Caveat:** this function is based on Zig-Zag indicator and may look into the future - this means that you can get unrealistic results when back testing trading system using this indicator. This function is provided rather for pattern and trend recognition formulas.
EXAMPLE	zig(close,5)
SEE ALSO	

References:

The **ZIG** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Andrews PitchforkV3.3
- Bollinger band normalization
- Divergence indicator
- Fib Fan Based on ZZ
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Heatmap V1
- Indicator Explorer (ZigZag)
- Multiple Ribbon Demo
- Ord Volume
- QP2 Float Analysis
- Schiff Lines
- The Fibonaccian behavior
- Volatility Quality Index
- Zig Explorer
- Zig Zag Indicator with Valid Entry and Exit Points
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag Hi Lo Barcount
- ZigZag Retracements

More information:

See updated/extended version on-line.

_DEFAULT_NAME - retrive default name of the plot

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **_DEFAULT_NAME()**

RETURNS STRING

FUNCTION This function returns the default name of plot in the drag-drop section. The default name consists of section name and comma separated list of values of numeric parameters defined in given section.

EXAMPLE

```
_SECTION_BEGIN("MA1");
P = ParamField("Price field");
Periods = Param("Periods", 15, 2, 200, 1, 10 );
Plot( MA( P, Periods ), _DEFAULT_NAME(), ParamColor( "Color",
colorCycle ), ParamStyle("Style") );
_SECTION_END();
```

_DEFAULT_NAME will evaluate to "MA1(Close,15)" string.

SEE ALSO _SECTION_BEGIN() function , _SECTION_NAME() function , _SECTION_END() function

References:

The **_DEFAULT_NAME** function is used in the following formulas in AFL on-line library:

- Advanced MA system
- AFL Example
- AFL Example - Enhanced
- AFL to Python COM Link
- AO+Momentum
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- babaloo chapora
- Button trading using AB auto trading interface
- Dave Landry PullBack Scan
- DEBAJ
- DiNapolis 3x Displaced Moving Averages
- DPO with shading
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- Elliott Wave Oscillator
- Fibonacci Moving averages
- Gann level plotter
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Herman
- IB Backfiller
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Intraday Trend Break System

- Manual Bracket Order Trader
- mitalpradip
- Mndahoo ADX
- Modified-DVI
- Nadaraya-Watson Envelope
- Neural Network Powered Smooth/Predictive RSI V2
- Noor_Doodie
- ParabXO
- prakash
- PVT Trend Decider
- Random Walk Index
- RI - Auto Trading System
- RSI indicator with Upper & Lower Zone Bars
- RSI styleClipMinMax
- shailu lunia
- Simple Momentum
- Triangular Moving Average new
- Twiggs Money Flow
- Ultimate plus
- UltraEdit editor highlight wordfile
- Vikram's Floor Pivot Intraday System
- visual turtle trading system
- ZigZag filter rewrited from scratch in AFL
- ZigZag Hi Lo Barcount

More information:

See updated/extended version on-line.

_DT
- convert string to datetime

Date/Time
(AmiBroker 5.40)

SYNTAX	**_DT("date string");**
RETURNS	NUMBER
FUNCTION	Converts string representing date/time value to the corresponding DateTime number (that can be later compared to output of DateTime() function for example). This function is synonym / shortcut for StrToDateTime function.
EXAMPLE	
SEE ALSO	StrToDateTime() function

References:

The **_DT** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

_exit - terminate the execution of AFL formula

Miscellaneous functions
(AmiBroker 6.40)

SYNTAX **_exit()**

RETURNS NOTHING

FUNCTION The function gracefully ends AFL execution at the point of the call. The code after _exit() call will not be executed.

It is useful if you want to terminate AFL processing quicker than usual in certain situations

EXAMPLE

```
if( Status("action" == actionScan )
{
   // if we are running scan we just create composite
   AddToComposite( Volume, "~~~TotalVol", "X" );
   // and end the execution quickly
   _exit();
}

// rest of the code that will only be executed if NOT running SCAN
```

SEE ALSO

References:

The **_exit** function is used in the following formulas in AFL on-line library:

- Add SL/TGT other params to any strategy
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- MFE and MAE and plot trades as indicator

More information:

See updated/extended version on-line.

_N - no text output

Exploration / Indicators
(AmiBroker 4.10)

SYNTAX	**_N(string)**
RETURNS	nothing
FUNCTION	Protects from printing string to the commentary output window
EXAMPLE	_N(ticker = name()); // thanks to _N function ticker symbol is not printed
SEE ALSO	ENABLETEXTOUTPUT() function

References:

The **_N** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 'R' Channel
- 10-20 Indicator
- 2 Timeframes Candlestick Bar Chart
- 3 ways to use RMI in one script
- 30 Week Hi Indicator - Display
- 3TF Candlestick Bar Chart
- 4% Model - Determine Stock Market Direction
- 52 Week New High-New Low Index
- A simple AFL Revision Control System
- Abhimanyu
- AccuTrack
- Adaptive Laguerre Filter, from John Ehlers
- Add Nifty 50 IB Equity Symbol Automatically
- Add SL/TGT other params to any strategy
- Advanced MA system
- Advanced Search and Find
- Advanced Trend Lines with S & R
- Advisory NRx price chart display.
- ADXbuy
- ADXVMA
- AFL Example
- AFL Example - Enhanced
- AFL to Python COM Link
- AFL_Glossary_1
- AFL_Glossary_Converter
- Alert Output As Quick Rewiev
- ALJEHANI
- AllinOneAlerts - Module
- Alpha and Beta and R_Squared Indicator
- Alphatrend
- Alternative ZIG function
- Alternative ZIG type function, multi TF
- Andrews Pitchfork
- Andrews PitchforkV3.3
- Animated BackGround

- Animated BackGround 1.1
- AO+Momentum
- Appel's ROC or The Triple Momentum Timing Model
- Aroon Indicators
- Aroon The Advisor
- AR_Prediction.afl
- ATR Study
- Auto Fib Ext&Retracement
- Auto Trade Step by Step
- Auto-Optimization Framework
- Automatic Linear Trend Channel
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trend-line
- automatic trendlines using fractal patterns
- Automatic Trendlines using multiple timeframes
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Dollar Price Volatility Exploration
- Average Price Crossover
- B-Xtrender
- babaloo chapora
- Backup Data of 1min Interval
- Bad Tick Trim on 5 sec database
- Baseline Relative Performance Watchlist charts V2
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- BEANS-Summary of Holdings
- Bid Vs Ask Dashboard
- Binary to Decimal Converter
- Black Scholes Option Pricing
- Bman's HaDiffCO
- BMTRIX Intermediate Term Market Trend Indicator
- Bollinger band normalization
- Bollinger Band Squeeze
- Bollinger Band Width
- Bottom Fisher Exploration
- Bull Fear / Bear Fear
- Bullish Percent Index
- Bullish Percent Index 2 files combined
- Bullish Percent Index 2004
- Button trading using AB auto trading interface
- Calculate composites for tickers in list files
- Caleb Lawrence
- CAMSLIM Cup and Handle Pattern AFL
- Candle Identification
- Candle Pattern Function
- Candle Stick Analysis
- Candle Stick Demo
- candlestick chart for Volume/RSI/OBV
- CandleStick Comentary--Help needed

- Candlestick Commentary
- Candlestick Commentary Modified
- Candlestick Commentary-modified
- CCI 14 DrBobStyle
- CCI 50 DrBob Style
- CCI(20) Divergence Indicator
- Chandelier Exit
- Chandelier Exit
- changing period moving avarage
- channel indicator
- Channel/S&R and trendlines
- Chart Zoom
- CoinToss ver 1
- Cole
- Color Combination
- Color Display.afl
- Color Price Bar - Impulse System
- Color Price Bars with MACD Histogram Changes
- Colorfull Price
- com-out
- Commodity Selection Index (CSI)
- Computing Cointegration and ADF Dashboard
- Congestions detection
- Connors TPS
- Continuous Contract Rollover
- Coppock Histogram
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- correlerror
- COT REPORT
- Count Tickers in Watchlist
- Create a list of functions in your program
- crMathLib
- CVR--severe filter
- Cybernertic Hilbert Sine Wave
- Cycle Highlighter
- Cycle Highlighter (auto best-fit)
- Cycle Period
- Daily High Low in Advance
- danningham penetration
- Darvas Amibroker
- Darvas Johndeo Research
- Darvas Wisestocktrader
- DateNum_DateStr
- Date_To_Num(), Time_To_Num()
- Dave Landry PullBack Scan
- Day Bar No
- DEBAJ
- Demand Index
- Detailed Equity Curve
- Digital indiactors
- DiNapolis 3x Displaced Moving Averages

- Divergence indicator
- Divergences
- DMI Spread Index
- Dominant Cycle Phase
- Double Super Trend System
- Double top detection
- Dynamtic Momentum Index
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- Ehlers Hilbert Transformer Indicator
- Ehlers Reflex Indicator
- Ehlers Trendflex Indicator
- ekeko price chart
- EKEKO SAR-MF
- Elder Impulse Indicator V2
- Elder Ray - Bull Bear
- Elder Ray Oscillator with MA
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- elliott wave manual labelling
- Elliott Wave Oscillator
- EMA Crossover
- End Of Year Trading
- Envelope System
- Evaluating Candle Patterns in a trading system
- Even Better Sinewave Indicator
- Expiry day/days - Last thursday of month
- Export All Daily Data to TXT with MS import format
- Export EOD or Intraday to .csv file
- Export Intraday Data
- Extract specific lines of code from your program
- Fast Refreshed KAGI Swing Charts (Price Swing)
- FastStochK FullStochK-D
- Fib Fan Based on ZZ
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- FirstBarIndex(), LastBarIndex()
- Fisher Oscillator
- Follow the Leader
- For Auto Trading Setup
- Force index
- Fre
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Fund Screener
- Future MA Projection
- Future Plotting of Time and Price
- Future Plotting of Time and Price
- Gabriel Linear Regression Angle Indicator
- Gann HiLo Indicator and System
- Gann level plotter

- Gann Swing Chart
- Gann Swing chart v41212
- Gann Swing Charts in 3 modes with text
- Geometric Mean of Volume
- Get Moneycontrol News Snippets into Amiboker
- Gfx Toolkit
- GFX ToolTip
- Gordon Rose
- Graphical sector analysis
- Graphical sector stock amalysis
- Guppy Cloud
- half-automated Trading System
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Head & Shoulders Pattern
- Heatmap V1
- Heikin Ashi Candles
- Heikin Ashi Delta
- Heikin Ashi System
- Heikin-Ashi(Koma-Ashi) with Moving Average
- Heinkin-Ashi
- Herman
- HH-LL-PriceBar
- High Low Detection code
- Hilbert Sine Wave
- Hilbert Sine Wave Support & Resistance
- Hilbert Sine Wave with Hull Moving Average
- Hilbert Study
- Historical Volatility Index
- Historical Volotility Scan - 6/100
- Historical Volotility Scan - 50 Day
- HLspread
- How to add IB Option Symbols
- Hull Rate of Return Indicator
- IB Backfiller
- IBD relative strength database ranker
- IBD relative strength database Viewer
- ICHIMOKU SIGNAL TRADER
- Ichimoku System
- IchimokuBrianViorelRO
- Improved NH-NH scan / indicator
- In Watch List
- Index of 30 Wk Highs Vs Lows
- Indicator Explorer (ZigZag)
- Inside Bar
- Inside Bar
- Inter-market Yield Linear Regression Divergence
- interactively test discretionary trading
- Intraday Average Volume
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new

- Intraday Strength
- Intraday Trend Break System
- Inverted Plotted Volume Overlay Indicator
- Javascript Formulas for skalp and update portfolio
- JEEVAN'S SRI CHAKRA
- John Ehler
- Kelly criterion
- Kiss and Touch with the Modified True Range
- Last Five Trades Result Dashboard – AFL code
- Laugerre PPO Oscillator
- Least Squares Channel Indicator
- Linear Candle
- Luna Phase
- LunarPhase
- MACD BB Indicator
- MACD commentary
- MACD Histogram - Change in Direction
- MACD indicator display
- Manual Bracket Order Trader
- Market Breadth Chart-In-Chart
- Market Facilitation Index VS Volume
- Market Meanness Index
- Market Profile
- MCDX (Multi Color Dragon Extended)
- MFE and MAE and plot trades as indicator
- mitalpradip
- MOCS
- Modified Head & Shoulder Pattern
- Modified-DVI
- Momentum Volume Price (MVP) Indicator
- Monthly bar chart
- Moving Average "Crash" Test
- MO_CrashZone
- MS Darvas Box with Exploration
- Multi-color Volume At Price (VAP)
- MultiCycle 1.0
- Multiple Ribbon Demo
- Multiple sinus noised
- NABIN SINHA
- Nadaraya-Watson Envelope
- NASDAQ 100 Volume
- Neural Network Powered Smooth/Predictive RSI V2
- New HL Scanner
- Nick
- nifty
- Non-repaitning Zigzag line
- Noor_Doodie
- Now Send Push Notifications From Amibroker
- NRx Exploration
- nth (1 - 8) Order Polynomial Fit
- OBV with Linear Regression
- Open Range Breakout Trading System

- Option Calls, Puts and days till third friday.
- Ord Volume
- P&F Chart - High/Low prices Sept2003
- P&F chart with range box sizes
- Parabolic SAR in VBScript
- Parametric Chande Trendscore
- pattenz
- Pattern - Rectangle Base Breakout on High Vol
- Pattern Recognition Exploration
- Pattern_-_Rectangle_Base_Breakout_on_High_Vol 2
- Percentage Price Oscillator
- Perceptron
- Performance Check
- Periodically ReBalance a BUY & HOLD Portfolio
- Peter Cooper
- Peterson
- PF Chart - Close - April 2004
- Pivot End Of Day Trading System
- Pivot Finder
- Pivots And Prices
- Pivots for Intraday Forex Charts
- Plot the Equity Curve without Backtesting?
- plot tomorrows pivots on an intraday database
- Plot visual stop / target ratio.
- Point & figure Chart India Securities
- Polyfit Lines
- Position Sizer vers2, stocks and CFDs
- Positive Bars Percentage
- prakash
- Prashanth
- Price Persistency
- Prior Daily OHLC
- Probability Density & Gaussian Distribution
- Profit Table (Color Coded)
- Projection Oscillator
- Pullback System No. 1
- PVT Trend Decider
- qp2 industry charts as a panel in the stocks chart
- R-Squared
- Rainbow Oscillator
- Random Walk Index
- Range Filter - Trading Strategy
- Ranking and sorting stocks
- Ranking Ticker WatchList
- Rebalancing Backtest avoiding leverage
- Reconnect TWS
- Relative Strength
- Relative Strength Index
- Relative Strength Multichart of up to 10 tickers
- Relative Volume
- Rene Rijnaars
- Renko Chart

- Renko Chart
- Reverse MFI Crossover
- Revised Renko chart
- RI - Auto Trading System
- Robert Antony
- RSI + Avgs
- RSI Double-Bottom
- RSI indicator with Upper & Lower Zone Bars
- RSI of Weekly Price Array
- RSI Trendlines and Wedges
- RUTVOL timing signal with BB Scoring routine
- Sainath Sidgiddi
- SAR-ForNextBarStop
- Say Notes
- Scale Out: Futures
- Scan New High and New Low
- Schiff Lines
- Sector Tracking
- SectorRSI
- Send Alerts from Amibroker to Telgram
- shailu lunia
- Signal to Noise
- Simple Candle Exploration
- Sine Wave Indicator
- Smoothed Adaptive Momentum
- Sony
- Spearman Rank Correlation Coefficient
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- STD_STK Multi
- Steve Woods' Cum. Vol. Float + Cum. Vol. Channels
- Steve Woods' Cumulative Vol. Percentage Indicator
- Stochastic %J - KDJ
- Stochastic Divergence, negative
- Stochastic Divergence, positive
- Stochastic Divergences, PDI, NDI
- Stochastic of Weekly Price Array
- Stochastic Oscillator
- Stochastics Trendlines
- StochD_StochK Single.afl
- Stops Implementation in AFS
- Stops on percentages
- Strength and Weakness
- Stress with SuperSmoother
- SUPER PIVOT POINTS
- Super Trend Indicator
- Super Trend Indicator
- Support and Resistance
- Support and resistance
- Support Resistance levels
- suresh
- TAPE READING

- TAZ Trading Method Exploration
- TD Channel-1
- TD Channel-2
- TD Moving Average 2
- TD Moving Average I
- TD REI
- TD sequential
- TD Sequential
- testing multiple system simulataneously
- The D_oscillator
- The Fibonaccian behavior
- Three Day Balance Point
- Three Line Break - TLB
- Three Pole Butterworth
- Time Frame Weekly Bars
- Time Left in Bar
- tomy_frenchy
- Tracking Error
- Tracking Error
- Trades per second indicator
- Trend Analysis_Comentary
- Trend Detection
- Trend exploration with multiple timeframes
- Trend Exploration: Count Number of New Highs
- Trend Exploration: Slope Moving Average
- Trend Following System
- Trend Lines from 2 points
- Trending Ribbon
- TrendingRibbonArrowsADX
- Triangle exploration using P&F Chart
- Triangle Search Extended
- Triangular Moving Average new
- Trigonometric Fit - TrigFit with AR for cos / sin
- TSV
- TTM Squeeze
- Twiggs Money Flow
- Twiggs money flow weekly
- TWS auto-export Executions-file parser
- TWS trade plotter
- Ultimate plus
- UltraEdit editor highlight wordfile
- Updated Renko Chart
- Using From and To dates from Auto Analysis in Code
- Vikram's Floor Pivot Intraday System
- Visi-Trade
- Visible Min and Max Value Demo
- visual turtle trading system
- Visualization of stoploses and profit in chart
- Volatility Quality Index
- Volatility System
- Volume - compared with Moving Avg (100%)
- Volume based support resistance price finder

- Volume Charts
- Volume Occilator
- Volume Spread for VSA
- Volume Zone Oscillator
- Voss Predictive Filter (A Peek Into the Future)
- VSTOP (2)
- VSTOP (3)
- VWAP - Volume Weighted Average Price
- Weekly chart
- Weekly Trend in Daily Graph
- Weighted Index
- White Theme
- William's Alligator System II
- Williams Alligator system
- WILSON RELATIVE PRICE CHANNEL
- WLBuildProcess
- Wolfe Wave Patterns
- Woodie's CCI Panel Basic
- Woodie's CCI Panel Full Stats
- Woodie's Heikin-Ashi Panel
- Woodie's Price Panel With Woodie's Pivots
- Zig Zag
- Zig Zag Indicator with Valid Entry and Exit Points
- Zig-Hi Zap-Lo
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag filter rewrited from scratch in AFL
- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

_PARAM_VALUES
- retrieve param values string

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX	**_PARAM_VALUES()**
RETURNS	STRING
FUNCTION	_PARAM_VALUES retrieves the values of the parameters defined in current drag-drop section. It works the same as _DEFAULT_NAME except that no section name is included (so only the list of parameter values is returned).
EXAMPLE	
SEE ALSO	_DEFAULT_NAME() function

References:

The **_PARAM_VALUES** function is used in the following formulas in AFL on-line library:

- Advanced MA system
- ALJEHANI
- babaloo chapora
- BBAreacolor&TGLCROSSNEW
- Bman's HaDiffCO
- bolingerbands
- bonlinger bands
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Elder Triple Screen Trading System
- Guppy Cloud
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- prakash
- Spearman Rank Correlation Coefficient
- ZigZag - Days, Avg (Ord) Volume and Channels

More information:

See updated/extended version on-line.

_SECTION_BEGIN - section begin marker

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX **_SECTION_BEGIN("section name")**

RETURNS NOTHING

FUNCTION Marks beginning of the drag-drop section.

IMPORTANT: "section name" must be a constant, literal string, enclosed in double quotation marks. **You must NOT use variable here.**

EXAMPLE

SEE ALSO

References:

The **_SECTION_BEGIN** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 10-20 Indicator
- 2 Timeframes Candlestick Bar Chart
- 3 ways to use RMI in one script
- 3TF Candlestick Bar Chart
- Add Nifty 50 IB Equity Symbol Automatically
- Add SL/TGT other params to any strategy
- Advanced MA system
- Advanced Trend Lines with S & R
- ADXVMA
- AFL Example
- AFL to Python COM Link
- AFL_Glossary_Converter
- ALJEHANI
- AllinOneAlerts - Module
- Alphatrend
- Animated BackGround
- Animated BackGround 1.1
- AO+Momentum
- Appel's ROC or The Triple Momentum Timing Model
- Aroon The Advisor
- Auto Fib Ext&Retracement
- Auto Trade Step by Step
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- B-Xtrender
- babaloo chapora
- Bad Tick Trim on 5 sec database
- Basket Trading System T101

- BBAreacolor&TGLCROSSNEW
- Bid Vs Ask Dashboard
- Bman's HaDiffCO
- Bollinger Band Width
- Button trading using AB auto trading interface
- Caleb Lawrence
- Candle Identification
- Candle Stick Demo
- Chandelier Exit
- Chandelier Exit
- changing period moving avarage
- Chart Zoom
- CoinToss ver 1
- Colorfull Price
- com-out
- Commodity Selection Index (CSI)
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- Coppock Histogram
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Darvas Wisestocktrader
- DateNum_DateStr
- Dave Landry PullBack Scan
- Day Bar No
- DEBAJ
- Digital indiactors
- DiNapolis 3x Displaced Moving Averages
- Dynamtic Momentum Index
- Elder Impulse Indicator V2
- Elder Ray - Bull Bear
- Elder Ray Oscillator with MA
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- elliott wave manual labelling
- Elliott Wave Oscillator
- Expiry day/days - Last thursday of month
- FastStochK FullStochK-D
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- For Auto Trading Setup
- Force index
- Fre
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future MA Projection
- Future Plotting of Time and Price
- Future Plotting of Time and Price

- Gann level plotter
- Gann Swing chart v41212
- Gann Swing Charts in 3 modes with text
- Geometric Mean of Volume
- Get Moneycontrol News Snippets into Amiboker
- GFX ToolTip
- Graphical sector analysis
- Graphical sector stock amalysis
- Guppy Cloud
- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Heatmap V1
- Heikin Ashi Candles
- Heikin Ashi Delta
- HH-LL-PriceBar
- High Low Detection code
- Historical Volatility Index
- Historical Volotility Scan - 6/100
- Historical Volotility Scan - 50 Day
- How to add IB Option Symbols
- Hull Rate of Return Indicator
- IB Backfiller
- IchimokuBrianViorelRO
- Improved NH-NH scan / indicator
- In Watch List
- Inside Bar
- Inside Bar
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Intraday Strength
- Intraday Trend Break System
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Kiss and Touch with the Modified True Range
- Last Five Trades Result Dashboard – AFL code
- Laugerre PPO Oscillator
- Linear Candle
- Luna Phase
- MACD BB Indicator
- MACD Histogram - Change in Direction
- Manual Bracket Order Trader
- Market Breadth Chart-In-Chart
- Market Facilitation Index VS Volume
- MCDX (Multi Color Dragon Extended)
- mitalpradip
- Modified Head & Shoulder Pattern
- Modified-DVI
- MS Darvas Box with Exploration
- Multiple Ribbon Demo
- Nadaraya-Watson Envelope
- NASDAQ 100 Volume

- Neural Network Powered Smooth/Predictive RSI V2
- Nick
- Non-repaitning Zigzag line
- Noor_Doodie
- Now Send Push Notifications From Amibroker
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- Parametric Chande Trendscore
- pattenz
- Percentage Price Oscillator
- Peter Cooper
- Pivots And Prices
- Plot the Equity Curve without Backtesting?
- Plot visual stop / target ratio.
- Point & figure Chart India Securities
- Position Sizer vers2, stocks and CFDs
- Positive Bars Percentage
- prakash
- Prior Daily OHLC
- Random Walk Index
- Range Filter - Trading Strategy
- Reconnect TWS
- Relative Volume
- Rene Rijnaars
- RI - Auto Trading System
- Robert Antony
- Say Notes
- Scale Out: Futures
- Send Alerts from Amibroker to Telgram
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Stochastic %J - KDJ
- Stochastic Oscillator
- Stops on percentages
- Stress with SuperSmoother
- SUPER PIVOT POINTS
- Super Trend Indicator
- TAPE READING
- TD Channel-1
- TD Channel-2
- TD Moving Average 2
- TD Moving Average I
- TD REI
- TD Sequential
- Trades per second indicator
- Trending Ribbon
- TrendingRibbonArrowsADX
- Triangular Moving Average new
- TSV
- TTM Squeeze
- Twiggs Money Flow
- TWS trade plotter

- Ultimate plus
- UltraEdit editor highlight wordfile
- Updated Renko Chart
- Vikram's Floor Pivot Intraday System
- Visible Min and Max Value Demo
- visual turtle trading system
- Volume Occilator
- Volume Spread for VSA
- Volume Zone Oscillator
- VWAP - Volume Weighted Average Price
- William's Alligator System II
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag filter rewrited from scratch in AFL
- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

_SECTION_END - section end marker

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX	**_SECTION_END()**
RETURNS	NOTHING
FUNCTION	marks end of drag-drop section
EXAMPLE	
SEE ALSO	

References:

The **_SECTION_END** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- 10-20 Indicator
- 2 Timeframes Candlestick Bar Chart
- 3 ways to use RMI in one script
- 3TF Candlestick Bar Chart
- Add Nifty 50 IB Equity Symbol Automatically
- Advanced MA system
- Advanced Trend Lines with S & R
- ADXVMA
- AFL Example
- AFL to Python COM Link
- AFL_Glossary_Converter
- ALJEHANI
- AllinOneAlerts - Module
- Alphatrend
- Animated BackGround
- Animated BackGround 1.1
- AO+Momentum
- Aroon The Advisor
- Auto Fib Ext&Retracement
- Automatic Linear Trend Channel 2
- Automatic trend channel
- Automatic Trendlines using multiple timeframes
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- Average Price Crossover
- B-Xtrender
- babaloo chapora
- Bad Tick Trim on 5 sec database
- Basket Trading System T101
- BBAreacolor&TGLCROSSNEW
- Bid Vs Ask Dashboard
- Bman's HaDiffCO
- Bollinger Band Width
- Button trading using AB auto trading interface
- Caleb Lawrence

- Candle Identification
- Candle Stick Demo
- Chandelier Exit
- Chandelier Exit
- changing period moving avarage
- Chart Zoom
- CoinToss ver 1
- Colorfull Price
- com-out
- Commodity Selection Index (CSI)
- Computing Cointegration and ADF Dashboard
- Continuous Contract Rollover
- Coppock Histogram
- Coppock Trade Signal on Price Chart
- Coppock Trade Signal v1.1
- Daily High Low in Advance
- Darvas Amibroker
- Darvas Johndeo Research
- Darvas Wisestocktrader
- Dave Landry PullBack Scan
- Day Bar No
- DEBAJ
- Digital indiactors
- DiNapolis 3x Displaced Moving Averages
- Dynamtic Momentum Index
- Elder Impulse Indicator V2
- Elder Ray - Bull Bear
- Elder Ray Oscillator with MA
- Elder safe Zone Long + short
- Elder Triple Screen Trading System
- Elder's Market Thermometer
- elliott wave manual labelling
- Elliott Wave Oscillator
- Expiry day/days - Last thursday of month
- FastStochK FullStochK-D
- Fibonacci Calculations & Speed Resistance
- Fibonacci Internal and External Retracements
- Fibonacci Moving averages
- For Auto Trading Setup
- Force index
- Fre
- FTWHMS - FIFTY TWO WEEKS HIGH MOMENTUM STRATEGY
- Future MA Projection
- Gann level plotter
- Gann Swing chart v41212
- Gann Swing Charts in 3 modes with text
- Geometric Mean of Volume
- Get Moneycontrol News Snippets into Amiboker
- GFX ToolTip
- Graphical sector analysis
- Graphical sector stock amalysis
- Guppy Cloud

- Halftrend
- Harmonic Pattern Detection
- Harmonic Patterns
- Heatmap V1
- Heikin Ashi Candles
- Heikin Ashi Delta
- HH-LL-PriceBar
- High Low Detection code
- Historical Volatility Index
- Historical Volotility Scan - 6/100
- Historical Volotility Scan - 50 Day
- How to add IB Option Symbols
- Hull Rate of Return Indicator
- IB Backfiller
- ICHIMOKU SIGNAL TRADER
- IchimokuBrianViorelRO
- Improved NH-NH scan / indicator
- In Watch List
- Inside Bar
- Inside Bar
- Intraday Fibonacii Trend Break System
- INTRADAY HEIKIN ASHI new
- Intraday Strength
- Intraday Trend Break System
- Inverted Plotted Volume Overlay Indicator
- JEEVAN'S SRI CHAKRA
- Kiss and Touch with the Modified True Range
- Last Five Trades Result Dashboard – AFL code
- Laugerre PPO Oscillator
- Linear Candle
- Luna Phase
- MACD BB Indicator
- MACD Histogram - Change in Direction
- Manual Bracket Order Trader
- Market Breadth Chart-In-Chart
- Market Facilitation Index VS Volume
- MCDX (Multi Color Dragon Extended)
- mitalpradip
- Modified Head & Shoulder Pattern
- Modified-DVI
- MS Darvas Box with Exploration
- Multiple Ribbon Demo
- Nadaraya-Watson Envelope
- NASDAQ 100 Volume
- Neural Network Powered Smooth/Predictive RSI V2
- Non-repaitning Zigzag line
- Noor_Doodie
- Now Send Push Notifications From Amibroker
- Open Range Breakout Trading System
- Option Calls, Puts and days till third friday.
- Parametric Chande Trendscore
- pattenz

- Percentage Price Oscillator
- Peter Cooper
- Pivots And Prices
- Plot the Equity Curve without Backtesting?
- Plot visual stop / target ratio.
- Point & figure Chart India Securities
- Position Sizer vers2, stocks and CFDs
- Positive Bars Percentage
- prakash
- Prior Daily OHLC
- Random Walk Index
- Range Filter - Trading Strategy
- Reconnect TWS
- Relative Volume
- Rene Rijnaars
- RI - Auto Trading System
- Robert Antony
- Say Notes
- Scale Out: Futures
- Send Alerts from Amibroker to Telgram
- Square of Nine Roadmap Charts
- Stan Weinstein strategy
- Stochastic %J - KDJ
- Stochastic Oscillator
- Stops on percentages
- Stress with SuperSmoother
- SUPER PIVOT POINTS
- Super Trend Indicator
- TAPE READING
- TD Channel-1
- TD Channel-2
- TD Moving Average 2
- TD Moving Average I
- TD REI
- TD Sequential
- Trades per second indicator
- Trending Ribbon
- TrendingRibbonArrowsADX
- Triangular Moving Average new
- TSV
- TTM Squeeze
- Twiggs Money Flow
- TWS trade plotter
- Ultimate plus
- UltraEdit editor highlight wordfile
- Updated Renko Chart
- Vikram's Floor Pivot Intraday System
- Visi-Trade
- Visible Min and Max Value Demo
- visual turtle trading system
- Volume based support resistance price finder
- Volume Occilator

- Volume Spread for VSA
- Volume Zone Oscillator
- VWAP - Volume Weighted Average Price
- William's Alligator System II
- Woodie's CCI Panel Full Stats
- ZigZag - Days, Avg (Ord) Volume and Channels
- ZigZag filter rewrited from scratch in AFL
- ZigZag Hi Lo Barcount
- ZigZag Retracements
- ZLEMA ATR Long Only Trading System

More information:

See updated/extended version on-line.

_SECTION_NAME - retrieve current section name

Exploration / Indicators
(AmiBroker 4.70)

SYNTAX	**_SECTION_NAME()**
RETURNS	STRING
FUNCTION	The function that gives the name of the drag-drop section (given in previous _SECTION_BEGIN call).
EXAMPLE	
SEE ALSO	_SECTION_BEGIN() function

References:

The **_SECTION_NAME** function is used in the following formulas in AFL on-line library:

- ALJEHANI
- Elder Triple Screen Trading System
- UltraEdit editor highlight wordfile

More information:

See updated/extended version on-line.

_TRACE
- print text to system debug viewer

Miscellaneous functions
(AmiBroker 4.40)

SYNTAX **_TRACE("string")**

RETURNS NOTHING

FUNCTION Write debug messages from AFL code to system debug viewer (it calls internally OutputDebugString Win API function) or to internal Log window (Window->Log) To view debug messages sent to system debugger you have to run DebugView freeware program from https://learn.microsoft.com/en-us/sysinternals/downloads/debugview To view messages sent to internal log window you need to display log window (Window->Log menu)

Note for internal viewer: you can specify _TRACE("!CLEAR!"); to clear internal log window

EXAMPLE

```
_TRACE("This is a test");
_TRACE("This is selected value of close: " + Close );
_TRACE("This is first element of close array: " + Close[ 0 ] );
```

SEE ALSO

References:

The **_TRACE** function is used in the following formulas in AFL on-line library:

- Pivots And Prices And Swing Volume
- Auto Trade Step by Step
- AutoTrade using an Exploration
- AutoTrader Basic Flow - updated April 15, 2009
- AutoTrader Basic Flow - updated Nov 18, 2008
- babaloo chapora
- Button trading using AB auto trading interface
- Calculate composites for tickers in list files
- CCI(20) Divergence Indicator
- Congestions detection
- Create a list of functions in your program
- Detailed Equity Curve
- Ed Seykota's TSP: EMA Crossover System
- Ed Seykota's TSP: Support and Resistance
- Extract specific lines of code from your program
- For Auto Trading Setup
- Herman
- lastNDaysBeforeDate
- Market Meanness Index
- MFE and MAE and plot trades as indicator
- Ord Volume
- Pivot Finder
- Plot visual stop / target ratio.
- Profit Table (Color Coded)
- Scale Out: Futures
- shailu lunia
- suresh
- Volume Color with Dynamic Limit

- WLBuildProcess

More information:

See updated/extended version on-line.

_TRACEF
- print formatted text to system debug viewer

Miscellaneous functions
(AmiBroker 60)

SYNTAX	**_TRACEF("format string", arg1, ...)**
RETURNS	NOTHING
FUNCTION	This function is the same as _TRACE but the very first argument is printf-style formatting string and it allows to output formatted numbers like combination of _TRACE and StrFormat. The function writes debug messages from AFL code to system debug viewer (it calls internally OutputDebugString Win API function) or to internal Log window (Window->Log) To view debug messages sent to system debugger you have to run DebugView freeware program from https://learn.microsoft.com/en-us/sysinternals/downloads/debugview To view messages sent to internal log window you need to display log window (Window->Log menu) Note for internal viewer: you can specify _TRACE("!CLEAR!"); to clear internal log window
EXAMPLE	_TRACEF("Close price is = %g", Close);
SEE ALSO	_TRACE() function

References:

The **_TRACEF** function is used in the following formulas in AFL on-line library:

More information:

See updated/extended version on-line.

AFL Error List

- Error 1. Operation not allowed. Operator/operand type mismatch.
- Error 2. Incorrect type of argument. Expecting number or array here.
- Error 3. Type mismatch, Unary minus operator requires number or array
- Error 4. Incorrect type of argument(s). Expecting number here.
- Error 5. Argument #1 has incorrect type (the function expects different argument type here)
- Error 6. Condition in IF, WHILE, FOR statements has to be Numeric or Boolean type. You can not use array here
- Error 7. Condition in IF, WHILE, FOR statements has to be Numeric or Boolean type. You can not use STRING here
- Error 8. Type mismatch, the value assigned to the array element has to be a number. You can not use array...
- Error 9. Array subscript has to be a number
- Error 10. Subscript out of range. You must not access array elements outside 0..(BarCount-1) range
- Error 11. Subscript operator [] requires array or number type. String can not be used here.
- Error 12. Subscript operator [] requires array or number type.
- Error 13. Endless loop detected in WHILE loop
- Error 14. Endless loop detected in DO-WHILE loop
- Error 15. Endless loop detected in FOR loop
- Error 16. Too many arguments
- Error 17. Missing arguments
- Error 18. COM object variable is not initialized or has invalid type (valid COM object handle required)
- Error 19. COM method/function 'function name' call failed.
- Error 20. COM Method/function '%s' does not exist
- Error 21. Relative strength base symbol not found
- Error 22. Bad 'format' argument type - format has to be a number (not array)
- Error 23. GetExtraData call failed

- Error 24. This formula requires AmiBroker version '...' (or higher).
- Error 25. SetVariable() called from plug in was not successful. Identifier is already used in a different context.
- Error 26. File handle passed to the function is invalid (equal to zero).
- Error 27. Invalid number of arguments passed to Call Function (..) from plugin DLL
- Error 28. Out of memory
- Error 29. Variable used without having been initialized.
- Error 30. Syntax error
- Error 31. Syntax error, expecting 'list of tokens'
- Error 32. Syntax error, probably missing semicolon at the end of the previous line
- Error 33. The identifier is already in use. You attempted to assign value to the identifier representing a function.
- Error 34. The identifier is already in use. You attempted to define the function that has the same identifier as global variable.
- Error 35. Shift+BREAK pressed. Loop terminated.
- Error 36. N-th argument of the function call has no value set
- Error 37. Unsupported field in SetOptions
- Error 38. Unsupported field in GetOptions
- Error 39. CategoryAddSymbol: Setting sector is unsupported, set industry instead
- Error 40. CategoryRemoveSymbol: Removing from sector is unsupported, remove from industry instead
- Error 41. Unsupported field in GetRTData
- Error 42. #include failed because the file does not exist: 'filename' (current working directory is '...')
- Error 43. Variable stops are not supported in Rotational Trading mode
- Error 44. SectorID() is outside 0..63 range.
- Error 45. Failed to launch trading interface
- Error 46. Missing comma
- Error 47. Exception occurred during AFL formula execution

- Error 48. N-volume bar compressed data longer than base time frame. Use higher compression factor.

- Error 49. Optimization parameter name must not be empty.

- Error 50. Optimization parameter minimum value must be less than or equal to maximum and step parameter needs to be greater than zero.
- Error 51. Array subscript has Null value. Subscript must be within 0...BarCount-1 range.
- Error 52. Invalid argument value for name() function. Argument 'paramname' must be positive (and not Null)
- Error 53. You have N open file(s) that you forgot to close. You must call fclose() function for every file opened with Error 54. Incorrect espace sequence. Supported sequences are \n, \r, \t, \" and \\ (gives single backslash)
- Error 90. Specified Optimizer Engine not found
- Error 91. OptimizerSetOption expects STRING value
- Error 92. OptimizerSetOption expects NUMBER (scalar numeric) value
- Error 93. Specified option is not supported / not available by the selected optimizer
- Error 94. External Optimizer is not selected. You must call OptimizerSetEngine() prior to calling OptimizerSetOption()
- Warning 501. Assignment within conditional. Did you mean == instead of = ?
- Warning 502. You are calling Plot()/PlotOHLC() function over 500 times, it is highly inefficient. Reduce number of Warning 503. Using OLE /
  CreateObject / CreateStaticObject is not multi-threading friendly.
- Error 701. Missing buy/sell variable assignments
- Error 702. Missing short/cover variable assignments
- Error 703. Rotational trading requires PositionScore variable.
- Error 704. In Rotational trading you must NOT use buy/sell/short/cover signals.
- Error 705. You can not use HoldMinBars together with Allow Same Bar exit.
- Error 706. Show Arrows feature needs a Trade list.

Error 1. Operation not allowed. Operator/operand type mismatch.

An arithmetic, string, logical or comparison operator is being used with an invalid data type. This error would occur, for example, if you were to attempt to multiply two string values.

```
a = "x" * 5; // wrong, can not multiply string by number
b = "x" - "y"; // wrong, can not subtract strings

z = "x" + "y"; // correct, concatenation of strings is OK
```

Error 2. Incorrect type of argument. Expecting number or array here.

This occurs when calling single-argument mathematical function like sin() which accepts only numbers and arrays,
but the user specified string for example.

```
x= sin("test"); // sin requires number or array
```

Error 3. Type mismatch, Unary minus operator requires number or array

Occurs when trying to apply unary minus operator to strings.

```
text2 = - "test"; // can not use unary minus (negation) to texts
```

Error 4. Incorrect type of argument(s). Expecting number here.

This occurs in ApplyStop function when Type, Mode, ExitAtStop or Volatile parameter is an array

```
// wrong - stop mode (precent/point) can not be array
ApplyStop( stopTypeLoss, IIf( C > O, stopModePercent, stopModePoint ), 5 );
```

Error 5. Argument #1 has incorrect type (the function expects different argument type here)

This error occurs when argument passed during function call has invalid type. For example when you pass string instead of array

```
MA( "test", 5 ); // wrong, Moving average expects array as first argument
AddColumn("Test", "Caption"); // wrong, AddColum expects array as first argument

AddTextColumn("Test", "Caption"); // correct, AddTextColumn expects text
```

Error 6. Condition in IF, WHILE, FOR statements has to be Numeric or Boolean type. You can not use array here, please use [] (array subscript operator) to access array elements

The if keyword executes statement1 if expression is true (nonzero); if else is present and expression is false (zero), it executes statement2. After executing statement1 or statement2, control passes to the next statement. Expression must be boolean (True/False) type (so it CANNOT be ARRAY because there would be no way do decide whether to execute statement1 or not, if for example array was:
[True,True,False,.....,False,True])

if(expression)
  statement1;
else
  statement2;

EXAMPLE

```
if( Close > Open ) // WRONG
    Color = colorGreen; //statement 1
else
    Color = colorRed; //statement 2

Plot(Close,"Colored Price",Color,styleCandle);
```

The above example is wrong, as both Open and Close are arrays and such expression as Close > Open is also an ARRAY. The solution depends on the statement. It’s either possible to implement it on bar-by-bar basis, with use of FOR loop:

```
for( i = 0; i < BarCount; i++ )
{
   if( Close[ i ] > Open[ i ] ) // CORRECT
       Color[ i ] = colorGreen;
   else
       Color[ i ] = colorRed;
}

Plot( Close, "Colored Price", Color, styleCandle );
```

It is also possible in this case to use IIf( ) function:

```
Color = IIf( Close > Open, colorGreen, colorRed ); // ALSO CORRECT - working
directly on arrays
Plot( Close, "Colored Price", Color, styleCandle );
```

Error 7. Condition in IF, WHILE, FOR statements has to be Numeric or Boolean type. You can not use STRING here.

Occurs when you attempt to use string as a condition in if/while/for statements.

For example:

```
if( "text" ) // incorrect
{
   // do something
   x = 1;
}
```

The condition in if/while/for should evaluate to true/false:

```
if( "text" != "someothertext" ) // correct, != (not equal to) operator gives
true/false value
{
   // do something
   x = 1;
}
```

Error 8. Type mismatch, the value assigned to the array element has to be a number. You can not use array on the right-side of this assignment.

Occurs on attempt to assign entire array to single element of another array

```
test = 0;
for( i = 0; i < BarCount; i++ )
{
    test[ i ] = Close ; // wrong, single array element can take only one value,
not array

    test[ i ] = Close[ i ]; // correct
}
```

Error 9. Array subscript has to be a number

You can use only numbers as array subscripts, strings and arrays are not accepted:

```
table[ 1 ] = 10; // correct
table[ "text" ] = 10; // incorrect
table[ Close ] = 10; // incorrect
```

Error 10. Subscript out of range. You must not access array elements outside 0..(BarCount-1) range

Occurs when you attempt to access array elements with subscripts below 0 (zero) or above BarCount-1.

```
// incorrect
for( bar = 0; bar < BarCount; bar++ )
{
   a[ bar ] = C[ bar - 1]; // when i == 0 we are accessing C[-1] which is wrong
}


// correct
for( bar = 0; bar < BarCount; bar++ )
{
   if( bar > 0 )
       a[ bar ] = C[ bar - 1 ];   // only access C[ i - 1 ] when i is greater
than zero
   else
       a[ bar ] = C[ 0 ];
}
```

One of most common mistakes is using hard coded upper limit of for loop and assuming that all symbols have enough quotes.

For example:

```
MyPeriod = 10;

for( i = 0; i < MyPeriod; i++ ) // WRONG ! this assumes that you always have at
least 10 quotes !
{
  // ... do something
}
```

This will always fail on symbols that have less quotes than 10 and it may also fail if you zoom your chart in so less than 10 quotes are actually displayed on the chart.

To ensure error-free operation you must always check for upper index being LESS than BarCount, like shown in the code below:

```
MyPeriod = 10;

for( i = 0; i < MyPeriod AND i < BarCount; i++ ) // CORRECT - added check for
upper bound
{
  // ... do something
}
```

Alternativelly you can enter the loop only when there are enough bars, like shown in this code:

```
MyPeriod = 10;
```

```
if( MyPeriod <= BarCount ) // CORRECT - check if there are enough bars to run the
loop
{
 for( i = 0; i < MyPeriod; i++ )
 {
   // ... do something
 }
}
```

Error 11. Subscript operator [] requires array or number type. String can not be used here.

Occurs when you attempt to use subscript operator [] on strings, for example:

```
tt = "Test";

x = tt[ 0 ];
```

Error 12. Subscript operator [] requires array or number type.

Occurs when subscript operator [] is applied to some other unsupported type (such as COM object dispatch)):

```
a=CreateObject("Broker.Application");
b = a[0];
```

Error 13. Endless loop detected in WHILE loop

Occurs when AFL engine detect the *while* loop that never ends (the detection is based on number of iterations, AmiBroker simply counts the number of iterations and if it exceeds the threshold limit set in **Tools->Preferences->AFL: Endless loop detection threshold -** by default 100000 iterations) it displays this message). Example:

```
i = 0;

while( i < 5 ) x = i; // i variable is not incremented, so the loop never ends.
```

Error 14. Endless loop detected in DO-WHILE loop

Occurs when AFL engine detect the *do-while* loop that never ends (the detection is based on number of iterations, AmiBroker simply counts the number of iterations and if it exceeds the threshold limit set in **Tools->Preferences->AFL: Endless loop detection threshold -** by default 100000 iterations) it displays this message). Example:

```
i = 0;

do
{
   x = i;
}
while( i < 5 ); // i variable is not incremented, so the loop never ends.
```

Error 15. Endless loop detected in FOR loop

Occurs when AFL engine detect the *for* loop that never ends (the detection is based on number of iterations, AmiBroker simply counts the number of iterations and if it exceeds the threshold limit set in **Tools->Preferences->AFL: Endless loop detection threshold -** by default 100000 iterations) it displays this message). Example:

```
for( i = 0; i < BarCount; i ) // forgotten ++ (increment operator) so the loop
never ends.
{
    x = i;
}
```

Error 16. Too many arguments

Occurs when too many arguments were specified when calling the function. For example:

```
m = MACD( 12, 26, 3 ); // error: MACD needs only 2 parameters, but 3 are
specified
```

Error 17. Missing arguments

Occurs when too few arguments were passed during function call. Example:

```
Plot( C, "Price" ); // too few arguments, 3rd argument is required - color of the
plot
```

Error 18. COM object variable is not initialized or has invalid type (valid COM object handle required)

Occurs on attempt to use uninitialized variable or variable of incorrect type to call COM object methods:

```
Obj = 1;  // initialize as number
Obj.Test(); // attempt to call method fails because Obj is not COM object handle
```

Error 19. COM method/function <function name> call failed. <More info>

An OLE exception occurred during OLE/COM method/function/property call. <More info> may contain OLE exception description.

Commonly this is displayed when arguments passed to the OLE/COM method are incorrect or missing.

```
AB=CreateObject("Broker.Application");
AB.Import(); // <-- fails with error 19. In this case because of missing
arguments
```

Error 21. Relative strength base symbol not found

RelStrength() function called with non-existing symbol as a parameter:

```
x = RelStrength("NonExistingTicker"); // fails because of wrong symbol
```

Error 22. Bad 'format' argument type - format has to be a number (not array)

Format parameter specified in AddColumn() function call should be a number (not array). Example

```
AddColumn( C, "Close", IIf( C > 10, 1.2, 1.3 ) ); // wrong, format parameter
(3rd) has to be a number

AddColumn( C, "Close", 1.3 ); // correct
```

Error 23. GetExtraData call failed

GetExtraData() failed (or returned no value) either because current plugin does not support GetExtraData function or field specified is not supported by the plugin:

```
x=GetExtraData("nonexistingfieldname"); // wrong field name
```

Error 24. This formula requires AmiBroker version <...> (or higher).

The formula is intended to be used on higher version of AmiBroker and you should upgrade in order to use it.

```
Version(9.7); // there is no version 9.70 yet :-)
```

Error 25. SetVariable() called from plug in was not successful. Identifier is already used in a different context.

Happens if external plugin attempts to call SetVariable with identifier that is already used for some other purpose
(like built-in or user-defined function)

Error 26. File handle passed to the function is invalid (equal to zero). You have to check if file handle returned by fopen is not equal to zero. If it is zero it means that file could not be opened.

Occurs on attempt to call file write/read/close function on null file handle.

Example (incorrect):

```
fh = fopen("nonexistingfile.txt", "r"); // this file does not exist

fputs("Test", fh ); // error here, fh could be null
fclose( fh );  // wrong, fh could be null
```

Correct usage would look like this:

```
fh = fopen("nonexistingfile.txt", "r"); // this file does not exist

if( fh ) // correct, call subsequent file read/write/close only when file handle
is OK
{
   fputs("Test", fh );
   fclose( fh );
}
```

Error 27. Invalid number of arguments passed to Call Function (..) from plugin DLL

Occurs only when external plugin calls internal AmiBroker functions with incorrect number of arguments.

Error 28. Out of memory

Out-of-memory error occurred during parsing the formula (should not happen under normal circumstances)

Error 29. Variable <name> used without having been initialized.

You can not use (read) the variable that was not initialized first. You should assign the value to the variable before reading it.

Example (incorrect usage):

```
x = 1;
z = x + y; // wrong, y is not initialized
```

Correct usage would look like this:

```
x = 1; // initialize x
y = 2; // initialize y
z = x + y; // correct, both x and y are initialized
```

Error 30. Syntax error

General syntax error. Occurs when the syntax is incorrect (for example unbalanced parentheses, or unrecognized characters or invalid operators or incorrect/undefined function name or missing brace or semicolon)

```
x = 4;
y = 2;

z = x * ( 7 + y ; // syntax error here because of missing closing parenthesis
```

Error 31. Syntax error, expecting <list of tokens>

The syntax is incorrect because the parser expects specifiec tokens and finds something else.

For example:

```
while i < 5 // this generates Error 31. Syntax error, expecting '('
```

- the parser expects opening brace '(' after *while* statement

It may also occur if some closing brace, closing double quote or semicolon is missing in one of preceding lines and parser is looking for matching brace/quote until it finds something that does not really belong to previous unbalanced statement.

Error 32. Syntax error, probably missing semicolon at the end of the previous line

General syntax error occurred at the beginning of the line. In most cases it happens because of missing semicolon at the end of the statement in the previous line, see this:

```
a=5
b=4; // <--- here syntax error probably missing semicolon
```

but on some occassions the reason may be simply incorrect syntax at the beginning of the line:

```
a=5;
+b=4; // <-- the same error message but the problem is about an extra + character
at the beginning of the line
```

Error 33. The identifier is already in use. You attempted to assign value to the identifier representing a function. If you want to return value from function you should use RETURN statement instead.

Example 1:

```
function Test( x )
{
   return 2 * x;
}

VarSet( "Test", 7 ); // error, identifier 'Test' is already used for function
```

Example 2:

(incorrect)

```
function Test( x )
{
 Test = 2 * x; // error here because Test identifier is already used for
function,
 // you should use return statement to return values from function
}

x = Test(5);
```

Correct function returning value would look like this:

```
function Test( x )
{
    return 2 * x; // correct, returning values using return statement
}

x = Test(5);
```

Error 34. The identifier is already in use. You attempted to define the function that has the same identifier as global variable.

Occurs when function definition uses the same identifier as global variable defined earlier in the formula.

```
Test = 5; // global variable

function Test( x ) // incorrect, 'Test' identifier is already used for global
variable
{
 return 2 * x;
}
```

Error 35. Shift+BREAK pressed. Loop terminated.

Occurs when user manually terminates loop execution by pressing Shift+BREAK keys.

Error 36. N-th argument of the function call has no value set

May happen if N-th argument of the function is set to no value. This can not happen under normal circumstances, but only when calling functions from inside the plugin or by setting variables from inside plugin.

Error 37. Unsupported field in SetOptions

Occurs when wrong (or not supported) field name was used in SetOption call, for example:

```
SetOption("NoSuchOption", 1 );
```

Error 38. Unsupported field in GetOptions

Occurs when wrong (or not supported) field name was used in GetOption call, for example:

```
x = GetOption("NoSuchOption");
```

Error 39. CategoryAddSymbol: Setting sector is unsupported, set industry instead

Occurs when categorySector is used in CategoryAddSymbol function. You can not add symbol to sector because symbols are linked to industry groups only and industry groups are then assigned to sectors. One sector usually includes several industry groups and if you set the industry then sector is implict. Refer to AmiBroker Users Guide for more information about categories.

```
CategoryAddSymbol( "", categorySector, 2 ); // wrong, you can not use
categorySector

CategoryAddSymbol( "", categoryIndustry, 2 ); // correct
```

Error 40. CategoryRemoveSymbol: Removing from sector is unsupported, remove from industry instead

Occurs when categorySector is used in CategoryRemoveSymbol function. You can not remove symbol from sector because symbols are linked to industry groups only and industry groups are then assigned to sectors. One sector usually includes several industry groups and if you set the industry then sector is implict. Refer to AmiBroker Users Guide for more information about categories.

```
CategoryRemoveSymbol( "", categorySector, 2 ); // wrong, you can not use
categorySector

CategoryRemoveSymbol( "", categoryIndustry, 2 ); // correct
```

Error 41. Unsupported field in GetRTData

Occurs when not supported field was specified in GetRTData call:

```
GetRTData("EPSRank"); // EPSRank is not available from RT sources
```

Note that GetRTData is supported only for real-time data sources and in Professional edition only. If you attempt to call it without running RT data source or using AmiBroker Standard edition, it will quietly return Null value without displaying any error message.

Error 42. #include failed because the file does not exist: <filename> (current working directory is '...')

Occurs when specified include file does not exist. Example:

```
#include "not\existing\path\to\the\file.afl"
```

Error 43. Variable stops are not supported in Rotational and Raw Trading modes

Occurs on attempt to use variable amount in ApplyStop() function when using rotational trading backtester mode or in one of Raw modes (such as backtestRegularRaw or backtestRegularRawMulti). Example:

```
EnableRotationalTrading();

ApplyStop( stopTypeLoss, stopModePoint, H-L ); // variable stop amount not
supported in rotational mode
```

Error 44. SectorID() is outside 0..63 range.

May occur during call to SectorID() function if external data plugin sets sector IDs incorrectly (outide 0..63 range)

```
x = SectorID(); // the formula is correct, but may fail with error 44 if data
plugin sets sectors incorrectly
```

Error 45. Failed to lauch trading interface

Problem occurs when formula calls GetTradingInterface but required interface is not installed or registered properly.

```
ti = GetTradingInterface("DUMMY"); // fails because DUMMY interface is not
installed
```

Error 46. Missing comma

Problem occurs when there is a missing comma in the function declaration formal parameter list

```
function MyFun( x y ) // missing comma in the formal parameter list
{
   return x * y;
}
```

Error 47. Exception occurred during AFL formula execution

This error occurs when formula caused unhandled system exception. System exception may be due to memory overflow, accessing incorrect file handle, memory access violation, etc.

Example:

```
fclose( 123 ); // closing invalid file handle causes system exception
```

Error 48. N-volume bar compressed data longer than base time frame. Use higher compression factor.

This error occurs when N-volume compression setting produces data longer than original data set.

N-volume bars are very different from time-based bars(compression of data to N-volume bar may actually deliver MORE output bars - for example if one tick is 1000 shares and you have specified 100V bars then single tick will be expanded to TEN 100V bars - ten times original size)
TimeFrame functions are protected against array overrun and will not decompress beyond original array size (you will get an "Error 48"). Also switching main time frame to some weird N-volume bar value will result in limiting the output to maximum twice original data size(without error message).

You should keep that in mind and avoid using too small N-volume bar intervals that could lead to such condition. Also due to the nature of N-volume bars the only TimeFrameSet() function will yield correct N-volume bar values, TimeFrameGetPrice() may give slightly distorted results.

Example:

```
TimeFrameMode( 2 );
TimeFrameSet( 20 ); // possible Error 48 - 20-share bar compression may be too
small
```

Error 49. Optimization parameter name must not be empty.

This error occurs when Optimize() function is called with empty name argument.

Example:

```
period = Optimize("", 10, 10, 20, 1 ); // WRONG: name must NOT be empty
```

Error 50. Optimization parameter minimum value must be less than or equal to maximum and step parameter needs to be greater than zero

This error occurs when AFL's Optimize() function is called with minimum value greater than maximum or step less not greater than zero

Example:

```
period = Optimize("Period", 1, 20, 10, 1 ); // WRONG: minimum > maximum
period2 = Optimize("Period2", 1, 10, 20, 0 ); // WRONG: step = 0
```

Error 51. Array subscript has Null value. Subscript must be within 0...BarCount-1 range.

You attempted to use Null value as array subscript

```
index = Null;

value = Close[ index ]; // error 51.
```

Error 52. Invalid argument value for <name> function. Argument <paramname> must be positive (and not Null).
Error 52. Invalid argument value for <name>() function. The function does not support <paramvalue> specified.

You attempted to pass a negative value as a parameter to a function that expects positive values or the function does not accept given value of parameter.

```
Sum( Close, -15 ); // error 52 - range must be positive
```

Error 53. You have N open file(s) that you forgot to close. You must call fclose() function for every file opened with fopen()

You called fopen() to open files but failed to call fclose.

```
fh = fopen("test.txt", "w" );
if( fh )
{
   // fclose( fh ); // fclose() SHOULD be called
}
```

Error 54. Incorrect \ espace sequence. Supported sequences are \n, \r, \t, \" and \\ (gives single backslash)

You have specified incorrect espace sequence in the string constant. Supported sequences are \n - for new line, \r - for carriage return, \t - for tab, \" for quotation mark, \\ - single backslash.

```
fh = fopen("C:\windows\test.txt", "w"); // Error 54 - incorrect espace sequence
WRONG - a single backslash should be encoded as \\
fh = fopen("C:\\windows\\test.txt", "w" ); // CORRECT
fclose( fh );
```

Error 55. Invalid assignment. The identifier is read-only (constant) and can not be written to.

This happens when you are trying to assign a value to read-only identifier (constant), like for example

```
colorBlack = 1200; // WRONG ! colorBlack is a constant and can NOT be written to
```

This may also occur if you make such assignment as a part of function call:

```
Plot( Close, "Price", colorRed = 7 ); // WRONG ! assignment of read-only
identifier (colorRed) within function call
```

Error 56. Variable <*name*> has an invalid type

This may happen if variable has unknown (unsupported) type. Supported types are: number, array, string, object, file handle, matrix. Normally this error does not show up because AmiBroker internally does not use unknown types. This may however occur if you are using 3rd party DLL that is returning values that have type of AmiVar structure set incorrectly (or not set at all). In that case contact 3rd plugin vendor for the fix. The error may also happen if something gets wrong and variable is left in some random/uninitialized state.

Error 57. Invalid quotation mark. Replace it with straight double quote ("").

This error means that you have copy-pasted code from somewhere (like web page) and pasted code contains incorrect quotations marks. Only straight quotation marks " (US ASCII code 0x22) should ever be used in the code/formula.

Unfortunatelly some web pages and some browsers automatically change quotation marks to various other characters that look like quotation marks but they are technically different (use different code). There are actually 29 different codes for different quotation marks used on the Internet.

```
Title =  This is a text with weird (incorrect) quotation marks ; // WRONG
```

```
Title = "This is a text with correct quotation marks";
```

To fix the problem you need to replace quotation marks with straight double quote (on US keyboard you press the comma key with SHIFT)

Error 58. Expecting closing quote (`') in character literal.

This error occurs when you have entered single-character literal without closing apostrophe. Single-character literals are used instead of ASCII codes of letters, like 'a' for ASCII code of letter a. They can be used for example in AddColumn statements to display single-letter codes instead of numbers:

```
Buy = Cross( C, MA( C, 10 ) );

AddColumn( IIf( Buy, 'B', 'S' ), "Trade", formatChar ); // correct

AddColumn( IIf( Buy, 'B', 'S ), "Trade", formatChar ); // incorrect - missing '
apostrophe after S
```

Error 59. Too many subscripts

This error occurs when you are trying to use triple subscript (three dimensional subscript) on two dimensional matrix.

```
mx = Matrix( 2, 3 );

x = mx[ 1 ][ 2 ][ 3 ]; // mx has 2 dimensions only, you can not use 3rd subscript
operator
```

Error 60. Requested matrix size is too large (>2GB)

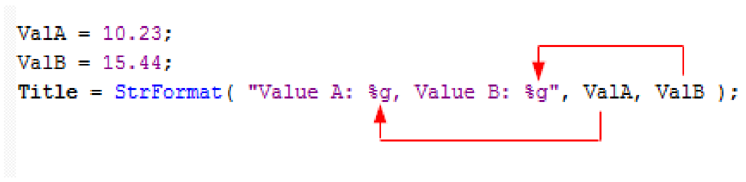
This error occurs when you are trying to create a matrix that would require more than 2GB of RAM. Since single-precision floating point value takes 4 bytes, it means that you can only create matrices of up to 500 million elements (rows * cols). Please also note that 32-bit process has its working memory limited to 2GB for the entire process, so when you want to work with large matrices, use 64-bit version.

```
mx = Matrix( 30000, 20000 ); // WRONG this attempts to create 30K*20K matix (600
million elements - too much)
```

Error 61. The number of format specifier(s) (%) does not match the number of arguments passed.

This error means that you have incorrect formatting sequence in printf/StrFormat call and/or the number of arguments passed to the function does not match the number of % specifiers in the formatting string.

When you call printf/StrFormat the very first string is a formatting string that tells how subsequent arguments are formatted for output. It is important for the list of subsequent arguments to match the number of % specifiers in the formatting string. In cases, where there is no match - AmiBroker will display Error 61.



```
ValA = 10.23;
ValB = 15.44;
Title = StrFormat( "Value A: %g, Value B: %g", ValA, ValB );
```

The most common source of problem is forgetting the fact that actual percent sign must be typed as %% (double percent signs) because single percent sign is a start of formatting sequence, not actual percent sign.

```
printf("Actual percent sign is not %." ); // WRONG use % instead of %%
printf("Actual percent sign is %%." ); // correct use of %%
```

Error 62. Formatting string contains unsupported sequence. Only floating point %g, %f and %e are supported

This error means that you have incorrect formatting sequence in printf/StrFormat call.

printf/StrFormat functions use the very first string parameter as a formatting string that decides how to display subsequent arguments. Currently you can only pass numbers and arrays as 2nd and subsequent arguments to printf/StrFormat so you can only use floating-point format specifiers, %f, %g, %e. Other format specifiers are not supported and would generate error 62.

```
printf("Close price with 4 decimal digits %.4f\n", Close ); // OK
printf("Close price as integer (without decimal places) %.0f\n", Close ); // OK
printf("Close price as integer %d\n", Close ); // WRONG, unsupported %d sequence
```

The other most common source of problem is forgetting the fact that actual percent sign must be typed as %% (double percent signs) because single percent sign is a start of formatting sequence, not actual percent sign.

```
printf("Actual percent sign is not %." ); // WRONG use % instead of %%
printf("Actual percent sign is %%." ); // correct use of %%
```

Error 90. Specified Optimizer Engine not found

OptimizeSetEngine function was called with non-existing engine name

```
OptimizerSetEngine("test"); // Error 90 - such engine does not exist
```

Error 91. OptimizerSetOption expects STRING value

OptimizerSetOption function expects string for given optionargument

```
OptimizerSetOption("a_string_field", 1 ); // Error 91 - incorrect argument
```

Error 92. OptimizerSetOption expects NUMBER (scalar numeric) value

OptimizerSetOption function expects number for given option argument

```
OptimizerSetOption("a_numeric_field", "string"); // Error 91 - incorrect argument
```

Error 93. Specified option is not supported / not available by the selected optimizer

OptimizerSetOption function called with unsupported option name.

```
OptimizerSetOption("NoSuchOption", 1); // Error 93 - no such option in external
engine
```

Error 94. External Optimizer is not selected. You must call OptimizerSetEngine() prior to calling OptimizerSetOption()

You must call OptimizerSetEngine before calling any other Optimizer... functions.

```
// OptimizerSetEngine("cmae"); // Not called but SHOULD BE
OptimizerSetOption("runs", 1); // Error 94 - you must call OptimizerSetEngine
before callign OptimizerSetOption
```

Warning 501. Assignment within conditional. Did you mean == instead of = ?

You attempted to do assignment inside if/while/for statements or inside IIF function. This may be a mistake because chances are that you meant to compare numbers not to assign them. Remember that == is equality check, not =. By using = you don't compare but assign a value to variable. If you really mean to assign inside conditional statement use extra braces.

```
if( x = 5 ) // warning, you are assigning 5 to variable x
{

}

if( x == 5 ) // correct - you probably meant to COMPARE instead (note ==
operator)
{

}
```

Warning 502. You are calling Plot()/PlotOHLC() function over 500 times, it is highly inefficient. Reduce number of calls.

You are calling Plot()/PlotOHLC() function over 500 times, it is highly inefficient. Reduce number of calls.

It is important to understand that Plot() function involves plotting entire chart and that is rather costly. If you want to draw segmented line generated using for example LineArray, don't call Plot() multiple times. Instead combine all LineArrays into one and use single Plot call.

```
for( i = 0; i < 600; i++ )
{
 x0 = ..;
 x1 = ..;
 y0 = ..;
 y1 = .. ;
 Plot( LineArray( x0, x1, y0, y1 ), "", colorRed ); // DO NOT DO THIS !
}
```

Instead use single Plot():

```
CombinedLine = Null;
for( i = 0; i < 600; i++ )
{
 x0 = ..;
 x1 = ..;
 y0 = ..;
 y1 = .. ;

 La = LineArray( x0, x1, y0, y1 );

 CombinedLine = IIf( IsNull( la ), CombinedLine, la ); // combine lines
}

Plot( CombinedLine, "", colorRed ); // THAT IS PROPER WAY - one call to Plot that
does all the plotting in one shot
```

For more examples see also http://www.amibroker.org/userkb/2007/04/20/plotting-trade-zigzag-lines/

Warning 503. Using OLE / CreateObject / CreateStaticObject is not multi-threading friendly.

Your formula contains CreateObject and/or CreateStatic object calls. While they are still supported, the performance of OLE in multi-threaded applications is poor, and you should consider replacing OLE with native AFL commands that do the same.

```
AB = CreateObject("Broker.Application"); // OLE is slow
```

AmiBroker fully supports calling OLE objects from AFL formula level, and it is still safe to use, but there are technical reasons to advocate against using OLE. The foremost reason is that OLE is slow especially when called not from "owner" thread.

OLE was developed by Microsoft back in 1990's in the 16-bit days it is old technology and it effectivelly prevents threads from running at full speed as all OLE calls must be served by one and only user-interface thread. For more details see this article:
http://blogs.msdn.com/b/oldnewthing/archive/2008/04/24/8420242.aspx

For this reason, if only possible you should strictly avoid using OLE / CreateObject in your formulas.

If you fail to do so, the performance will suffer. Any call to OLE from a worker thread causes posting a message to OLE hidden window and waiting for the main application UI thread to handle the request. If multiple threads do the same, the performance would easily degrade to single-thread level, because all OLE calls are handled by main UI thread anyway.

You should consider replacing OLE with native AFL commands that do the same.

For more details see Efficient use of multithreading

Error 701. Missing buy/sell variable assignments.

The formula that you are trying to backtest does not contain proper Buy and Sell rules. Buy and Sell rules should be written as assignments as shown below:

```
Buy = Cross( Close, MA( Close, 50 ) );
Sell = Cross( MA( Close, 50 ), Close ) );
```

For more details see Tutorial: Backtesting your trading ideas

Error 702. Missing short/cover variable assignments.

The formula that you are trying to backtest does not contain proper Short and Cover rules. Short and Short rules should be written as assignments as shown below:

```
Short = Cross( MA( Close, 50 ), Close ) );
Cover = Cross( Close, MA( Close, 50 ) );
```

If you do not want to test the short side, please go to the Settings and select **Long** from **Positions** combo-box. This will allow you to test long side only.

For more details see Tutorial: Backtesting your trading ideas

Error 703. Rotational trading requires PositionScore variable.

When trying to use the Rotational backtest mode you must define PositionScore variable that decides how symbols are ranked and sorted. Example PositionScore assignment is presented below:

```
EnableRotationalTrading();
SetOption("WorstRankHeld",5);

PositionSize = -25; // invest 25% of equity in single security
PositionScore = 50 - RSI(); // THIS IS REQUIRED for rotational mode
```

For more details see Rotational Trading mode

Error 704. In Rotational trading you MUST NOT use buy/sell/short/cover signals.

When trying to use the Rotational backtest mode you must **not** use buy/sell/short/cover variable assignments

```
EnableRotationalTrading();
SetOption("WorstRankHeld",5);

PositionSize = -25; // invest 25% of equity in single security
PositionScore = 50 - RSI(); // required for rotational mode

Buy = Cross( C, MA( C, 50 ) ); // WRONG - can not use that in rotational mode
```

On the other hand, if you want to use Buy/Sell/Short/Signals and just prioritize them - use regular backtest mode. For that, remove EnableRotationalTrading function call.

For more details see Rotational Trading mode

Error 705. You can not use HoldMinBars together with Allow Same Bar exit

You can not mix HoldMinBars option with AllowSameBarExit

```
SetOption("AllowSameBarExit", True );
SetOption("HoldMinBars", 5 ); // ERROR - these two do not mix
```

Error 706. Show Arrows feature needs a Trade list.

You attempted to use "Show arrows" feature (or double clicked on Analysis result list) but AmiBroker can not display trading arrows unless you run backtest with **Report** mode set to **Trade List.** You would need to go to the Settings window, **Report** tab, and change **Result list shows** to **Trade list** (the default setting).

For more information see Tutorial: Using New Analysis window and the Settings window

Error 707. Selected Optimizer Engine not found or failed to initialize

Optimizer engine specified by OptimizerSetEngine() call failed to load or initialize.

Error 708. Too few optimization steps for non-exhaustive (smart) optimization. Remove OptimizeSetEngine() function call.

You attempted to use smart (non-exhaustive) optimization engine for too small search space (too few optimization variables and/or too few steps). Non-exhaustive optimizers need at least 100 steps (parameter combinations). Small problems should be solved using exhaustive optimization (just remove or comment out OptimizerSetEngine call).

```
OptimizerSetEngine("spso");
x = Optimize( "x", 1, 1, 10, 1 ); // just 10 step optimization - too few for PSO
Buy = Cross( C, MA( C, x ) );
Sell = 0;
```

Error 709. Optimization search space is too large (more than 10^{19} combinations). Decrease number of Optimize() calls.

You have attempted to use optimization on search space larger than 10 000 000 000 000 000 000 combinations using exhaustive method. You need to reduce number of Optimize() statements and/or increase step size and/or reduce min-max range of optimization parameters in order to reduce search space size.

```
x = Optimize( "x", 1, 1, 1000, 1 ); // 10^3
y = Optimize( "y", 1, 1, 1000, 1 ); // 10^6
z = Optimize( "z", 1, 1, 1000, 1 ); // 10^9
i = Optimize( "i", 1, 1, 1000, 1 ); // 10^12
j = Optimize( "j", 1, 1, 1000, 1 ); // 10^15
k = Optimize( "k", 1, 1, 1000, 1 ); // 10^18
m = Optimize( "m", 1, 1, 1000, 1 ); // 10^21 -- TOO MUCH
```

You may try using smart (non-exhaustive) method instead although please keep in mind that such large search spaces are difficult to search and you are not very likely to find global minimum.

Error 710. Range Start date must not be greater than End date

You have attempted to run Analysis using "From-to" range, with **Start/From** date being greater than **End/To** date. Please correct the dates so **Start** is equal or earlier than **End** date.

Error 711. Optimization target selected in the Settings->Walk Forward page can not be found in the result list

You have entered Optimization target in the Analysis, Settings, Walk Forward tab that can not be found in the result list. It is usually because the user-defined optimization target is not generated by your custom backtest formula. If you want to use custom metrics as optimization target you have to calculate and add them to your backtest report as explained in Tutorial - Using custom metrics

If you define your own metric, please make sure that you enter exactly the same name (case sensitive) in the "optimization target" field in Walk forward tab. If there is mismatch, AmiBroker won't find your custom metric and would display this error message.

Error 712. PointValue must be greater than zero. Symbol: ...

You have attempted to run backtesting on symbol(s) that have incorrectly set PointValue. PointValue must be greater than zero as it is used as price movement multiplier. When PointValue was zero then all profits/losses would be zero regardless of price movement and that would mean completely useless backtest results.

Calculating multiple-security statistics with AddToComposite function

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

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

This is the area where AddToComposite function can help.

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

2.3 The solution

The key to the solution is the following algorithm:

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

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

Step 2 described above is implemented entirely inside AddToComposite function:

SYNTAX	AddToComposite(array, "ticker", "field", flags = atcFlagDefaults)
RETURNS	NOTHING
FUNCTION	Allows you to create composite indicators with ease. Parameters: array - the array of values to be added to "field" in "ticker" composite symbol "ticker" - the ticker of composite symbol. It is advised to use ~comp (tilde at the beginning) newly added composites are assigned to group 253 by default and have "use only local database" feature switched on for proper operation with external sources possible field codes: "C" - close , "O" - open, "H" - high, "L" - low, "V" - volume, "I" - open interest, "X" - updates all OHLC fields at once, "1" - aux1 field, "2" - aux2 field flags - contains the sum of following values

	• atcFlagResetValues = 1 - reset values at the beginning of scan (recommended) • atcFlagCompositeGroup = 2 - put composite ticker into group 253 and EXCLUDE all other tickers from group 253 (avoids adding composite to composite) • atcFlagTimeStamp = 4 - put last scan date/time stamp into FullName field • atcFlagEnableInBacktest = 8 - allow running AddToComposite in backtest/optimization mode • atcFlagEnableInExplore = 16 - allow running AddToComposite in exploration mode • atcFlagResetValues = 32 - reset values at the beginning of scan (not required if you use atcFlagDeleteValues) • atcFlagEnableInPortfolio = 64 - allow running AddToComposite in custom portfolio backtester phase • atcFlagDefaults = 7 (this is a composition of atcFlagResetValues | atcFlagCompositeGroup | atcFlagTimeStamp flags) AddToComposite function also detects the context in which it is run (it works ONLY in scan mode, unless atcFlagEnableInBacktest or atcFlagEnableInExplore flags are specified) and does NOT affect composite ticker when run in Indicator or Commentary mode, so it is now allowed to join scan and indicator into single formula.

EXAMPLE	AddToComposite(MACD() > 0, "~BullMACD", "V");
	graph0 = Foreign("~BullMACD", "V");
	
	(now you can use the same formula in scan and indicator)

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

Example 1:

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

```
/* AddToComposite statements are for analysis -> Scan */
/* add Close price to our index OHLC fields */
AddToComposite(Close, "~MyIndex", "X" );

/* add one to open intest field (we use this field as a counter) */
AddToComposite( 1, "~MyIndex", "I" );

buy = 0; // required by scan mode

/* this part is for Indicator */
graph0 = Foreign( "~MyIndex", "C" )/Foreign( "~MyIndex", "I" );
```

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

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

Example 2:

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

So the first line of our formula will be:

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

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

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

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

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

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

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

Example 3:

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

First we should have our original trading system formula:

```
      /* Your original formula here */
      /* In this example this is simple macd/signal crossover system)

      buy = cross( macd(), signal() );
      sell = cross( signal(), macd() );

      /* the following line uses Flip function to get "1" after the buy
      signal and reset it back to "0" after sell appears. */

      in_trade = flip( buy, sell );

      AddToComposite( in_trade, "~OpenPosCount", "V" );
```

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

Use

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

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

2.4 Notes

For mode details on composites check "Introduction to AddToComposite" (122KB PDF) by Herman van den Bergen.

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

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

Equity function, Individual and Portfolio Equity Charts

Introduction

The equity line is probably the best diagnostic tool for trading system developer. In one graph it shows the sum total of the success or failure of the system being tested, and the resulting effect on your equity.

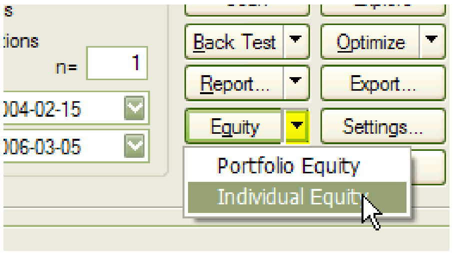
Numbers from Report telling of a drawdown is nice but with a graph, one can see how things were going before, during, and after a drawdown.

The line produced by the equity function tracks the amount of equity in your account. So, for example, if you backtest a system over the last year, you should see that, at the beginning of that year, the line starts at the amount of your your initial equity, and then rises and falls because, as each trade makes or loses money, the equity in your account will rise and fall. It shows how much money is in your account throughout the backtest period, so you can visually see how your system performed.

So, clearly you dont want to see the line go down - that means you lost money. It is generally accepted that you want to revise your system test parameters in order to get as close as possible to a smooth, straight, rising line. This means that your system has performed consistantly over time, and presumably over different market conditions. A line that goes up and down frequently means that your system works well only under certain conditions, and poorly under other conditions.

Individual (single-security) Equity chart

To display single security Equity chart is is enough to click on the drop down arrow on the "Equity" button and choose "Individual Equity" from the menu in the Automatic Analysis window AFTER running a backtest.



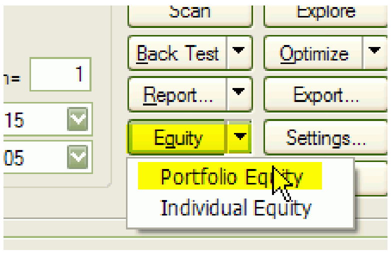
This will plot the equity for currently active symbol and recently backtested system. If you want to see the Equity curve for another symbol - just switch to this symbol and Equity line will be recalculated automatically.

You can also choose symbol that was not included in the original backtest set and AmiBroker will calculate correct equity curve as it would look if real backtest was performed on it.

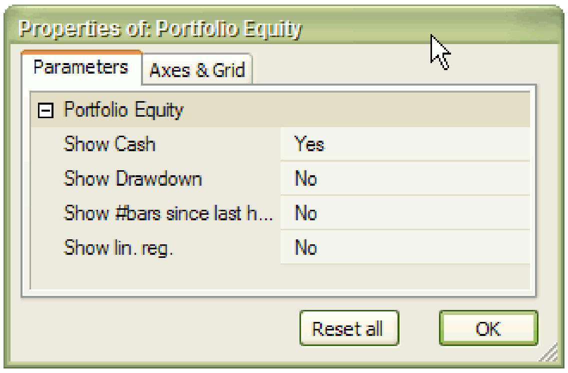
IMPORTANT: individual equity chart is single-security equity that does not show portfolio-level effects like skipping some of trades due to reaching maximum open position limit or funds being allocated to other securities, it also does not use some advanced functionality offered only by portfolio-level backtester. For more information see this.

Portfolio-level Equity chart

To display portfolio-level equity chart is is enough to double click on "Equity" button in the Automatic Analysis window or click on the drop down arrow on the "Equity" button and choose "Portfolio Equity" from the menu AFTER running a backtest.



Portfolio-level equity represents equity of your entire portfolio and reflects ALL real-world effects like skipping trades due to insufficient funds, reaching maximum number of open positions. It also reflects all scaling in/out, HoldMinBars effect, early exit fees and any other feature that you may have been using in your formula. Portfolio-level equity also by default shows the remaining cash in the portfolio. Using Parameters window (click with RIGHT mouse button over equity chart and select "Parameters" from the menu) you can turn on display of drawdown (underwater equity curve), number of bars sincel last equity high and linear regression of the equity.


Properties of: Portfolio Equity

Parameters | Axes & Grid

Portfolio Equity

Show Cash	Yes
Show Drawdown	No
Show #bars since last h...	No
Show lin. reg.	No

Equity function

Equity() function is a single-security backtester-in-a-box. It has many interesting applications that will be outlined here. Let's look at the definition of Equity function:

SYNTAX	**equity**(*Flags* = 0, *RangeType* = -1, *From* = 0, *To* = 0)
RETURNS	ARRAY
FUNCTION	Returns Equity line based on buy/sell/short/cover rules, buy/sell/short/coverprice arrays, all apply stops, and all other backtester settings. *Flags* - defines the behaviour of Equity function **0** : (default) Equity works as in 3.98 **-** just calculates the equity array **1** : works as 0 but additionally updates buy/sell/short/cover arrays so all redundant signals are removed exactly as it is done internally by the backtester plus all exits by stops are applied so it is now possible to visualise ApplyStop() stops. **2** : (advanced) works as 1 but updated signals are not moved back to their original positions if buy/sell/short/cover delays set in preferences are non-zero. Note: this value of flag is documented but in 99% of cases should not be used in your

formula. Other values are reserved for the future.

RangeType - defines quotations range being used:

-1 : (default) use range set in the Automatic analysis window
0 : all quotes
1 : n last quotes (n defined by 'From' parameter)
2 : n last days (n defined by 'From' parameter)
3 : From/To dates

From : defines start date (datenum) (when RangeType == 3) or "n" parameter (when RangeType == 1 or 2)

To: defines end date (datenum) (when RangeType == 3) otherwise ignored

datenum defines date the same way as DateNum() function as YYYMMDD where YYY is (year - 1900), MM is month, DD is day

December 31st, 1999 has a datenum of 991231
May 21st, 2001 has a datenum of 1010521

All these parameters are evaluated at the time of the call of Equity function. Complete equity array is generated at once. Changes to buy/sell/short/cover rules made after the call have no effect. Equity function can be called multiple times in single formula.

EXAMPLE

buy = your buy rule;
sell = your sell rule;
graph0 = Equity();

SEE ALSO

Using Equity function we can build up Equity "indicator" that will work without the need to run backtester. Just type the following formula in the Formula Editor and press Apply:

```
buy = ... your buy rule ...
sell = .... your sell rule ...

graph0 = Equity();
```

Equity() function uses the buy/sell/short/cover rules that are defined BEFORE this function is called. The whole backtesting procedure is done inside Equity function that generates equity line.

Notes:

1. Equity line is dependant of the parameters in the Automatic Analysis settings
2. Equity traces Interest Earnings when you are OUT of the market.
   If you don't want this just enter 0 into "Annual interest rate" field in the settings.
3. Equity also traces commissions. If commissions are not zero entry commission is taken using position size of the entry and exit commission is taken for each point to simulate how much money would you have if you closed position at given bar.
4. AmiBroker uses SellPrice array for long trades and CoverPrice array for short trades to calculate current equity value.

5. Equity() function is single-security and does not reflect portfolio-level effects like skipping trades, does not handle some advanced functionality offered only by portfolio-backtester. For more information see that table.

Portfolio Equity special symbol

After running portfolio-level backtest, AmiBroker writes the values of portfolio equity to special symbol "~~~EQUITY". This allows you to access portfolio-level equity of last backtest inside your formula. To do so, use Foreign function:

```
PortEquity = Foreign("~~~EQUITY", "C" );
```

This is exactly what for example built-in portfolio equity chart formula is doing.

Can I calculate system statistics using Equity function?

Yes you can. Here is a couple of example calculations kindly provided by Herman van den Bergen:

```
      E = Equity();

      //How rich you were :-)
      Plot(Highest(E,"I should have sold",1,3);

      //Your Current Drawdown:
      Plot(Highest(E) - E,"Current DrawDown",4,1);

      //Trade profits:
      LongProfit = IIf(Sell,E - ValueWhen(Buy,E),0);
      ShortProfit = IIf(Cover,ValueWhen(Short,E)-E,0);
      Plot(IIf(Sell,LongProfit,0),"LProfit",8,2+4);
      Plot(IIf(Cover,ShortProfit,0),"SProfit",4,2+4);

      //Current Trade Profit:
      Lastbar = Cum(1) == LastValue( Cum(1) );
      Plot(IIf(LastBar AND pos,E-ValueWhen(Buy,E),ValueWhen(Short,E) -
      E),"Current Profit",9,2+4);

      //DailyProfits:
      Plot(IIf(pos,E-Ref(E,-1),Ref(E,-1)-E),"Daily Profits",7,2);
```

How do you plot a composite Equity Curve ? I don't want the whole database, but would like to see the curve based on the watchlist I use for the backtesting.

Just use Portfolio Level equity chart (see above). It represents actual portfolio backtest equity, so if you run your backtest on the watch list it will represent what you need. You can also write your own formula that does the same thing:

```
Plot( Foreign("~~~EQUITY", "C" ), "PL Equity", colorRed );
```

Functions accepting variable periods

The following functions support variable periods (where periods parameter can be array and change from bar to bar):

- AMA
- AMA2
- DEMA
- HHV
- HHVBars
- IIR
- LinRegSlope
- LinearReg
- LinRegIntercept
- LLV
- LLVBars
- MA
- PercentRank
- Ref
- StdErr
- StDev
- Sum
- TEMA
- TSF
- WMA

User-definable functions, procedures. Local/global scope

User-definable functions allow to encapsulate user code into easy-to-use modules that can be user in many places without 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.

A function definition establishes the name, and attributes (or parameters) of a function. A function definition must precede the call to the function. The definition starts with **function** keyword then follows function name, opening parenthesis then optional list of arguments and closing parenthesis. Later comes function body enclosed in curly braces.

A function call passes execution control from the calling function to the called function. The arguments, if any, are passed by value to the called function. Execution of a return statement in the called function returns control and possibly a value to the calling function.

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

Following is an example of function definition:

```
// the following function is 2nd order smoother

function IIR2( input, f0, f1, f2 )
{
    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 ];
    }

   return result;
}

Plot( Close, "Price", colorBlack, styleCandle );
Plot( IIR2( Close, 0.2, 1.4, -0.6 ), "function example", colorRed );
```

In this code **IIR2** is a user-defined function. **input, f0, f1, f2** are formal parameters of the functions.
At the time of function call the values of arguments are passed in these variables. Formal parameters behave like local variables.
Later we have **result** and **i** which are local variables. Local variables are visible inside function only. If any other function uses the same variable name they won't interfere between each other.

Due to the fact that AFL does not require to declare variables the decision whenever given variable is treated as local or global is taken depends on where it is FIRST USED.

If given identifier appears first INSIDE function definition - then it is treated as LOCAL variable.
If given identifier appears first OUTSIDE function definition - then it is treated as GLOBAL variable.

This default behaviour can be however overriden using **global** and **local** keywords (introduced in 4.36) - see example 2.

Example (commentary):

```
k = 4; // this is GLOBAL variable

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 local variable in
function f

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

// Now even if we call function
// the value of our global variable z
// is not affected by function call because
// 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 'return' statement that is used to return the result to the caller. Note that currently 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 );
}
```

Note that although there are two separate keywords 'function' and 'procedure' AmiBroker currently treats them the same (they both accept return values but not require them), but in the future the rules maight get enforced to use
return statement ONLY in conjunction with function keyword. So it is advised to use function keyword in case when your function returns any value and procedure keyword otherwise.

Note also that recursion (having a function call itself from within itself) is NOT supported as for now.

More information

Please read also Understanding how AFL works article to learn more.

AFL Tools

Automatic technical analysis

Introduction

Since version 2.5 AmiBroker features automatic technical analysis tools. AmiBroker can check for user defined buy/sell conditions giving you an idea about the current situation on the market. It can also perform a system test (simulation) telling you about the performance of your trading system. Version 3.0 of AmiBroker introduced new formula language (AFL) allowing you to write not only system tests but also custom indicators and guru advisor commentaries.

In order to do this you have to define buy and sell rules, indicator formulas or commentaries using a special *AmiBroker Formula Language (AFL)*, which is described below. For more information about using analysis tools see also the description of the *Automatic analysis window, Formula Editor, and Commentary window* in Chapter 2.

AmiBroker Formula Language

AFL is used for defining your trading rules and explorations in Automatic analysis window, custom commentaries in the Guru Commentary window and indicator formulas in Formula Editor window.

Detailed reference of AFL language is given here.

Examples

Below you will find some simple buy and sell rules. They are just formal equivalents of some of the most common indicators□ interpretation rules. You can treat them as a starting point for developing your own trading strategies, but before you get too excited when you think you've found the "holy grail" of trading systems, check the following:

- Test the system on different symbols and different time frames. The results should be similar to those on the original data tested.
- Test the system on different types of markets (e.g., upward trending, downward trending, and sideways). A good system should work in all types of markets, since you won't always know when a market changes from trending to trading or vice versa.
- Pay close attention to the number of trades generated by a trading simulation. If there are a large number of trades and large profits, be sure you specified realistic commissions. The results of the test may be much different once commissions are factored in.

```
buy = cross( macd(), 0 );
sell = cross( 0, macd() );

buy = cross( ema( close, 9 ), ema( close, 15 ) );
sell = cross( ema( close, 15 ), ema( close, 9 ) );

buy = cross( rsi(), 30 );
sell = cross( 70, rsi() );

buy = cross( ultimate(), 50 );
sell = cross( 50, ultimate() );
```

Automatic analysis window

Automatic analysis window enables you to check your quotations against defined buy/sell rules. AmiBroker can produce report telling you if buy/sell signals occurred on given symbol in the specified period of time. It can also simulate trading, giving you an idea about performance of your system.

In the upper part of window you can see text entry field. In this field you should enter buy and sell rules. These rules are assignment statements written in AmiBroker's own language. You can find the description of this language in AFL reference guide.

In order to make things work you should write two assignment statements (one for buy rule, second for the sell rule), for example:

```
buy = cross( macd(), 0 );
sell = cross( 0, macd() );
```

Automatic analysis window allows you also to optimize your trading system and perform in-depth explorations

See also: detailed description of Automatic Analysis window controls

Formula Editor

Formula Editor allows you to write formulas to be used as indicators or in Automatic Analysis window. More on this
here.

Guru Advisor Commentary window

Commentary window enables you to view textual descriptions of actual technical situation on given market. Commentaries are generated using formulas written in AmiBroker's own formula language. You can find the description of this language in AmiBroker Formula Language Reference Guide.

Moreover Commentary feature gives you also graphical representation of buy & sell signals by placing the marks (arrows) on the price chart.

NOTE: Using complex commentary formulas you may observe long execution times.

See also: detailed description of Guru Advisor Commentary window controls

AFL Scripting Host

IMPORTANT NOTE: Since the introduction of native looping and flow control statements like *if-else* and *while* in version 4.40 the significance of scripting has been greatly reduced. Currently most of the tasks requiring scripting in previous versions could be handled in native AFL. What's more AFL loops are 3-6 times faster than JScript/VBScript.

Basics

AFL scripting host is an interface between AFL engine and JScript/VBScript engines (aka. Active Scripting technologies) available as a part of Internet Tools & Technologies platform provided by Microsoft.
It allows you to build the formulas that are have parts in AFL code and parts in JScript/VBScript.

Requirements

- AmiBroker 3.59 or higher
- Microsoft JScript/VBScript engines installed

Microsoft JScript/VBScript engines come installed with Windows.

JScript/VBScript documentation can be found on official scripting page at:
http://msdn.microsoft.com/scripting/

Enabling AFL Scripting Host

If you want to use scripts within your formulas you have to call EnableScript() function in the beginning of your formula. The function takes one input parameter - engine name:

```
EnableScript("jscript");
```

or

```
EnableScript("vbscript");
```

From then on, you will be able to embody parts written in scripting language in your formulas. The begin and the end of the script must be marked with <% and %> sequences, as shown in the example below:

```
EnableScript("vbscript");

// "normal" AFL statements
buy = cross( macd(), 0 );
sell = cross( 0, macd() );

<%
..... your script code here.....
%>
```

```
        // "normal" AFL statements
        buy = ExRem( buy, sell );
```

Using variables

Currently the only way to exchange the information between "normal" AFL part and script part is to use variables. AFL scripting host exposes one object (predefined, no creation/initialization needed)
called **AFL**.

The AFL object has one (default) parametrized property called Var(varname) that can be used to access AFL variables from the script side:

```
// example in VBScript
       <%

buyarrayfromscript = AFL.Var("buy")
       ' get the buy array from AFL to the script-defined variable

AFL.Var("buy") = buyarrayfromscript
       ' set the buy array in AFL from the script-defined variable

%>
```

Since Var is default property you can omit its name and write simply AFL(varname) as shown in the example below:

```
<%

buyarrayfromscript = AFL("buy")
       ' gets the buy array from AFL to the script-defined variable

AFL("buy") = buyarrayfromscript
       ' sets the buy array in AFL from the script-defined variable

%>
```

In AFL there are three data types possible: array (of floating point numbers), a number (floating point) and a string. The VBScript and JScript engines use variant data type that can hold any type of the variable including three used by AFL. As in AFL, you don't declare variables in scripting languages, the type is determined by the first assignment. In case of VBScript you can get/set AFL variables of any supported type using syntax shown above. But in JScript, due to the fundamental difference in handling arrays in JScript (array elements in JScript are implemented as dynamic properites of an array object) you need to use the following code to get the value of AFL array into JScript array:

```
// example in JScript
<%
function GetArray( name )
{
   return VBArray( AFL( name ) ).toArray();
}

myJScriptArray = GetArray( "close" );
```

```
%>
```

The GetArray() function shown above makes it easy to convert automation-type safe array into JScript array. This example shows also how to define and use functions in JScript;

Assigning AFL variables from script-side arrays is much more simple, AFL scripting host detects JScript arrays and can get their contents directly:

```
// example in JScript
<%
   AFL("buy") = myJScriptBuyArray;
%>
```

All other data types are handled the same in JScript and VBScript

```
// example in VBScript
ticker = name();
<%
  tickerstring = AFL("ticker")
  AFL("ticker") = "new name"
%>
```

or in JScript:

```
// example in JScript
ticker = name();
<%
   tickerstring = AFL("ticker");
   AFL("ticker") = "new name";
%>
```

Iterating through arrays

One of the most basic task that everyone would probably do is to iterate through array. In VBScript this can be done using For..To..Next statement, in JScript using for(;;) statement. Both these constructs need to know array size or number of elements in the array. In VBScript you should use UBound(arrary) function to get the upper bound of the array, in JScript you just use length property of the array. The following examples show this. (Please remember that in both VBScript and JScript arrays are zero-based.)

```
// example in VBScript
<%
myArray = AFL("close")

sum = 0
for i = 0 to UBound( myArray )

sum = sum + myArray( i )

next

%>
```

or in JScript:

```
// example in JScript
```

```
<%
function GetArray( name )
{
   return VBArray( AFL( name ) ).toArray();
}

myArray = GetArray( "close" );

sum = 0;

for( i = 0; i < myArray.length; i++ )
{
       sum += myArray[ i ];
}

%>
```

Examples

a) Indicator example - Exponential moving average:

```
EnableScript("jscript");
<%

close = VBArray( AFL( "close" ) ).toArray();

output = new Array();

// initialize first element
output[ 0 ] = close[ 0 ];

// perform the loop that calculates exponential moving average
factor = 0.05;

for( i = 1; i < close.length; i++ )
{
  output[ i ] = factor * close[ i ] + (1 - factor) * output[ i - 1 ];
}


AFL.Var("graph0") = close;
AFL.Var("graph1") = output;

%>
WriteVal( graph1 );
```

b) Profit-target stop example

Here comes the example of the formula that realizes profit-target stop at the fixed 10% percentage from the buy price. Note that buy condition is met when the price reaches a new high, so it happens multiple times after initial buy. Therefore ValueWhen(buy, close) can not give you initial buy price and that kind of trading rule could not be implemented in AFL itself. But, with scripting there is no problem...

```
EnableScript("VBScript");
```

```
hh = HHV( close, 250 );

// buy when prices reaches a new high
buy = Close == HHV( close, 250 );

// ensure that sell is an array,
// sell = 0 would set the type to number
sell = buy;

<%
        close = AFL("close")
    buy = AFL( "buy" )
    sell = AFL("sell")

        ' this variable holds last buying price
    ' if it is zero it means that no trade is open
    lastbuyprice = 0

        ' iterate along the array
    for i = 0 to UBound( close )
       sell( i ) = 0

      ' Remove Buy signals if trade was already initiated
       if( lastbuyprice > 0 ) then
          buy( i ) = 0
       end if

      ' if there is no open trade and buy signal occurs
           ' get the buying price
       if ( lastbuyprice = 0 ) AND (buy( i ) = 1) then
          lastbuyprice = close( i )
       end if

      ' if trade is open and sell condition is valid
       ' generate sell signal
       ' and close the trade
       if (lastbuyprice >0 ) AND ( close( i ) > ( 1.1 * lastbuyprice ) ) then
          sell( i ) = 1
          lastbuyprice = 0
       end if
    next

   AFL("buy") = buy
    AFL("sell") = sell

%>

buy = buy;
```

Further information

More scripting samples are available at the AFL on-line library at:
http://www.amibroker.com/library/list.php

In case of any further questions, comments and suggestions please use customer support page at http://www.amibroker.com/support.html . Please note that AFL scripting is fairly advanced topic and you should play a little bit with AFL first before going too deep into scripting.

Component Object Model support in AFL

Introduction

The Component Object Model (COM) is the technology that defines and implements mechanisms that enable software components, such as applications, data objects, controls, and services, to interact as objects. The COM support in AFL introduced in version 3.75beta allows to create instances of COM objects and call the functions (methods) exposed by those objects.

The COM object can be created in virtually any language including any C/C++ flavour, Visual Basic, Delphi, etc. This enables you to write parts of your indicators, systems, explorations and commentaries in the language of your choice and run them at full compiled code speed.

The scripting engines used by AFL scripting host (JScript and VBScript) also expose themselves as COM objects. AFL COM support now allows you to call functions defined in scripting part directly from AFL without the use of variables to pass the data to and retrieve data from the script.

Calling functions defined in script

Until version 3.75 the only way to exchange information between AFL and the script was using variables - this technique is explained in detail in AFL scripting host documentation .

Let's suppose we need a function that calculates second order IIR (infinite impulse response) filter:

y[n] = f0 * x[n] + f1 * y[n - 1] + f2 * y[n - 2]

Please note that well known exponential smoothing is a first order IIR filter. Implementing higher order filters minimizes lag, therefore our second order IIR may be used as a "better" EMA.

In the "old way" we would need to write the following code:

```
EnableScript("jscript");

x = ( High + Low )/2;

f0 = 0.2;
f1 = 1.2;
f2 = -0.4;

<%
x = VBArray( AFL( "x" ) ).toArray();
f0 = AFL( "f0" );
f1 = AFL( "f1" );
f2 = AFL( "f2" );

y = new Array();

// initialize first 2 elements of result array
y[ 0 ] = x[ 0 ];
y[ 1 ] = x[ 1 ]
```

```
for( i = 2; i < x.length; i++ )
{
  y[ i ] = f0 * x[ i ] + f1 * y[ i - 1 ] + f2 * y[ i - 2 ];
}

AFL.Var("y") = y;
%>

Graph0 = Close;
Graph0Style = 64;
Graph1 = y;
```

While it is OK for one-time use, if we need such a function multiple times we had to have repeat the script part which is not very nice. Much nicer approach is to have a function that can be called from multiple places without the need to repeat the same code. Defining functions in JScript of VBScript is no problem at all:

```
EnableScript("jscript");

<%


function IIR2( x, f0, f1, f2 )
{

      x = VBArray( x ).toArray();

      y = new Array();

      // initialize first 2 elements of result array
      y[ 0 ] = x[ 0 ];
      y[ 1 ] = x[ 1 ];

      for( i = 2; i < x.length; i++ )
      {
         y[ i ] = f0 * x[ i ] + f1 * y[ i - 1 ] + f2 * y[ i - 2 ];
      }

      return y;

}

%>
```

.. but how to call such a function from AFL?

The most important thing is that script engine exposes itself as a COM object. A new AFL function GetScriptObject() can be used to obtain the access to the script engine. The rest is simple - once we define the function in the script it is exposed as a method of script object retrieved by GetScriptObject:

```
script = GetScriptObject();
Graph0 = script.IIR2( ( High + Low )/2, 0.2, 1.2, -0.4 );
Graph1 = script.IIR2( ( Open + Close )/2, 0.2, 1.0, -0.2 ); // call it again and
```

```
again...
```

Note also, that with this approach we may pass additional arguments so our IIR2 filter may be re-used with various smoothing parameters.

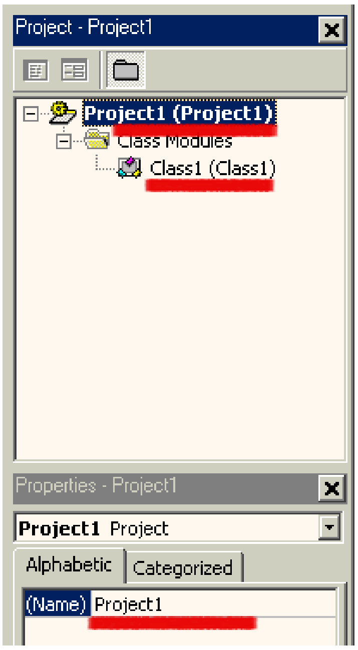
So, thanks to a new COM support in AFL, you can define functions in scripts and call those functions from multiple places in your formula with ease.

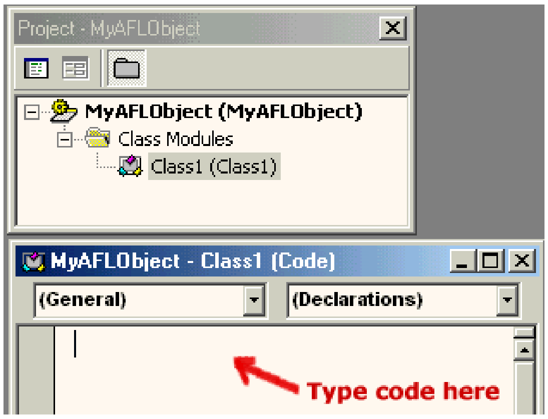
Using external COM/ActiveX objects in AFL

In a very similar way we can call functions (methods) in an external COM objects directly from the AFL formula. Here I will show how to write such external ActiveX in Visual Basic but you can use any other language for this (Delphi for example is very good choice for creating ActiveX/COM objects).

It is quite easy to create your own ActiveX DLL in Visual Basic, here are the steps required:

- Run Visual Basic
- In the "New project" dialog choose "ActiveX DLL" icon - this will create the "Project1" that looks like in the picture on the right:
- Now click on the (Name) and rename the "Project1" to something more meaningfull, for example "MyAFLObject"
- Then double click on the "Class1" in the project tree item. The code window will get the title of "MyAFLObject - Class1 (Code)" as shown below:





- Now you are ready to enter the code

As an example we will implement a similar function to one shown in the JScript. The function will calculate second order Infinite Impulse Response filter. We will call this function "IIR2"

```
Public Function IIR2(InputArray() As Variant, f0 As Variant, f1 As Variant, f2 As
Variant) As Variant

Dim Result()

ReDim Result(UBound(InputArray)) ' size the Result array to match InputArray

'initialize first two elements

Result(0) = InputArray(0)
Result(1) = InputArray(1)

For i = 2 To UBound(InputArray)

  Result(i) = f0 * InputArray(i) + f1 * Result(i - 1) + f2 * Result(i - 2)

Next

IIR2 = Result

End Function
```

The code is quite similar to the JScript version. The main difference is declaring types. As you can see all variables passed from and to AFL must be declared as Variants. This is so, because AmiBroker does not know what kind of object it speaks to and puts all arguments to the most universal Variant type and expects the function to return the value as Variant also. Currently AmiBroker can pass to your object floating point numbers, arrays of floating point numbers, strings, and pointers to other objects (dispatch pointers) - all of them packed into Variant type. When you write the ActiveX, it is your responsibility to interpret Variants received from AmiBroker correctly.

Now you should choose Run->Start in Visual Basic to compile and run the component. The code in Visual Basic will wait until external process accesses the code.

To access the freshly created ActiveX we will use the following AFL formula (enter it in the Formula Editor and press Apply):

```
myobj = CreateObject("MyAFLObject.Class1");

Graph0 = Close;
Graph0Style = 64;
Graph1 = myobj.IIR2( Close, 0.2, 1.2, -0.4 );
```

The AFL formula simply creates the instance of our ActiveX object and calls its member function (IIR2). Note that we are using new dot (.) operator to access myobj members.
Now click the "Apply" button in the Formula Editor to see how all this setup works. You should see candlestick chart with a quite nice moving average.

2.4 Conclusion

Introduction of COM support in AFL brings even more power to AFL and AmiBroker. Now you can write indicators, trading systems, explorations and commentaries using custom functions that are easy to create using scripting language or full-featured development environment of Visual Basic, Borland Delphi, C++ Builder, Visual C++ and many, many others. Using integrated development environments like those mentioned makes debugging, testing and developing much easier and faster. Also resulting compiled code executes several times faster than interpreted script or AFL.

But this is not the end of the story... C/C++ programmers can choose to write plugin DLLs that do not use COM technology at all. Plugin DLLs has some additional features including ability to call back AFL built-in functions, directly retreive and set AFL variables and support automatic syntax colouring of functions exposed by the plugin. This topic is covered in the AmiBroker Development Kit available from the member's area of AmiBroker site.

Plug-in in AFL

This section describes regular plug-in DLLs. If you are interested in ActiveX plugins check "COM support in AFL" section.

Plugin interface

AmiBroker Plugin interface allows to call an external module (DLL library) directly from AFL. Such a library can expose multiple functions that can be used in your trading systems, indicators, commentaries, scans and explorations. The plugin DLL can be created using any C/C++ compiler, Delphi and other languages supporting creation of regular DLLs. As the code of a plug in is compiled to a native processor machine code it runs several times faster than AFL. Also since you can use full power of C/C++ (or other) language to build the most complex functions with ease.

In addition to exposing functions to AFL, plugin DLLs have ability to call back AFL built-in functions, directly retreive and set AFL variables and support automatic syntax colouring of functions exposed by the plugin.

A detailed description of plugin interface and the sample code of plug in DLL are included in the AmiBroker Development Kit (ADK). **The ADK is available for registered users only (downloadable from members area).**

Getting 3rd party plugins

Freeware 3rd party plugins are available for download from http://www.amibroker.net/3rdparty.php

Using third-party plugins:

To use third-party plugin DLL just copy the DLL file to the Plugins folder in the AmiBroker directory. Then run AmiBroker. Then choose Tools->Plugins menu. In the Plugins window you should see the list of all loaded plugin DLLs. If AmiBroker was running when you copied the DLL you should click on "Unload" and then on "Load" button. This will force rescanning the Plugins folder and loading the DLLs.

When the plugin DLL is loaded the new functions exposed by this DLL become available to all your AFL formulas. For the list of functions exposed by plugin you should consult the documentation of the plugin itself.

IMPORTANT NOTE: AmiBroker makes no representations on features and performance of non-certified third-party plug-ins. Specifically certain plug-ins can cause instabilities or even crashes. Entire use of non-certified third-party plugins is at your own risk.

Common Coding mistakes in AFL

This document presents most common mistakes and problems that users encounter when writing their custom formulas. Please read carefully to avoid making similar errors.

= (assignment) vs == (equality check)

There are two similar looking but completely different operators in AFL.

= is a variable assignment operator

== is an equality check operator

EXAMPLE

Incorrect code:

```
result = IIf( Variable = 10 , High, Low ); // WRONG
```

If you want to check if variable is equal to 10, you MUST use “==”

Correct code:

```
result = IIf( Variable == 10 , High, Low ); // CORRECT
```

Using parentheses

Parentheses can be used to control the operation precedence (the order in which the operators are calculated). AmiBroker always does operations within the innermost parentheses first. To learn the the precedence of operations when parentheses are not used, visit:
http://www.amibroker.com/guide/a_language.html

EXAMPLE:

 I would like to buy whenever either Close is higher that it s 10-periods Moving Average or Close is at least 10% higher than yesterday s close, but buy should only apply when Current Volume is higher than it s 10-period Moving Average. However I get Buy signals for the days when Volume is lower than MA(Volume,10). Why?

```
Buy = Close > MA( Close, 10 ) OR Close == 1.1 * Ref( Close, -1 ) AND Volume > MA(
Volume, 10 );
```

The solution is to add parentheses, otherwise system buys whenever **Close** > MA(**Close**,10) condition is met (or **Close** == 1.1*Ref(**Close**,-1) **AND Volume** > MA(**Volume**,10) are both met).

```
Buy = ( Close > MA( Close, 10 ) OR Close == 1.1 * Ref( Close, -1 ) )
      AND Volume > MA( Volume, 10 );
```

IIf function

The IIf( ) function is used to create **conditional assignments.**

variable = IIf(EXPRESSION, TRUE_PART, FALSE_PART);

The above "IIf" statement means: For each bar EXPRESSION is true assign TRUE_PART to the *variable*, otherwise (when EXPRESSION is false) assign FALSE_PART.

EXAMPLE

Incorrect code

```
IIf( Close > 10, result = 7, result = 9 ); // WRONG
```

Correct code:

```
result = IIf( Close > 10, 7, 9 ); // CORRECT
```

IIf is for arrays, WriteIf is for strings

IIf functions should be used to handle arrays, if you need conditional text function use WriteIf instead.

EXAMPLE

Incorrect code:

```
variable = IIf(Condition, "Text 1","Text 2" ); // WRONG
```

IIf( ) function returns array, NOT STRING, so it’s impossible to assign text to variable with use of IIF. Use WriteIf( ) function instead:

Correct code:

```
variable = WriteIf( condition, "Text 1", "Text 2" ); // CORRECT
```

Please note however that WriteIf function returns just single STRING, not arrays of strings, so only the selected value is used for evaluation.

---

if-else statement needs boolean (or single numeric expression), not array

The **if** keyword executes *statement1* if *expression* is true (nonzero); if **else** is present and *expression* is false (zero), it executes *statement2*. After executing *statement1* or *statement2*, control passes to the next statement. Ex*pression* must be boolean (True/False) type (so it CANNOT be ARRAY because there would be no way do decide whether to execute *statement1* or not, if for example array was:

[True,True,False,.....,False,True])

if(expression)
  statement1
else
  statement2

EXAMPLE

```
if( Close > Open ) // WRONG
   Color = colorGreen; //statement 1
else
   Color = colorRed; //statement 2

Plot(Close,"Colored Price",Color,styleCandle);
```

The above example is wrong, as both **Open** and **Close** are arrays and such expression as **Close** > **Open** is also an ARRAY. The solution depends on the statement. It’s either possible to implement it on bar-by-bar basis, with use of FOR loop:

```
for( i = 0; i < BarCount; i++ )
{
  if( Close[ i ] > Open[ i ] ) // CORRECT
       Color[ i ] = colorGreen;
  else
       Color[ i ] = colorRed;
}

Plot( Close, "Colored Price", Color, styleCandle );
```

It is also possible in this case to use IIf( ) function:

```
Color = IIf( Close > Open, colorGreen, colorRed ); // ALSO CORRECT - working
directly on arrays
Plot( Close, "Colored Price", Color, styleCandle );
```

Barcount vs BarIndex()
There is a fundamental difference between *BarCount* and *BarIndex*(). *BarCount* is a **numeric variable** that holds just one number (the count of elements in array). On the other hand *BarIndex*() is a function that returns ARRAY representing consecutive index of each bar.

EXAMPLE

Incorrect code:

```
for (i = 0; i < BarIndex();i++ ) // WRONG
{
  // your formula
}
```

It’s not allowed to use ARRAY inside **for** loop, and Barindex() returns ARRAY. That is why it’s necessary to change the formula.

Correct code:

```
for (i =0 ; i < BarCount ;i++ )  // CORRECT
{
  //your formula
}
```

TimeFrameExpand( ) is required to match data with original time frame

The **TimeFrameSet( )** replaces current price/volume arrays: open, high, low, close, volume, openint, avg with time-compressed bars of specified interval once you switched to a different time frame all calculations and built-in indicators operate on selected time frame. To get back to original interval call **TimeFrameRestore( )** function. The **TimeFrameExpand( )** is used to decompress array variables that were created in different time frame. Decompressing is required to properly display and use the array created in different time frame.

EXAMPLE

Incorrect code:

```
TimeFrameSet( inWeekly );
MA14_Weekly = MA( Close, 14 );
TimeFrameRestore();
Buy = Cross( Close, MA14_Weekly ); // WRONG - Close and MA15_Weekly use different
time scales
```

The above formula is wrong, as MA14_Weekly variable should be EXPANDED to match original timeframe. The right contents should be:

Correct code:

```
TimeFrameSet( inWeekly );
MA14_Weekly = MA( Close, 14 );
TimeFrameRestore();
Buy = Cross( Close, TimeFrameExpand( MA14_Weekly, inWeekly ) ); // CORRECT,
expanded weekly MA can be matched against daily close
```

EXAMPLE 2:

Incorrect code:

```
TimeFrameSet( inWeekly );
MA14_Weekly = MA( Close, 14 );
TimeFrameRestore();
Buy = Cross( Close, TimeFrameExpand( MA14_Weekly, inDaily ) ); // WRONG
```

It's always necessary to indicate in TimeFrameExpand( ) function, which timeframe was variable calculated in. So if MA14_Weekly was calculated in out of weekly data, **inWeekly** should be the correct parameter of TimeFrameExpand( ) function.

Correct code:

```
TimeFrameSet( inWeekly );
MA14_Weekly = MA( Close, 14 );
TimeFrameRestore();
Buy = Cross( Close, TimeFrameExpand( MA14_Weekly, inWeekly ) ); // CORRECT
```

Porfolio Backtester Interface Reference Guide

(Updated February 20th, 2010 to cover enhancements and additions introduced in **AmiBroker 5.30.0**)

Basics

AmiBroker version 4.67.0 exposes new object-oriented interface to porfolio backtester allowing to control 2nd phase of the backtest. This allows multitude of applications including, but not limited to:

- position sizing based on portfolio-level equity
- implementing advanced rotational systems (you have now access to ranking arrays and can decide what trades to take after knowing which symbols scores best on bar-by-bar basis)
- adding your custom metrics to backtest and optimization statistics
- implementing custom formulas for slippage control
- advanced scaling-in/-out based on portfolio equity and other run-time stats
- advanded trading systems that use portfolio-level statistics evaluated on bar-by-bar basis to decide which trades to take

This document describes all objects, methods and properties exposed by portfolio interface.

Requirements

To use new interface the user needs AmiBroker 4.67.0 or higher and needs to have AFL coding skills including understanding the terms: an object, method and property.

Various approaches for various applications

The porfolio backtester interface supports various approaches to customization of backtest process that suit different applications.

- **high-level** approach **(the easiest)**
  - using Backtest() method and it runs default backtest procedure (as in old versions) - allows simple implementation of custom metrics
- **mid-level** approach
  - using PreProcess()/ProcessTradeSignal()/PostProcess() methods - allows to modify signals, query open positions (good for advanced position sizing)
- **low-level** approach **(the most complex)**
  - using PreProcess()/EnterTrade()/ExitTrade()/ScaleTrade()/UpdateStats()/HandleStops()/PostProcess() methods - provides full control over entire backtest process for hard-code programmers only

Getting access to the interface

To access new portfolio backtester interface you need to:

- enable custom backtesting procedure by calling:

  ```
  SetOption("UseCustomBacktestProc", True );
  ```

  or calling

```
SetCustomBacktestProc( "C:\\MyPath\\MyCustomBacktest.afl" );
```

in your formula

or by enabling it in Automatic Analysis->Settings window, "Portfolio" tab and specifying external custom procedure file.

- get access to backtester object by calling GetBacktesterObject() method. Note that GetBacktester method should only be called when Status("action") returns actionPortfolio:

```
if( Status("action")== actionPortfolio )
{
    // retrieve the interface to portfolio backtester
     bo = GetBacktesterObject();

     ...here is your custom backtest formula.
}
```

When using external custom procedure file you don't need to check for actionPortfolio, because external backtest procedures are called exclusively in actionPortfolio mode.

Typing Conventions

- *bool* - italic represents the type of parameter/variable/return value (*Trade, Signal, Stats* - represent the type of object returned)
- **AddSymbols** - bold represents function / method / property name
- <u>SymbolList</u> - underline type face represents formal parameter
- *[optional]* - denotes optional parameter (that does not need to be supplied)
- *variant* - represent variable type that can be either string or a number

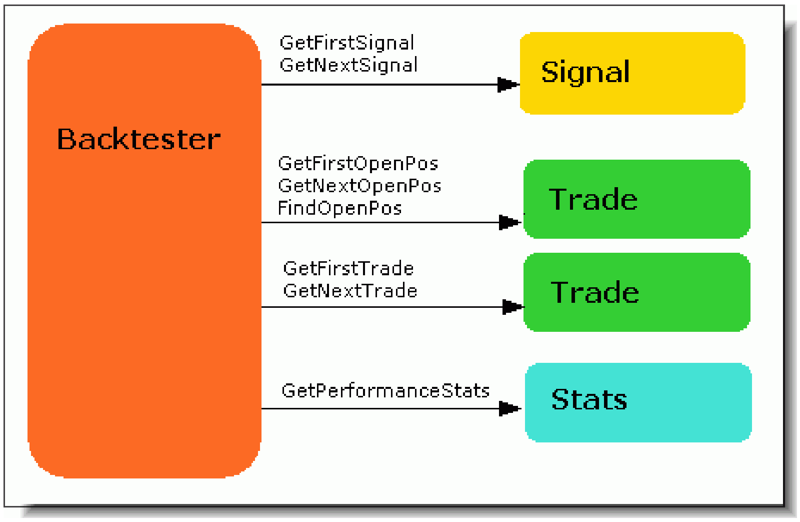
Despite the fact that interface handles integer data type such as long, short, bool and two different floating point types: float and double, the AFL itself converts all those data types to float because AFL treats all numbers as floats (32-bit IEEE floating point numbers).

Objects

The interface exposes the following objects:

- Backtester object
- Signal object
- Trade object
- Stats object

The only object directly accessible from AFL is Backtester object, all other objects are accessible by calling Backtester object methods as shown in the picture below.



Backtester object

Backtester object allows to control backtest process (process signals, enter/exit/scale trades) and get access to signal list, open position and trade list and to performance statistics object.

Methods:

- *bool* **AddCustomMetric**(*string* Title, *variant* Value, *[optional] variant* LongOnlyValue, *[optional] variant* ShortOnlyValue , *[optional] variant* DecPlaces = 2, [optional] variant CombineMethod = 2)

  This method adds custom metric to the backtest report, backtest "summary" and optimization result list. Title is a name of the metric to be displayed in the report, Value is the value of the metric, optional arguments LongOnlyValue, ShortOnlyValue allow to provide values for additional long/short-only columns in the backtest report. DecPlaces argument controls how many decimal places should be used to display the value. The last CombineMethod argument defines how custom metrics are combined for walk-forward out-of-sample summary report

  Supported CombineMethod values are:
  1 first step value, - summary report will show the value of custom metric from very first out-of-sample step
  2 last step value (default), - summary report will show the value of custom metric from the last out-of-sample step
  3 sum, - summary report will show the sum of the values of custom metric from all out of sample steps
  4 average, - summary report will show the average of the values of custom metric from all out of sample steps
  5 minimum, - summary report will show the smallest value of custom metric from all out of sample steps
  6 maximum.- summary report will show the largest value of custom metric from all out of sample steps

  Note that certain metrics calculation methods are complex and for example averaging them would not lead to mathematically correct representation of all out of sample test.
  Summaries of all built-in metrics are mathematically correct out-of-the-box (i.e. they are *not* averages, but properly calculated metrics using method that is appropriate for given value). This contrasts with custom metrics, because they are user-definable and it is up to the user to select 'combining' method, and still it may happen that none of the available methods is appropriate.
  For that reason the report includes the note that explains what user-definable method was used to

combine custom metrics.

- *bool* **Backtest**(*[optional] bool* NoTradeList)

  This high-level method performs default portfolio backtest procedure in single call. It should be used if you just need to obtain custom metrics and do not want to change the way backtest is performed. (Version 4.68.0 and above): If optional parameter NoTradeList is set to True, then trade list is not generated automatically. This is useful if you want to add some per-trade metrics. Once you add them, you can generate trade list with your metrics using **ListTrades** method.

- *long* **EnterTrade**(*long* Bar, *string* Symbol, *bool* bLong, *float* Price, *float* PosSize, *[optional] variant* PosScore, *[optional] variant* RoundLotSize, *[optional] variant* MarginDeposit, *[optional] variant* TickSize, *[optional] variant* PointValue)

  Low-level method that enters trade on any symbol. This allows to take trades even on symbols that have no corresponding signals. If values for optional parameters are not provided then AmiBroker uses values defined in Symbol->Information window

- *long* **ExitTrade**(*long* Bar, *string* Symbol, *float* Price, *[optional] variant* ExitType)

  Low-level method that exits trade on any symbol. This method searches open trade list and if there is no open trade on given symbol it does nothing. Optional ExitType parameter specifies the reason for exit (1 - regular exit, 2 - max. loss, 3 - profit, 4 - trail, 5 - N-bar, 6 - ruin)

- *Trade* **FindOpenPos**(*string* Symbol)

  This method looks for the Symbol in open position list and returns matching trade object if it finds one or returns null object otherwise.

- *Signal* **FindSignal**(long Bar, string Symbol, long Type) - new in v5.10

  where bar is a bar number, type represents type of signal to find: 0 - both entries and exits, 1 - only entries, 2 - only exits

  The method finds for first matching signal that has fPrice != -1 (different than -1). If 0 is used as type, and entry and exit is on the same bar then entry signal will be returned. Note: fPrice = -1 is a special marker meaning that given signal should be ignored.

- *long* **GetSignalQty**(long Bar,long Type) - new in v5.30

  where bar is a bar number, symbol is ticker symbol, type represents type of signal to find: 0 - both entries and exits, 1 - only entries, 2 - only exits

  The method retrieves the number of signals occuring on given bar. Note that AmiBroker to conserve memory keeps track only of 2 * MaxNumberOfPositions entry signals on any single bar, however it keeps track of ALL exit signals (because they consume much less space and at the time of signal collection it is not known which positions are already open, so all exits must be tracked to prevent missing an exit)

- *Trade* **GetFirstOpenPos**()

  This method returns first *Trade* object from open position list

- *Signal* **GetFirstSignal**(*long* Bar)

  This method returns first trading *Signal* object for given Bar

- *Trade* **GetFirstTrade()**

  This method returns first *Trade* object from closed trade list

- *Trade* **GetNextOpenPos**()

  This method returns next *Trade* object from open positions list. You should call GetFirstOpenPos before calling this method for the first time. Returns null object when no more open positions are found.

- *Signal* **GetNextSignal**(*long* Bar)

  This method returns next *Signal* object from closed signal list of given Bar. You should call GetFirstSignal before calling this method for the first time. Returns null object when no more signals are found.

- *Trade* **GetNextTrade()**

  This method returns next *Trade* object from closed trade list. You should call GetFirstTrade before calling this method for the first time. Returns null object when no more trades are found.

- *long* **GetOpenPosQty**()

  This method returns number of currently open positions
- *Stats* **GetPerformanceStats**(*long* Type)

  Calculates built-in statistics and metrics and returns *Stats* object. Type parameter specifies what trades should be counted in. Type = 0 means all trades, Type = 1 means long-only, Type = 2 means short-only.

- **HandleStops**(*long* Bar)

  This low-level method handles automatic stops (applystops). **This method MUST NOT be used in high-level and mid-level approaches**. In low-level mode you should call this method once for each bar inside trading loop.

- **ListTrades**()

  (Version 4.68.0 and above) This outputs trades to the result list of Automatic Analysis window. Usually this function does NOT need to be called because **Backtest**() method by default lists trades already. This function should only be used when you disabled trade listing in Backtest method to add some custom per-trade metrics.

- **PostProcess**()

  This mid-level and low-level method performs final processing required to complete backtest correctly. Among other things it frees price cache, closes out any open trades and outputs trade list to the Automatic Analysis window. It should NOT be used when you call Backtest() method because Backtest() already performs necessary processing.

- **PreProcess**()

  This mid-level and low-level method performs initial processing required to perform backtest correctly. Among other things it initializes price cache and sets up initial variables. It should NOT be used when you call Backtest() method because Backtest() already performs necessary processing.

- *bool* **ProcessTradeSignals**(*long* Bar)

  This mid-level method processes all trading signals for given bar. It should be called once per every bar in your custom backtesting loop.

- **RawTextOutput**(*string* Text)

  (Version 4.68.0 and above) This method outputs any string to the Automatic Analysis result list at the time of the call. The user can output text formatted in multiple columns using \t (tab) character.

- *long* **ScaleTrade**(*long* Bar, *string* Symbol, *bool* bIncrease, *float* Price, *float* PosSize, *[optional] variant* Deposit)

  Low-level method that scales trade on any symbol. This method searches open trade list and if there is no open trade on given symbol it does nothing. Optional Deposit parameter specifies margin deposit for futures, if not given then Price parameter is used.

- **UpdateStats**(*long* Bar, *long* TimeInsideBar)

  Low-level method that updates equity, exposure, trade excursions (for MAE/MFE calculations) and other internal variables required for correct calculation of statistics. **You must NOT use this function in high-level and mid-level approaches.** TimeInsideBar parameter specifies intraday time position. TimeInsideBar = 0 means opening of the bar, TimeInsideBar = 1 means middle of the bar, TimeInsideBar = 2 means end of bar. As certain internal calculations depend on end-of-bar calculations, this method must be called once and only once with TimeInsideBar parameter set to 2 at the end of processing of every bar inside in your custom backtesting loop. May be called zero or more times for every bar inside backtesting loop with TimeInsideBar parameter set to 0 (zero) or 1.
- **GetMonteCarloSim**()

  get the instance of MonteCarloSim object to access MC distributions

Properties:

- *double* **Cash**

  available funds (cash) in your portfolio

- *double* **Equity**

  current portfolio-level Equity (read-only property)
- *array* **EquityArray**

  portfolio-level Equity array (read-only property)

  Note: Since version 5.50 backtester object now has EquityArray property that returns entire equity array (not just current value). Please note that values are filled during backtest and only after backtest is complete, all values are valid. If you call it in the middle, it will contain only "upto given point" data. Avoid abusing this function and it is costly in terms of RAM/CPU.e). You may use bo.EquityArray instead of Foreign("~~~Equity", "C") in custom backtester code.
- *double* **InitialEquity**

  funds that are available at the beginning of the backtest

- *double* **MarginLoan**

  loan amount (only if you are using margin account) (read-only property)

Signal object

Signal object represents trading signal (buy/sell/short/cover) or ranking array element generated by AmiBroker during first phase of backtest when your formula is executed on every symbol under test. During this first phase scan AmiBroker collects data from buy/sell/short/cover signal, price, position size and score arrays, performs sorting of signals and put top-ranked entry signals and all scale and exit signals into the list. Separate list of trading signals is maintaned for every bar. Signal list is sorted so first entry signals appear (top ranked first) and after that scaling and exit signals follow. To conserve memory AmiBroker stores only (2*MaxOpenPositons) top-ranked entry signals per bar. It keeps however all exit and scaling signals. Once first phase is completed and backtester enters 2nd phase (real backtest) it iterates through bars and through all signals within given bar and executes trades based on this signals.

To iterate through signal list you should use GetFirstSignal() / GetNextSignal() methods of Backtester object, as shown below:

```
// retrieve the interface to portfolio backtester
bo = GetBacktesterObject();

for( i = 0; i < BarCount; i++ )
{
    for( sig = bo.GetFirstSignal( i ); sig; sig = bo.GetNextSignal( i
) )
    {
       if( sig.IsEntry() )
       {
          // handle entry signal
         ....
       }
    }

    bo.ProcessTradeSignals( i );
```

```
}
```

Methods:

- *bool* **IsEntry**()

  True if this is entry signal, False otherwise
- *bool* **IsExit**()

  True if this is exit signal, False otherwise
- *bool* **IsLong**()

  True if this is long entry (buy) or long exit (sell) or scale-in signal, False otherwise
- *bool* **IsScale**()

  True if this is scale-in or scale-out signal, False otherwise

Properties:

- *float* **MarginDeposit**

  margin deposit (for futures)
- *float* **PointValue**

  point value (for futures, currencies)
- *float* **PosScore**

  position score
- *float* **PosSize**

  requested position size (positive numbers mean dollar value, negative values mean percent of portfolio equity)
- *float* **Price**

  entry/exit/scale price
- *short int* **Reason**

  this specifies reason of exit (1 - regular exit, 2 - max. loss, 3 - profit, 4 - trail, 5 - N-bar, 6 - ruin)
- *float* **RoundLotSize**

  round lot size

- *string* **Symbol**

  symbol of security
- *float* **TickSize**

  tick size (minimum price change)
- *short int* **Type**

  this specifies signal type (0 - rank (rotational systems only), 1 - buy, 2 - sell, 3 - short, 4 - cover, 5 - scale-in, 6 - scale-out)

Trade object

Trade object represents either currently open position (open trade) or closed trade. AmiBroker maintains 2 lists of trades: open position list (accessible using GetFirstOpenPos/GetNextOpenPos methods of backtester object) and closed trade lists (accessible using GetFirstTrade/GetNextTrade methods of the backtester objects). Once open position is closed by the backtester it is automatically moved from open position list to trade list. When backtest is completed (after PostProcess call) AmiBroker closes out all open positions, so trade list includes all trades. You can access both lists any time during backtest, you can also access trade list after completion to generate trade-related stats.

To iterate through open position list you should use GetFirstOpenPos() / GetNextOpenPos() methods of Backtester object, as shown below:

```
// 'bo' variable holds Backtester object retrieved earlier

for( openpos = bo.GetFirstOpenPos(); openpos; openpos = bo.GetNextOpenPos() )
 {
     // openpos variable now holds Trade object

 }
```

To iterate through closed trade list you should use GetFirstTrade() / GetNextTrade() methods of Backtester object, as shown below:

```
for( trade = bo.GetFirstTrade(); trade; trade = bo.GetNextTrade() )
 {
     // trade variable now holds Trade object

 }
```

Methods:

- *long* **AddCustomMetric**(*string* Title, *variant* Value, *[optional] decplaces = 2)*

  (Version 4.68.0 BETA and above) This method adds PER-TRADE custom metric to the trade list only. Title is a name of the metric to be displayed in the report, Value is the value of the metric. When using this function you have to ensure that you the same metrics in the same order to every trade.

Otherwise output may be messed up. Note that in contrast to **Backtester.AddCustomMetric** method that is usually called after PostProcess, the **Trade.AddCustomMetric** should be called before **PostProcess** call because **PostProcess** lists trades. Using **Trade.AddCustomMetric** after **PostProcess** gives no result, because trades are already listed. Also if you are using **Backtester.Backtest()** method you should call it with NoTradeList parameter set to True, add your per-trade metrics and then call **ListTrades** to allow your custom metrics to be included in the output.

- *float* **GetCommission**(*[optional] bool* InclExit)

  retrieves commission paid for that trade (includes all scale in/out commissions). Depending on InclExit parameter the function returns commission including (True, default) or exluding (False) exit commission.

- *double* **GetEntryValue**()

  retrieves dollar entry value of the trade

- *float* **GetMAE**()

  retrieves trade's Maximum Adverse Excursion in percent

- *float* **GetMFE**()

  retrieves trade's Maximum Favorable Excursion in percent

- *double* **GetPercentProfit**()

  retrieves current percent profit of the trade

- *double* **GetPositionValue**( )

  retrieves current dollar value of the position.

- *float* **GetPrice**(*long* Bar, *string* Field)

  (Version 4.68.0 BETA and above) provides quick access to price arrays of open positions. Bar parameter represents the data bar to query price for, Field parameter specifies which price field you want to get, allowable values are:

  "O" (Open)
  "H" (High)
  "L" (Low)
  "C" (Close)
  "F" (Fx currency rate)

  NOTES:
  1. GetPrice method is available for OPEN POSITIONS only, when called on closed trade returns Null value
  2. Open Interest field is NOT available via GetPrice
  3. Bar must be between 0..BarCount-1, otherwise exception will occur

- *double* **GetProfit**()

  retrieves current dollar (point) profit of the trade

Properties:

- *long* **BarsInTrade**

  bars spent in trade (counting starts from 0)

  Note however that the value of zero is available only when trade is just opened in "low-level" approach, so normally you would see numbers >= 1 (all other reporting in AB remains as it was, so enter today and exit tommorrow counts as 2-bar trade)
- *float* **EntryDateTime**

  entry date/time in internal AmiBroker format (the same as used by AFL function DateTime())
- *float* **EntryFxRate**

  entry foreign exchange currency rate, if any scaling-in occurred this holds average entry fx rate
- *float* **EntryPrice**

  entry price, if any scaling-in occurred this holds average entry price
- *float* **ExitDateTime**

  exit date/time in internal AmiBroker format (the same as used by AFL function DateTime())
- *float* **ExitFxRate**

  exit foreign exchange currency rate, if any scaling-out occurred this holds average exit fx rate
- *float* **ExitPrice**

  exit price, if any scaling-out occurred this holds average exit price
- *double* **Handle**

  internal handle value that allows to uniquely identify and manage (for example exit or scale in/out) multiple trades open on the same symbol at the same time. It can be passed to ExitTrade / ScaleTrade instead of the symbol.
- *bool* **IsLong**

  True if trade is long, False otherwise
- *bool* **IsOpen**

  True if trade is open, False otherwise

- *float* **MarginDeposit**

  initial margin deposit
- *double* **MarginLoan**

  loan amount used for this trade
- *float* **PointValue**

  point value (for futures / currencies)
- *float* **RoundLotSize**

  round lot size
- *float* **Score**

  entry score
- *float* **Shares**

  number of shares / contracts
- *string* **Symbol**

  symbol of the security
- *string* **FullName**

  full name of the instrument (added in 5.69)
- *float* **TickSize**

  tick size (minimum price change)

Stats object

Stats object provides the access to built-in backtester statistics and metrics. Metrics are usually calculated once backtest is completed but it is also possible to calculate metrics during backtest. To calculate current metrics and get the access to them simply call GetPerformanceStats method of Backtester object. Please note that if you calculate statistics in the middle of the backtest they will include only closed trades.

To calculate and access stats use the following code:

```
// 'bo' variable holds Backtester object retrieved earlier

stats = bo.GetPerformanceStats( 0 );
```

Methods:

double **GetValue**(*string* <u>MetricName</u>)

retrieves the value of a metric, <u>MetricName</u> can be one of the following:

"InitialCapital" ,
"EndingCapital"
"NetProfit"
"NetProfitPercent"
"ExposurePercent"
"NetRAR"
"CAR"
"RAR"

"AllQty"
"AllPercent"
"AllAvgProfitLoss"
"AllAvgProfitLossPercent"
"AllAvgBarsHeld"

"WinnersQty"
"WinnersPercent"
"WinnersTotalProfit"
"WinnersAvgProfit"
"WinnersAvgProfitPercent"
"WinnersAvgBarsHeld"
"WinnersMaxConsecutive"
"WinnersLargestWin"
"WinnersLargestWinBars"

"LosersQty"
"LosersPercent"
"LosersTotalLoss"
"LosersAvgLoss"
"LosersAvgLossPercent"
"LosersAvgBarsHeld" ,
"LosersMaxConsecutive"
"LosersLargestLoss"
"LosersLargestLossBars"

"MaxTradeDrawdown"
"MaxTradeDrawdownPercent"
"MaxSystemDrawdown"
"MaxSystemDrawdownPercent"
"RecoveryFactor"
"CAR/MDD"
"RAR/MDD"
"ProfitFactor"
"PayoffRatio"
"StandardError"
"RRR"
"UlcerIndex"
"UlcerPerformanceIndex"

- "SharpeRatio"
  "KRatio"

Properties:

-none-

MonteCarloSim object

The object allows to access MonteCarlo simulation results and has only one method

Methods:

- GetValue(*string* "field", *float* percentile)

  retrieves "field" value at specified percentile level. Available fields: "FinalEquity", "CAR", "LowestEquity", "MaxDrawdown", "MaxPercDrawdown"

Further information

Examples and more documentation can be found in this Houston presentation covering custom backtester interface (300 KB PDF format) and the Knowledge Base: http://www.amibroker.com/kb/category/afl/custom-backtest/

How to add user-defined metrics to backtest/optimization report

One of the new additions in 4.67.x/4.68.x BETA is portfolio backtester programming interface providing full control of 2nd phase of portfolio backtest. This allows multitude of applications including, but not limited to:

- user-defined metrics (appear as an additional column in "summary" backtest result list, in optimization result and as a new row in backtest report "statistics" page, as well as per-trade metrics)
- access to portfolio-level equity when backtest loop is run - allows for example complext position sizing based on portfolio equity
- read/write access to portfolio cash - allows adding funds to portfolio
- read/write access on bar-by-bar basis to trading signals generated in 1st backtest phase allows reading ranking array and modifying signal price (for example custom slippage formulas), position size, etc
- access to list of currently open positions, each position can be queried for various properties including profit, MAE/MFE, bars in trade, etc
- access to list of closed trades, each closed trade can be queried for various properties including profit,MAE/MFE, bars in trade, etc
- three different levels of programming:
    - high-level - using Backtest() method and it runs default backtest procedure (as in old versions) - great for adding custom metrics
    - mid-level - using PreProcess()/ProcessTradeSignal()/PostProcess() methods - allows to modify signals, query open positions (good for advanced position sizing)
    - low-level - using PreProcess()/EnterTrade()/ExitTrade()/ScaleTrade()/UpdateStats()/HandleStops()/PostProcess() methods provides full control over backtest process for hard-code programmers

Technical reference of new interface is available here, in this chapter we will just focus on some practical examples.

Adding user-defined metrics

Example 1

Let's start with the easiest application: in the very first example I will show you how to add user-defined metric to portfolio report and optimization result list.

In the first step we will add Expectancy to backtest and optimization report. There is some discussion about how expectancy should be calculated but the easiest formula for it is:

Expectancy ($) = %Winners * AvgProfit - %Losers * AvgLoss

or (the other way of calculating the same)

Expectancy ($) = (TotalProfit - TotalLoss) / NumberOfTrades = NetProfit / NumberOfTrades

Let us start with this simple formulation. With this approach expectancy simply tells us expected profit per trade in dollars. The custom backtest formula that implements this user-defined metric looks as follows:

```
/* First we need to enable custom backtest procedure and
** tell AmiBroker to use current formula
*/
```

```
SetCustomBacktestProc("");

/* Now custom-backtest procedure follows */

if( Status("action") == actionPortfolio )
{
    bo = GetBacktesterObject();

    bo.Backtest(); // run default backtest procedure

    st = bo.GetPerformanceStats(0); // get stats for all trades

    // Expectancy calculation (the easy way)
    // %Win * AvgProfit - %Los * AvgLos
    // note that because AvgLos is already negative
    // in AmiBroker so we are adding values instead of subtracting them
    // we could also use simpler formula NetProfit/NumberOfTrades
    // but for the purpose of illustration we are using more complex one :-)
    expectancy =
st.GetValue("WinnersAvgProfit")*st.GetValue("WinnersPercent")/100 +
                st.GetValue("LosersAvgLoss")*st.GetValue("LosersPercent")/100;

    // Here we add custom metric to backtest report
    bo.AddCustomMetric( "Expectancy ($)", expectancy );
}

// your trading system here
fast = Optimize("fast", 12, 5, 20, 1 );
slow = Optimize("slow", 26, 10, 25, 1 );
Buy=Cross(MACD(fast,slow),Signal(fast,slow));
Sell=Cross(Signal(fast,slow),MACD(fast,slow));
```

First we need to tell AmiBroker to use custom backtest formula instead of built-in one. We are doing so by calling SetCustomBacktestProc. First parameter defines the path to the custom backtest formula (which can be stored in some external file, independent from actual trading system). If we provide empty string there, we are telling AmiBroker to use current formula (the same which is used for trading system).

In the next line we have "if" statement that enters custom backtest formula if the analysis engine is in actionPortfolio (2nd phase of portfolio backtest) stage. This is important as formula is executed in both scanning phase (when trading signals are generated) and in actual portfolio backtest phase. "if" statement allows us to enter custom backtest procedure part only when analysis engine is in actual backtesting phase.

In the next line we obtain the access to backtester programming interface by calling GetBacktesterObject function. This returns Backtester object that is used to access all functionality of new interface (more details on objects available see: http://www.amibroker.com/docs/ab401.html)

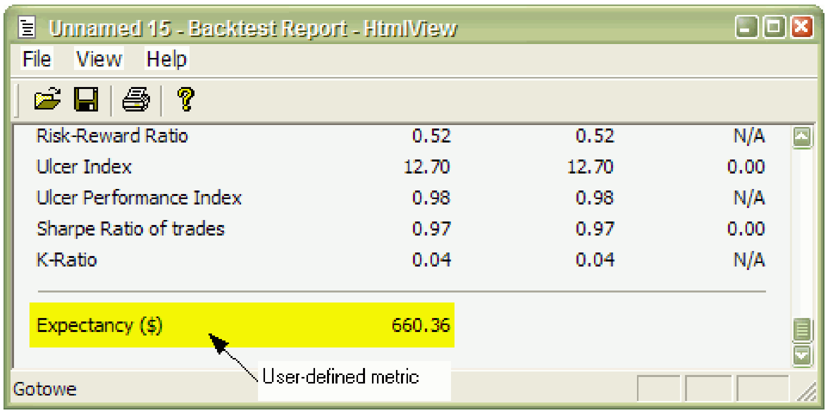
Later we obtain access to built-in metrics by calling GetPerformanceStats method of backtester object. This method returns Statistics object that allows us to access any built-in metric by calling GetValue method.

As a next step we calculate expectancy value from built-in metrics retrieved using GetValue method. For the list of metrics supported by GetValue method please check: http://www.amibroker.com/docs/ab401.html

In the final step we simply add our custom metric to the report by calling AddCustomMetric function of Backtester object. The first parameter is the name of the metric, the second is the value.

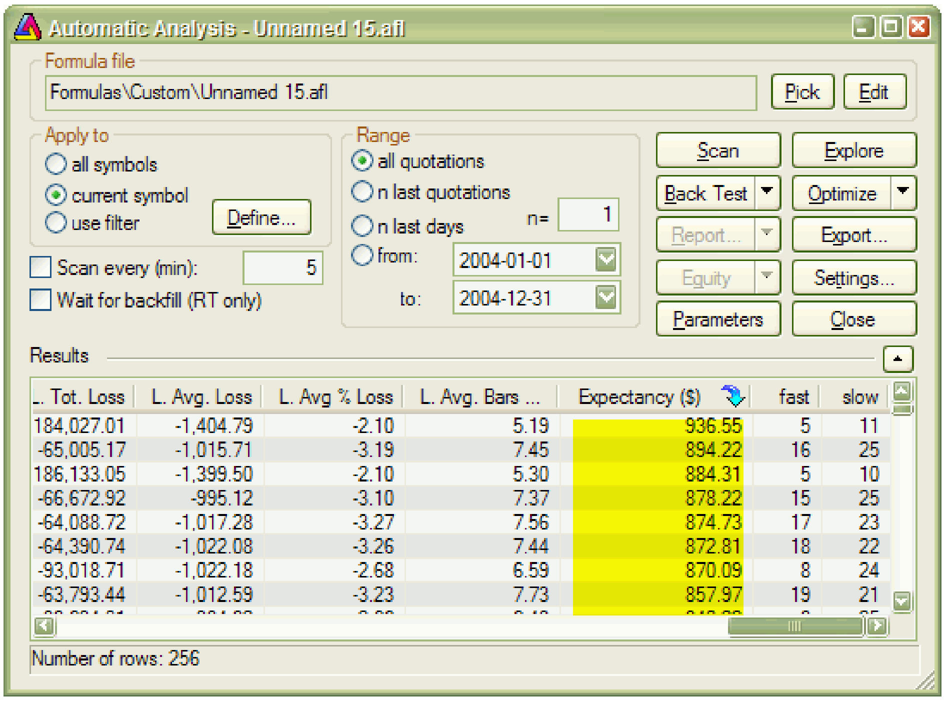
After "if"-statement implementing our custom backtest procedure usual trading system rules follow.

Now when you run Backtest and click Report button in Automatic Analysis window you will see your custom metric added at the bottom of statistics page:



Risk-Reward Ratio	0.52	0.52	N/A
Ulcer Index	12.70	12.70	0.00
Ulcer Performance Index	0.98	0.98	N/A
Sharpe Ratio of trades	0.97	0.97	0.00
K-Ratio	0.04	0.04	N/A
Expectancy ($)	660.36		

User-defined metric also appears in the Optimization result list:



Formula file: Formulas\Custom\Unnamed 15.afl

Apply to: all symbols [✓ current symbol] use filter

Scan every (min): 5
Wait for backfill (RT only)

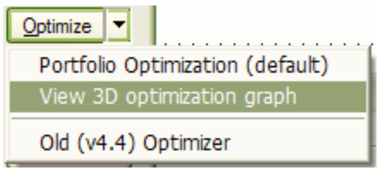
Range: [✓ all quotations] n last quotations n= 1 n last days from: 2004-01-01 to: 2004-12-31

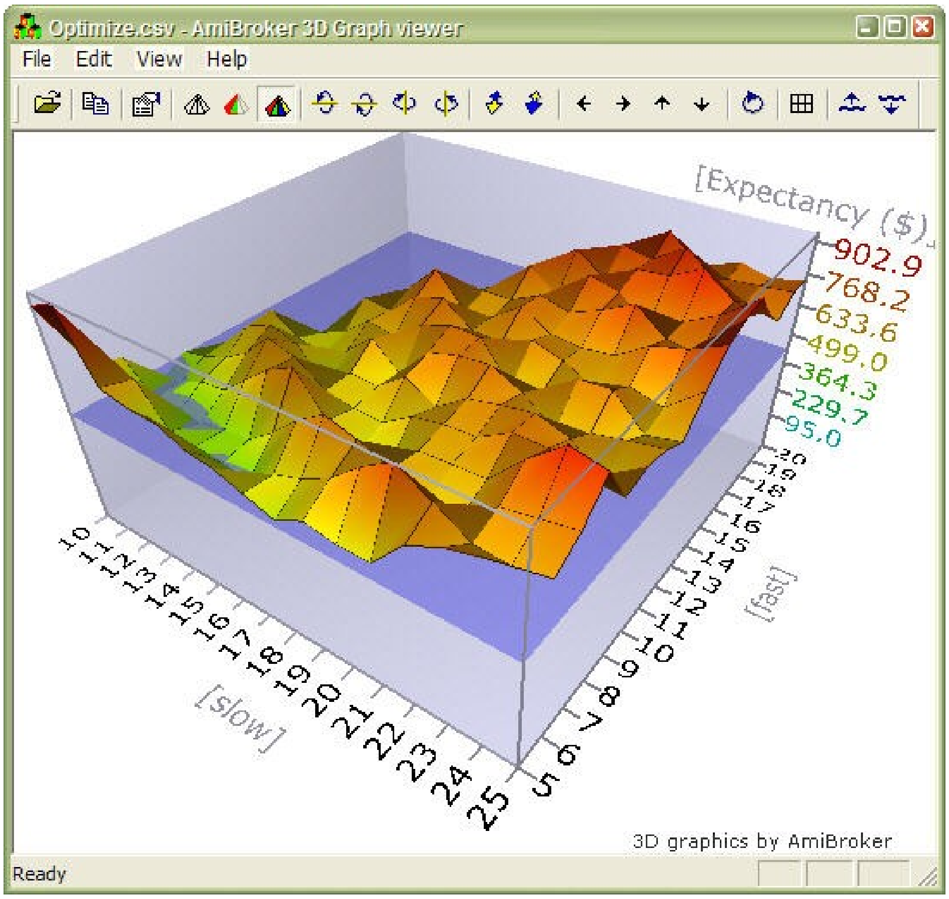
Results

.. Tot. Loss	L. Avg. Loss	L. Avg % Loss	L. Avg. Bars ...	Expectancy ($)	fast	slow
184,027.01	-1,404.79	-2.10	5.19	936.55	5	11
-65,005.17	-1,015.71	-3.19	7.45	894.22	16	25
186,133.05	-1,399.50	-2.10	5.30	884.31	5	10
-66,672.92	-995.12	-3.10	7.37	878.22	15	25
-64,088.72	-1,017.28	-3.27	7.56	874.73	17	23
-64,390.74	-1,022.08	-3.26	7.44	872.81	18	22
-93,018.71	-1,022.18	-2.68	6.59	870.09	8	24
-63,793.44	-1,012.59	-3.23	7.73	857.97	19	21

Number of rows: 256

When you click on the custom metric column, the optimization results will be sorted by your own metric and you will be able to display 3D chart of your user-defined metric plotted against optimization variables.





Example 2

Some people point out that this simple method of calculating expectancy works well only with constant position size. Otherwise, with variable position sizing and/or compounding, larger trades weight more than smaller trades and this leads to misleading expectancy values. To address this problem one could calculate expectancy for example as expected profit per $100 invested. To do calculate such statistic, one needs to iterate through trades, summing up profits per $100 unit, and dividing this sum by the number of trades. Appropriate formula follows:

```
/* First we need to enable custom backtest procedure and
** tell AmiBroker to use current formula
*/

SetCustomBacktestProc("");

/* Now custom-backtest procedure follows */

if( Status("action") == actionPortfolio )
{
    bo = GetBacktesterObject();

    bo.Backtest(); // run default backtest procedure
```

```
   SumProfitPer100Inv = 0;
   NumTrades = 0;

   // iterate through closed trades first
   for( trade = bo.GetFirstTrade(); trade; trade = bo.GetNextTrade() )
   {
      // here we sum up profit per $100 invested
        SumProfitPer100Inv = SumProfitPer100Inv + trade.GetPercentProfit();
       NumTrades++;
   }

   // iterate through eventually still open positions
   for( trade = bo.GetFirstOpenPos(); trade; trade = bo.GetNextOpenPos() )
   {
       SumProfitPer100Inv = SumProfitPer100Inv + trade.GetPercentProfit();
       NumTrades++;
   }

   expectancy2 = SumProfitPer100Inv / NumTrades;

    bo.AddCustomMetric( "Expectancy (per $100 inv.)", expectancy2 );

}

// your trading system here
fast = Optimize("fast", 12, 5, 20, 1 );
slow = Optimize("slow", 26, 10, 25, 1 );
Buy=Cross(MACD(fast,slow),Signal(fast,slow));
Sell=Cross(Signal(fast,slow),MACD(fast,slow));
```

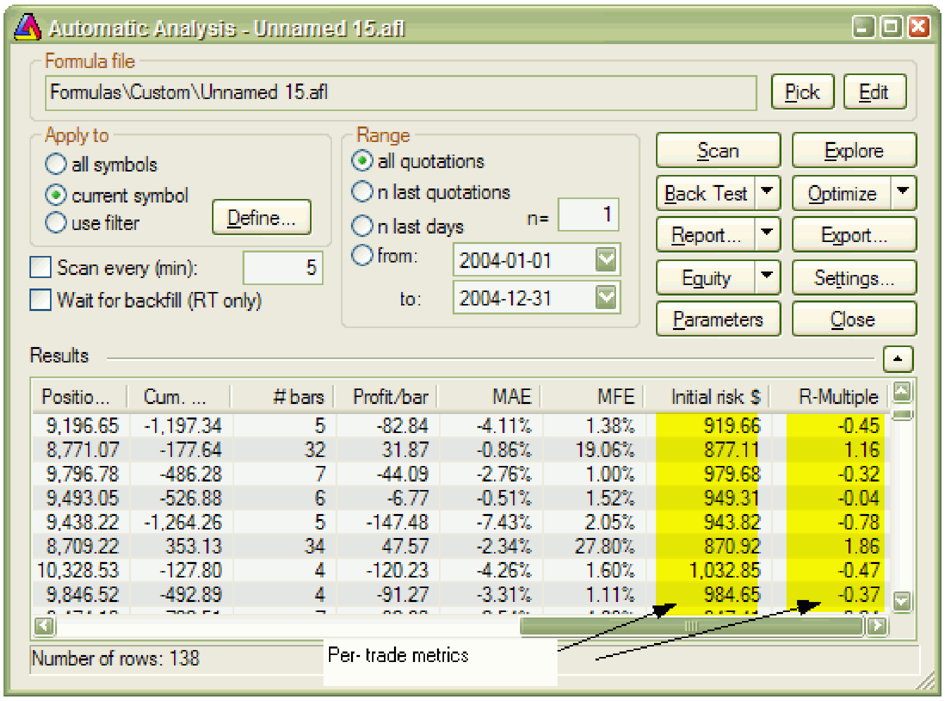
The only difference between this and previous formula is that we do not use built-in metrics to calculate our own expectancy figure. Instead we sum up all percentage profits of each trade (which are equivalent to dollar profits from $100 unit investment) and at the end divide the sum by the number of trades. Summing up is done inside the "for" loop. GetFirstTrade/GetNextTrade function pair of the backtester object allows us to step through the list of closed trades. We use two loops (second loop uses GetFirstOpenPos/GetNexOpenPos) because there may be some open positions left at the end of the backtest. If we wanted to include only closed trades then we could remove second "for" loop.

After running this code we find out that expectancy calculated this way even adjusted to initial equity (by multiplying by factor InitialEquity/$100) is smaller than expectancy calculated in the first example. This shows that "easy" method of expectancy calculation (from example 1) may lead to overly optimistic results.

Example 3

Some Van Tharp followers prefer yet slightly differnt "twist" of expectancy measure. They express expectancy in terms of expected profit per "unit of risk". The profit is then expressed in terms of R-multiples, where 1R is defined as the amount risked per trade. The amount risked is the maximum amount of money you can lose, and most often it is set by the amount of maximum loss stop (or trailing stop). According to Tharp, the easiest way to calculate expectancy is simply to add up all your R-multiples and net them out by subtracting the negative R-multiples from the positive ones, then divide by the no. of trades. This gives you your expectancy per trade.

This is very similar to approach presented in example 2, but for the calculations we do not use the value of the trade but rather risk per trade. The risk depends on the stop we use in our trading system. For simplicity in this example we have used 10% max. loss stop. In this example we also add per-trade metrics for better illustration of how R-multiples are calculated. Per-trade metrics appear in each row of the trade list in the backtest results.



Formulas\Custom\Unnamed 15.afl

Apply to: all symbols [✓ current symbol] use filter

Range: [✓ all quotations] n last quotations n last days n= 1 from: 2004-01-01 to: 2004-12-31

Scan every (min): 5

Positio...	Cum. ...	# bars	Profit/bar	MAE	MFE	Initial risk $	R-Multiple
9,196.65	-1,197.34	5	-82.84	-4.11%	1.38%	919.66	-0.45
8,771.07	-177.64	32	31.87	-0.86%	19.06%	877.11	1.16
9,796.78	-486.28	7	-44.09	-2.76%	1.00%	979.68	-0.32
9,493.05	-526.88	6	-6.77	-0.51%	1.52%	949.31	-0.04
9,438.22	-1,264.26	5	-147.48	-7.43%	2.05%	943.82	-0.78
8,709.22	353.13	34	47.57	-2.34%	27.80%	870.92	1.86
10,328.53	-127.80	4	-120.23	-4.26%	1.60%	1,032.85	-0.47
9,846.52	-492.89	4	-91.27	-3.31%	1.11%	984.65	-0.37

Number of rows: 138

The formula that implements this kind of expectancy measure follows:

```
/* First we need to enable custom backtest procedure and
** tell AmiBroker to use current formula
*/

SetCustomBacktestProc("");

MaxLossPercentStop = 10; // 10% max. loss stop

/* Now custom-backtest procedure follows */
if( Status("action") == actionPortfolio )
{
    bo = GetBacktesterObject();

    bo.Backtest(1); // run default backtest procedure

   SumProfitPerRisk = 0;
   NumTrades = 0;

   // iterate through closed trades first
   for( trade = bo.GetFirstTrade(); trade; trade = bo.GetNextTrade() )
   {
      // risk is calculated as the maximum value we can loose per trade
      // in this example we are using  max. loss stop
      // it means we can not lose more than (MaxLoss%) of invested amount
      // hence ris

       Risk = ( MaxLossPercentStop / 100 ) * trade.GetEntryValue();
       RMultiple = trade.GetProfit()/Risk;

       trade.AddCustomMetric("Initial risk $", Risk  );
       trade.AddCustomMetric("R-Multiple", RMultiple  );

       SumProfitPerRisk = SumProfitPerRisk + RMultiple;
       NumTrades++;
   }

    expectancy3 = SumProfitPerRisk / NumTrades;

    bo.AddCustomMetric( "Expectancy (per risk)", expectancy3 );

    bo.ListTrades();

}

// your trading system here

ApplyStop( stopTypeLoss, stopModePercent, MaxLossPercentStop );

fast = Optimize("fast", 12, 5, 20, 1 );
slow = Optimize("slow", 26, 10, 25, 1 );
Buy=Cross(MACD(fast,slow),Signal(fast,slow));
Sell=Cross(Signal(fast,slow),MACD(fast,slow));
```

The code is basically very similar to example 2. There are only few differences. First is that we call Backtest method with NoTradeList parameter set to 1. This way we disable default trade listing, so we can add custom per-trade metrics and list trades later by calling ListTrades method. Later we iterate through trades and calculate risk based on trade entry value and amount of max. loss stop used. The RMultiple is then calculated as trade profit divided by the amount risked per trade. Both risk and r-multiple are then added as custom per-trade metrics (note that we are callind AddCustomMetric method of **Trade** object here). Later on we do remaining calculations. At the end of the custom backtest procedure we are adding custom backtest metric (this time calling AddCustomMetric method of **Backtester** object), and after that we trigger listing of the trades using ListTrades method. For simplicity we ignore any open positions that may have left at the end of analysis period. The only change to the trading system itself was addition of maximum loss stop (ApplyStop line).

Conclusion

A new portfolio backtester programming interface provides ability to add user-defined statistics of any kind, allowing the user to move the analysis of backtesting results to completely new level.

Using low-level graphics functions

Completely new low-level graphic AFL interface allows complete flexibility in creating any kind of user-defined display.

The interface mimics closely Windows GDI API, with same names for most functions for easier use for GDI-experienced programmers. The only differences are:
1. compared to Windows GDI all functions are prefixed with 'Gfx'
2. pen/brush/font creation/selection is simplified to make it easier to use and you don't need to care about deletion of GDI objects
3. three overlay modes are available so you can mix low-level graphics with regular Plot() statements (mode = 0 (default) - overlay low-level graphic on top of charts, mode = 1 - overlay charts on top of low-level graphic, mode =2 - draw only low-level graphic (no regular charts/grid/titles/etc))

All functions use PIXELS as co-ordinates (when used on screen). For printouts and metafiles pixels are mapped to logical units to match higher resolution of printers. Use Status("pxwidth") and Status("pxheight") to find pixel dimensions of drawing surface.

Available low-level gfx functions (click on the links for detailed explanation):

GfxMoveTo(x, y)
GfxLineTo(x, y)
GfxSetPixel(x, y, color)
GfxTextOut("text", x, y)
GfxSelectPen(color, width = 1, penstyle = penSolid)
GfxSelectSolidBrush(color)
GfxSelectFont("facename", pointsize, weight = fontNormal, italic = False, underline = False, orientation = 0)
GfxRectangle(x1, y1, x2, y2)
GfxRoundRect(x1, y1, x2, y2, x3, y3)
GfxPie(x1, y1, x2, y2, x3, y3, x4, y4)
GfxEllipse(x1, y1, x2, y2)
GfxCircle(x, y, radius)
GfxChord(x1, y1, x2, y2, x3, y3, x4, y4)
GfxArc(x1, y1, x2, y2, x3, y3, x4, y4)
GfxPolygon(x1, y1, x2, y2, ...)
GfxPolyline(x1, y1, x2, y2, ...)
GfxSetTextColor(color)
GfxSetTextAlign(align)
GfxSetBkColor(color)
GfxSetBkMode(bkmode)
GfxGradientRect(x1, y1, x2, y2, fromcolor, tocolor)
GfxDrawText("text", left, top, right, bottom, format = 0)
GfxSetOverlayMode(mode = 0)

NEW FUNCTIONS IN 5.80

GfxSetCoordsMode(mode) - allows to choose between pixel and bar/price mode.
GfxSetZOrder(layer) - allows to place Gfx graphics on user-specified Z-axis (depth) layer
GfxGetTextWidth("text") - returns pixel width of specified string. NOTE: it is slow because it has to create temporary DC and font to measure the text. It takes 40us (microseconds), that is about 40 times more than

other Gfx functions.

CO-ORDINATE MODES

Starting with version 5.80 AmiBroker supports using two different co-ordinate modes in low-level graphics: pixel mode and bar/price mode.

GfxSetCoords mode function allows to switch co-ordinate system for low-level gfx functions from sceen pixel (mode = 0) - the default, to
bar / price mode (mode = 1) where X is expressed in bar index and Y is expressed in price.
This new mode allows way easier overlays on top of existing charts without need to do conversion between bars/price pixels
and without any extra refresh normally required in old versions when Y scale changed.

The function can be called to switch back and forth from pixel -> bar/price mode and vice versa a number of times
allowing to mix different modes in the same chart.

When co-ordinate mode 1 is selected (bar/price), co-ordinates can be fractional. For example if x is 2.5 it means half way between bar 2 and 3.

Example:
```
// The sample shows how using GfxSetCoordsMode( 1 )
// results in
// a) easier coding of overlay charts that plot on top of built-in charts (no
need to convert from bar/price to pixels)
// b) perfect matching between built-in Plot() and Gfx positioning
Plot( C, "Price", colorDefault, styleLine );
GfxSetOverlayMode( 1 );
GfxSetCoordsMode( 1 ); // bar/price mode (instead of pixel)

GfxSelectSolidBrush( colorRed );
GfxSelectPen( colorRed );

boxheight = 0.01 * ( HighestVisibleValue( C ) - LowestVisibleValue( C ) );

bi = BarIndex();

start = FirstVisibleValue( bi );
end = LastVisibleValue( bi );

for ( i = start; i <= end; i++ )
{
    Cl = Close[ i ];
    Op = Open[ i ];

    Color = IIf( Cl > Op, colorGreen, colorRed );
    GfxSelectPen( Color );
    GfxSelectSolidBrush( Color );

    bodyup = Max( Op, Cl );
    bodydn = Min( Op, Cl );
    GfxEllipse( i - 0.4, bodyup, i + 0.4, bodydn );

    GfxMoveTo( i, H[ i ] );
    GfxLineTo( i, bodyup );
    GfxMoveTo( i, bodydn );
    GfxLineTo( i, L[ i ] );
}
```

Z-ORDER SUPPORT

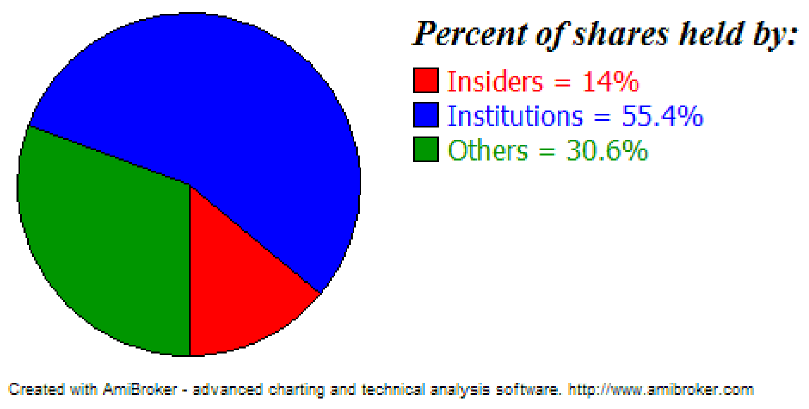
Starting from version 5.80 AmiBroker supports placing low-level graphics in different Z-order layers. A new GfxSetZOrder function is provided to allow this as shown in the example:

```
Plot( C, "Price", colorDefault );
GraphGridZOrder = 1;

GfxSetZOrder(0);
GfxSelectSolidBrush( colorGreen );
GfxCircle( 100, 100, 100 );

GfxSetZOrder(-1);
GfxSelectSolidBrush( colorRed );
GfxCircle( 150, 150, 100 );

GfxSetZOrder(-2);
GfxSelectSolidBrush( colorBlue );
GfxCircle( 180, 180, 100 );
```

Usage examples:

Example 1. Pie-chart showing percentage holding of various kinds of shareholders

Here is how it looks:



Here is the formula:

```
// OverlayMode = 2 means that nothing except
// low-level gfx should be drawn
// there will be no grid, no title line, no plots
// and nothing except what we code using Gfx* calls
GfxSetOverlayMode(2);

HInsiders = GetFnData("InsiderHoldPercent");
HInst = GetFnData("InstitutionHoldPercent");

function DrawPiePercent( x, y, radius, startpct, endpct )
{
PI = 3.1415926;
sa = 2 * PI * startpct / 100;
ea = 2 * PI * endpct / 100;
xsa = x + radius * sin( sa );
ysa = y + radius * cos( sa );
xea = x + radius * sin( ea );
yea = y + radius * cos( ea );

GfxPie( x - radius, y - radius, x + radius, y + radius, xsa, ysa, xea, yea );
}

radius = 0.45 * Status("pxheight"); // get pixel height of the chart and use 45%
for pie chart radius
textoffset = 2.4 * radius;
GfxSelectSolidBrush( colorRed );
DrawPiePercent( 1.1*radius, 1.1*radius, radius, 0, HInsiders );
GfxRectangle( textoffset , 42, textoffset +15, 57 );
GfxSelectSolidBrush( colorBlue );
DrawPiePercent( 1.1*radius, 1.1*radius, radius, HInsiders, HInst + HInsiders );
GfxRectangle( textoffset , 62, textoffset +15, 77 );
GfxSelectSolidBrush( colorGreen );
DrawPiePercent( 1.1*radius, 1.1*radius, radius, HInst + HInsiders, 100 );
GfxRectangle( textoffset , 82, textoffset +15, 97 );


GfxSelectFont("Times New Roman", 16, 700, True );
GfxTextOut("Percent of shares held by:", textoffset , 10 );
GfxSelectFont("Tahoma", 12 );
GfxSetTextColor( colorRed );
GfxTextOut( "Insiders = " + HInsiders + "%", textoffset + 20, 40 );
GfxSetTextColor( colorBlue );
GfxTextOut( "Institutions = " + HInst + "%", textoffset + 20, 60 );
GfxSetTextColor( colorGreen );
GfxTextOut( "Others = " + ( 100 - (HInst+HInsiders) ) + "%", textoffset + 20, 80
);

GfxSelectFont("Tahoma", 8 );
```

Example 2. Formatted (table-like) output sample using low-level gfx functions

```
// formatted text output sample via low-level gfx functions


CellHeight = 20;
CellWidth = 100;
GfxSelectFont( "Tahoma", CellHeight/2 );

function PrintInCell( string, row, Col )
{
GfxDrawText( string, Col * CellWidth, row * CellHeight, (Col + 1 ) * CellWidth,
(row + 1 ) * CellHeight, 0 );
}

PrintInCell( "Open", 0, 0 );
PrintInCell( "High", 0, 1 );
PrintInCell( "Low", 0, 2 );
PrintInCell( "Close", 0, 3 );
PrintInCell( "Volume", 0, 4 );

GfxSelectPen( colorBlue );
for( i = 1; i < 10 && i < BarCount; i++ )
{
PrintInCell( StrFormat("%g", O[ i ] ), i, 0 );
PrintInCell( StrFormat("%g", H[ i ] ), i, 1 );
PrintInCell( StrFormat("%g", L[ i ] ), i, 2 );
PrintInCell( StrFormat("%g", C[ i ] ), i, 3 );
PrintInCell( StrFormat("%g", V[ i ] ), i, 4 );
GfxMoveTo( 0, i * CellHeight );
GfxLineTo( 5 * CellWidth, i * CellHeight );
}
GfxMoveTo( 0, i * CellHeight );
GfxLineTo( 5 * CellWidth, i * CellHeight );

for( Col = 1; Col < 6; Col++ )
{
GfxMoveTo( Col * CellWidth, 0);
GfxLineTo( Col * CellWidth, 10 * CellHeight );
}


Title="";
```

Example 3. Low-level graphics demo featuring pie section, polygon, color-wheel, animated text and chart overlay

```
// overlay mode = 1 means that
// Low-level gfx stuff should come in background
GfxSetOverlayMode(1);
```

```
Plot(C, "Close", colorBlack, styleCandle );

PI = 3.1415926;

k = (GetPerformanceCounter()/100)%256;
for( i = 0; i < 256; i++ )
{
   x = 2 * PI * i / 256;

  GfxMoveTo( 100+k, 100 );
  GfxSelectPen( ColorHSB( ( i + k ) % 256, 255, 255 ), 4 );
  GfxLineTo( 100 +k+ 100 * sin( x  ), 100 + 100 * cos( x  ) );
}

GfxSelectFont("Tahoma", 20, 700 );
GfxSetBkMode(1);
GfxSetTextColor(colorBrown);
GfxTextOut("Testing graphic capabilites", 20, 128-k/2 );

GfxSelectPen( colorRed );
GfxSelectSolidBrush( colorBlue );
GfxChord(100,0,200,100,150,0,200,50);

//GfxPie(100,0,200,100,150,0,200,50);
GfxSelectPen( colorGreen, 2 );
GfxSelectSolidBrush( colorYellow );
GfxPolygon(250,200,200,200,250,0,200,50);

RequestTimedRefresh(1);
```

Example 4. Low-level graphic positioning - shows how to align built-in plots() with the low-level graphics. Note that if scale changes (pxheight changes) due to new data or different zoom level, it needs additional refresh to read new scale and adjust positions properly.

```
Plot(C, "Price", colorBlack, styleLine );

GfxSetOverlayMode(0);

Miny = Status("axisminy");
Maxy = Status("axismaxy");

lvb = Status("lastvisiblebar");
fvb = Status("firstvisiblebar");

pxwidth = Status("pxwidth");
pxheight = Status("pxheight");

TotalBars = Lvb - fvb;

axisarea = 56; // may need adjustment if you are using non-default font for axis

GfxSelectSolidBrush( colorRed );
```

```
GfxSelectPen( colorRed );
for( i = 0; i < TotalBars AND i < ( BarCount - fvb ); i++ )
{
   x = 5 + i * (pxwidth - axisarea - 10) / ( TotalBars + 1 );

   y = 5 + ( C[ i + fvb ] - Miny ) * ( pxheight - 10 )/ ( Maxy - Miny );

  GfxRectangle( x - 1, pxheight - y - 1, x + 2, pxheight - y + 2);
}
```

Technical information

Troubleshooting guide

Quick jump: AmiQuote, data plugins (eSignal, myTrack, IQFeed, QuoteTracker, Quotes Plus, TC2000)

CATEGORY: CRASH OR HANGUP

Where, where	Problem	Reason	Solution
AmiBroker, **Just after installation**	AmiBroker hangs at startup or frequent errors occur	Old/incompatible system components	Download and install the latest FULL version of AmiBroker from http://www.amibroker.com/download.html
AmiBroker, **At startup** **(it was working fine before)**	AmiBroker crashes or hangs	Corruption of some data file	Please try the following: • rename default database directory and try to run AmiBroker, if it works it means that some files inside this database are corrupted. You may send us DEFAULT.AWL file from "Layouts" subfolder for checking • delete or rename all DEFAULT.AWL files that you can find on your disk and try to run AmiBroker. (.AWL files are created inside "Layouts" subfolder of main AmiBroker database and "Layout" subfolders of each database directory • rename broker.charts and broker.bcharts files (in AmiBroker directory) and try to run AmiBroker • rename default.layout file (in AmiBroker directory) and try to run AmiBroker
AmiBroker, **When running some AFL code**	AmiBroker crashes or hangs	Bug in AFL formula or in AmiBroker	Send offending code to AmiBroker support (bugs@amibroker.com) and delete / modify / comment-out it to continue to work until we find the reason of the problem and solution.

CATEGORY: OTHER

Where, when	Problem	Reason	Solution
AmiBroker, **Just after installation**	Help (User's guide) is not accessible	HTMLHelp system not installed	Install the update to the HTML Help system available from Microsoft here
	Scripts do not work	Windows Scripting Host	Check if your antivirus does not block scripting

		not installed	
AmiBroker, **At startup (it was working fine before)**	Some feature does not work	Missing component of AmiBroker	Please check if you have not deleted any vital AmiBroker files. If you are unsure you may run the setup again over the old installation (please do NOT uninstall if you want to have your settings preserved)
AmiQuote - **Downloading quotes**	Download from Quote.com fails with 'the connection with the server is reset'	Livecharts password incorrect or not entered properly as described here	Please go to the AmiQuote SETTINGS page, UNMARK "Use livecharts account" and select "SERVER #1" or follow exactly the instructions here
	Download from Yahoo starts fine but then stops (especially on DSL, ADSL, cable modem connections)	Yahoo is blocking you. After infamous Internet worm attack, Yahoo considers quick, repeating multiple downloads as a "denial of service" attack.	If possible reconnect with diffferent IP (otherwise you would need to wait half an hour or more until Yahoo unblocks you) go to the AmiQuote "Settings" and set "Initial delay between requests" to 1000 and "max number of simultaneous downloads" to 1.
	Download from Yahoo fails	It may be too soon to get historical data from Yahoo or Yahoo temporary problem	First check Yahoo historical page: http://table.finance.yahoo.com/k?s=utx&g=d replace utx by the symbol in question. ========================== If the page shows old quotes - it is the problem with Yahoo not with AmiQuote. In fact AmiQuote uses " Download spreadsheet format" link on previously mentioned page: http://table.finance.yahoo.com/k?s=utx&g=d If historical data are not available you can always use "Current" mode of AMiQuote to get the data of today (even during trading hours) If you do not know the ticker symbol for index or stock or mutual fund please use symbol lookup feature: a) at Yahoo (for historical and current modes): http://finance.yahoo.com/l

			b) at Lycos (for Intraday mode): http://finance.lycos.com/home/misc/symbol_search.asp
	Download works but there are no quotes added	Local configuration problem	Follow these steps to troubleshoot the problem • Go to C:\Program Files\AmiBroker\AmiQuote\Download and check if .AQH files are there. • Open them with Notepad and see if data are there • Check if you have aqh.format file in the Formats subdirectory of AMiBroker • Check its contents and eventually send me for checking • Try to import manually as per instructions given in the user's guide • If you ever restored data directory from CD ROM check the "read-only" flag of data files in AmiBroker database. They must not be read-only. If they are you have to mark all files in the data directory and all subfolders and UNMARK read only flag.
	Download from any source fails	Connection problem	Please check your internet connection. If you are using firewall make sure to allow AmiQuote (QUOTE.EXE) to access the internet on ports 80 and 443.
AmiBroker /eSignal plugin	RT data spikes, bad ticks, other data problems.	Connection problem or other reason	Click with RIGHT mouse button over plug-in status area (green "OK" field) and choose "Fixup data for symbol" and wait a while. This will cause that entire intraday history for given symbol is re-downloaded.
AmiBroker /myTrack /IQFeed /QuoteTracker plugin	RT data spikes, bad ticks, other data problems.	Connection problem or other reason	Click with RIGHT mouse button over green "OK" field and choose "Force backfill" and wait a while. This will cause that entire intraday history for given symbol is re-downloaded.
AmiBroker /QuoteTracker plugin	Data does not update	Not enough 'ad-clicks' in QuoteTracker	You have to click on advertisements in QuoteTracker or register it. For more details see: http://www.amibroker.com/qthelp.html
AmiBroker /any RT plugin	Disconnection	Connection problem	Click with RIGHT mouse button over plug-in status area (will show red "WAIT" or "SHUT") and choose "Disconnect" and then "Connect". If this does not help restart AmiBroker
AmiBroker /QuotesPlus plugin	Error message "Can not initialize C-TREE"	Quotes Plus database problem	Check if Quotes Plus own chart program works - if not it means that this is a problem in QP2 database module and you need to contact Quotes Plus technical support for help

AmiBroker /TC2000 /TCNet plugin, sometimes	Error message "Variable or object not set"	Timing problem inside TC2000 own components.	The only partial workaround to this problem is to move all data from TC2000 CDROM to your hard disk (by Hard disk lists).

Miscellaneous questions

Question	**Answer**
Must I use Internet Explorer? Can I use Netscape Navigator, Opera, (other browser) instead?	**You can use any browser**. In certain server versions of Windows it must be selected to be installed. Internet Explorer is now just a part of operating system and AmiBroker uses extensively those parts. But you don't need to use Internet Explorer.
Where is the help file (manual)?	User's guide is available from Help->Help topics menu in AmiBroker (accessible also by F1 key)

Files and directories used by AmiBroker

AmiBroker main directory - the directory where you installed AmiBroker. You can find main AmiBroker executable file (BROKER.EXE) there.

PROGRAM FILES

- **Broker.exe** - main application file
- **CoolTool.dll**, **MiscTool.dll**, **Brokey.dll**- additional application files required by broker.exe
- **HTMLView.exe** - the report viewer, used to display backtest reports
- **emailer.exe, emailerssl.exe** - a small add-on command line utilities that sends user-defined alerts via e-mail
- **DBCAPI.DLL** - (present only if RT version is installed) - eSignal DBCAPI support library
- **Plugins** - the subfolder that contains all plug-in DLLs.
- **unins000.exe and unins000.dat** - uninstaller program.
- **Broker.chm** - compiled HELP file

DATA FILES

- **Data** - default database folder, contains **broker.master, broker.workspace**, 0-9, A-Z, '_' **symbol data subfolders** and **Layouts** subfolder. Read this to learn more about AmiBroker database concepts.
- **broker.master** - binary data file containing the table of all symbols available in the database. Used for quick loading of symbols. The table of symbols includes information about assignments of symbols to categories (markets, groups, sectors, industries, watch lists, indices and favorites). If you delete this file AmiBroker will re-create it because this information is available also in the individual symbol data files.
- **broker.workspace** - binary data file containing the information about the database settings (interval, data source used, etc), category names, global advance/decline data, etc. If you delete this file you will lost database settings and category names will be reset to defaults.
- **default.awl, *.awl** - Amibroker Workspace layout files. Text files that contain the information about the layouts and chart sheets. The default.awl file stored in "Layouts" subfolder inside database folder contains default LOCAL layout for this database. The default.awl file stored in "Layouts" subfolder of AmiBroker main directory contains default GLOBAL layout (used when there is no local layout present). If you delete this file then default layout will be generated based on default.layout file that is supplied with AmiBroker.
- **broker.newcharts** - binary file containing references between chart layouts and formula files used. If you delete it you will see "formula file empty or can not be found" in your charts because reference between chart ID and formula is lost. You will NOT however lose your formulas because formulas are separate. You will be able to reinsert them into your charts.
- **broker.layers -** text file that contains information about chart layers. If you delete this file layers will be reset to factory defaults.
- **broker.groups, broker.markets, broker.sectors, broker.industries** - text files that contain default names for groups, markets, sectors and industries. Used only at the database creation time. Later this information is stored per-database in broker.workspace file. If you delete them, AmiBroker will default to group *n*, market *n*, sector *n*, industry *n* names, where *n* is 1...256
- **broker.prefs** - binary file that contains user preference settings (available from Tools->Preferences). If you delete this file AmiBroker will reset to factory default settings
- **broker.params** - text file that contains persistent information about user-defined indicator parameters.

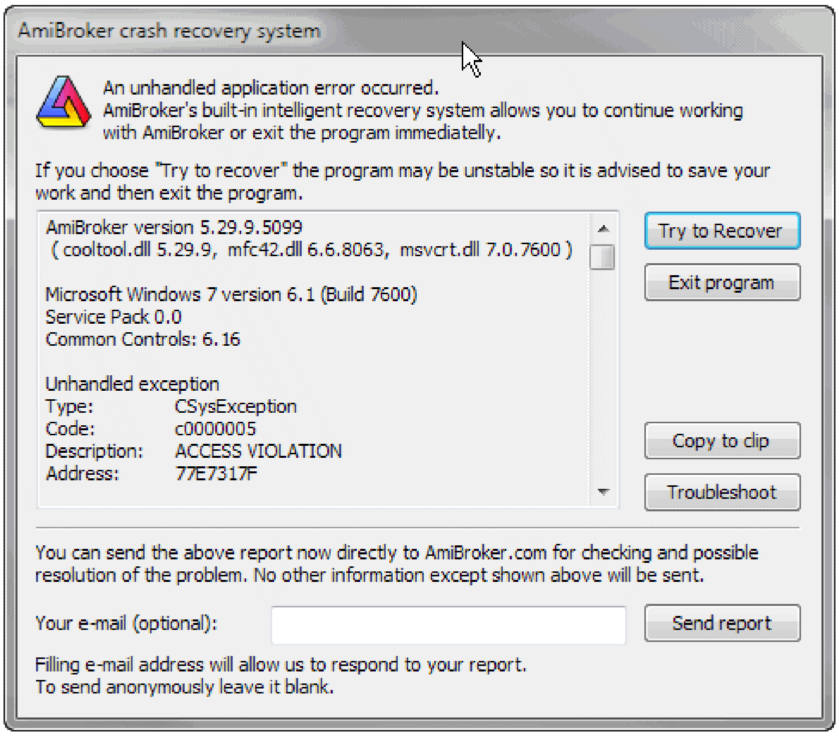
Crash recovery system and automatic bug reporting

AmiBroker features a system of detecting and reporting bugs called "Crash recovery system". The name suggests that AmiBroker is now able to recover from such unexpected situations and indeed **it can!**.

How could this be done? Well... some tricks are needed to wrap the exception handling mechanism used by Windows :-)

Normally when Windows application performs some illegal memory access, illegal operation (for example division by zero) or illegal instruction the system pops the dead-end message box saying "This program has performed an illegal operation and will be shut down". Now you have got no choice - the application is terminated when you click on OK button.

AmiBroker's crash recovery system introduced in v3.47 beta intercepts the exception generated by Windows and instead of standard dead-end message box it displays the following dialog:


AmiBroker crash recovery system

An unhandled application error occurred.
AmiBroker's built-in intelligent recovery system allows you to continue working with AmiBroker or exit the program immediatelly.

If you choose "Try to recover" the program may be unstable so it is advised to save your work and then exit the program.

```
AmiBroker version 5.29.9.5099
 ( cooltool.dll 5.29.9,  mfc42.dll 6.6.8063,  msvcrt.dll 7.0.7600 )

Microsoft Windows 7 version 6.1 (Build 7600)
Service Pack 0.0
Common Controls: 6.16

Unhandled exception
Type:         CSysException
Code:         c0000005
Description:  ACCESS VIOLATION
Address:      77E7317F
```

You can send the above report now directly to AmiBroker.com for checking and possible resolution of the problem. No other information except shown above will be sent.

Your e-mail (optional):

Filling e-mail address will allow us to respond to your report.
To send anonymously leave it blank.

As you can see there is a window that displays important system information and there are five buttons **Try to recover**, **Exit program, Copy to clip, Troubleshoot, Send report**. Clicking on the **Exit program** button works exactly the same as clicking on "OK" button in the standard Windows dead-end message box. But the first two buttons give you brand new possibilities. If you click on **Try to recover** button AmiBroker will try to recover from the error and continue running. In most cases you will be able to save your work and modifications you have made so you will not lose anything. In fact you will be able to work normally. There are however some cases when recovery will not succeed and AmiBroker may be unstable, so it is advised just to save your data and exit. It may also happen that this window will pop up for a couple of times - then you should just click on **Try to recover** several times. **Copy to clip** - copies bug report details with system information to clipboard so you can paste this information in e-mail program and send it to us.

The recover function is quite nice but the main purpose of this system is to find and fix the problems in future version and this is why the most important function was provided - **Send report**. If you the crash recovery window popped up on your screen please click on **Send report** button before attempting to continue work. This will automatically send the details shown in the crash recovery window to us. Please do add your e-mail (in **Your e-mail** field), so we can respond to your report. If you do not provide e-mail, the report will be sent anonymously, but we won't be able to respond to it and provide you any guidance.

```
AmiBroker version 5.29.9.5099
( cooltool.dll 5.29.9, mfc42.dll 6.6.8063, msvcrt.dll 7.0.7600 )

Microsoft Windows 7 version 6.1 (Build 7600)
Service Pack 0.0
Common Controls: 6.16

Unhandled exception
Type: CSysException
Code: c0000005
Description: ACCESS VIOLATION
Address: 77E7317F

Graph0=C;
Sum(C,-C)
--------^

Error 47.
Exception occurred during AFL formula execution at address: 77E7317F, code:
C0000005
Detailed exception information:
Broker.exe caused an EXCEPTION_ACCESS_VIOLATION in module ntdll.dll at
0023:77E7317F, RtlImageNtHeader()+0411 byte(s)

Call Stack:
0023:77E7317F ntdll.dll, RtlImageNtHeader()+0411 byte(s)
0023:77E73407 ntdll.dll, RtlImageNtHeader()+1059 byte(s)
0023:77E732F2 ntdll.dll, RtlImageNtHeader()+0782 byte(s)

CPU Registers:
EAX=071BD978 EBX=071BCDA8 ECX=00000000 EDX=00000000 ESI=071BD970
EDI=071B0000 EBP=00000000 ESP=0018F490 EIP=76D18FBA FLG=00010246
CS=0023 DS=002B SS=002B ES=002B FS=0053 GS=002B

AFL Parser status:
Processing stage: EXCEPTION
Formula ID: 1995 (Unnamed 190)
Action 1 (INDICATOR)

Additional information:

Number of stock loaded: 35
Currently selected stock: MCD
Number of quotes (current stock): 751

Workspace:
Data source = (default), Data local mode = 1, NumBars = 250

Preferences:
Data source = (local), Data local mode = 1, NumBars = 1000

Command history:
2783 - Preferences settings--Preferences
2824 - Shows AFL formula editor--Formula Editor
```

```
Cache manager stats:
Number of list elements: 1
Number of map elements: 1
Hash table size: 5987

Memory status:
MemoryLoad: 25 %
TotalPhys: 4194303K AvailPhys: 4194303K
TotalPageFile: 4194303K AvailPageFile: 4194303K
TotalVirtual: 4194176K AvailVirtual: 4029976K

Last Windows message:
HWnd: 0x13075e
Msg: 0x0110
wParam: 0x0025092c
lParam: 0x00000000
```

As you can see AmiBroker generates itself most important details for the bug report including even some history of menu selections (Command history) but the most essential thing at this point is to provide the description of steps needed to reproduce the bug. It would be nice if you could send us also the e-mail with the description of steps required to reproduce the problem (what you have done before the bug occurred, what special conditions must be met to reproduce it, maybe an AFL formula that you have tried and anything that you suppose might be important (even though AmiBroker includes a few lines of offending formula automatically)). This is critical since automatically generated information is very nice but can not cover all the details.

Clicking **Troubleshoot** brings up Troubleshooting page at http://www.amibroker.com/troubleshoot.html that contains descriptions of most common problems and how to solve them.

Some final notes: I have put significant amount of work in making this system reliable, however you should be aware that not all exception and/or system errors could be handled by this system and it may happen that AmiBroker will not be able to recover from some fatal error. It is also possible that this system would not be able to intercept all low level exceptions. In that case just prepare the report by yourself giving me as much details as possible.

Please remember that the final goal is making AmiBroker rock-solid and bug-free. This is what I am working on constantly.

Performance tuning tips

AmiBroker is one of the fastest (if not the fastest) technical analysis program, still the user by applying incorrect settings, poor formula coding and other sub-optimum choices can significantly slow it down. This short chapter we will give few hints on how to make program perform as it was originally designed to.

There are three areas of performance tuning:

1. Operating system / machine level
2. AmiBroker settings level
3. Formula coding

On operating system level you should do:

- Avoid installing unnecessary 3rd party programs

  Specifically avoid all kinds of "memory turbo", "windows optimizers", "windows cleaners" and such tools as they bring just more load to the CPU and system resources and provide zero or virtually zero benefit. What is worse, they could affect normal operation of Windows and cause compatibility problems
- Turn OFF virus checking for DATA files

  Many antiviruses turn on so called "live" protection on all files on all disks. This will bring system performance to its knees. Why? Simply because anti-virus will intercept each and every file access and run its extensive checking on each data access. This could result in drastic slow down of file accesses to the extent that Windows boots 3 times slower than normal.
  See the following article http://www.thepcspy.com/read/what_really_slows_windows_down/5 that shows some real measurements on impact of anti-virus.

  While I am not saying to turn off antivirus completely, it is strongly recommended to **turn OFF** anti-virus checking on **DATA FILES, specifically on all data files inside AmiBroker folder**. Generally speaking there are two kind of files: executable (.exe, .dll, .ocx) and data files (.txt, .html, .wav, .jpg, .gif, .chm, etc...) . Checking data files for viruses makes actually very little or no sense at all because they don't contain any executable code therefore can not really be the source of infection. As long as file does not include any executable part - it is safe. Note however that some data files (such as Excel .xls, can contain macros and macros _are_ executable and can be infected). But as far as AmiBroker goes, anything inside AmiBroker directory that does **not** have .exe, .dll, or .js extension is non-executable and safe.

- Choose anti-virus wisely

  As pointed out above, some anti-virus products can be really resource hogs and you should check the performance tests (see link above) before deciding on what antivirus to use.

On AmiBroker settings level

- Avoid unnecesarily large "Number of bars" in File->Database Settings

  This is the most often mistake people make. For example I have seen 50000 entered into "Number of bars" entered for end-of-day database. This does not make sense because 50000 daily bars is 192 years!!! There is no data source that offers such history and entering that amount is waste of memory.

You need to keep in mind that what ever you enter here forces AmiBroker to actually allocate memory for that many bars. Each data bar is 40 bytes. So if you enter 100000 bars, you will force AmiBroker to actually allocate 4MB of memory *per symbol*. It may not sound too much but AmiBroker keeps a cache of recently used data and if you have defined the cache that is 500 symbols large you will be able to hit 2GB of memory that way. So always look at what File->Database Settings dialog displays next to "Number of bars" field. It will display the number of days equivalent to entered number of bars and choosen base time interval. Make sure that you only enter resonable values and be aware that whatever you enter here has performance consequences. The consequences do not end with memory consumption alone, but also in speed. Processing more bars means more CPU time, especially considering the fact that while small amounts of data can be kept in on-chip CPU cache, large amounts can not. Now considering that on-chip cache is usually 10 (ten) times faster than regular memory you immediately realize that specifying too much bars here result in performance drop. Again: one bar is 40 bytes. For best performance, make sure that you don't exceed CPU on-chip cache size, so if your CPU has 4MB cache, for best performance it is strongly advised not to use more than 100000 bars in File->Database Settings.

- Decrease the size of in-memory cache, if you are using very large databases (>2GB in disk size)

  The "in-memory cache" size defined in Tools->Preferences, "Data" tab controls the number of symbols in the cache and maximum amount of memory consumed by the cache. While larger cache generally speeds up processing because data do not need to be read from disk and written back all the time, you may run out of memory on certain situations when your cache is too large, especially on 32-bit operating system that limits the available virtual memory per process to 2GB. To avoid running out of memory, go to Tools->Preferences, "Data" and decrease the size of in-memory cache. To set it to minimum, enter "11" (eleven) into "in-memory cache (max. symbols)".
  IMPORTANT: It is advised NOT to decrease the cache when your databases are relatively small (few hundred MB) as larger cache will speed up the access. Large cache is good thing as long as you do not run out of memory.

On Formula coding level

Poor formula coding is the foremost reason of slow down. People coming from "other" languages often do not realize the full potential of AFL array processing and code everything "old" style (i.e. with loops). Loops can be 10..50 times slower than equivalent array-based code. So for best speed you should avoid loops at all replacing them with array processing, or at least make looping code as short as possible.

Consider the following code:

```
SetBarsRequired( sbrAll );
GetPerformanceCounter(1);

for( i = 0; i<BarCount; i++)
{
   med[ i ] = (H[ i ] + L[ i ])/2;
}

"Loop time: "+GetPerformanceCounter(1);

med = ( H + L )/2;

"Array time: "+GetPerformanceCounter(1);
```

When running this code on 350000 bars the loop version it takes 100ms (0.1s) to execute loop version and only 2ms (0.002ms) to execute array version (so array code is 50 times faster than looping). The difference is not so significant with less bars, but still with even 300 bars, loop requires 0.1ms and array needs 0.01ms so it is 10 times faster.

So there are few guidelines for AFL coding:

- Use array processing instead of loops wherever possible. To learn more about array processing, please read the "Understanding the AFL" tutorial
- Inside loops use scalars only or refer to individual array elements using [] subscript operator
- Array operations and all loop invariant code should be moved outside of loops whenever possible to avoid repeated evaluation of the same function over and over

  Consider this code:

  ```
  for( i = 0; i < BarCount; i++ )
  {
     x = MA( C, 10 ); // WRONG ! loop invariant code is called over and over
  ! It should be OUTSIDE of loop
     y[ i ] = C[ i ]/x[ i ];
  }
  ```

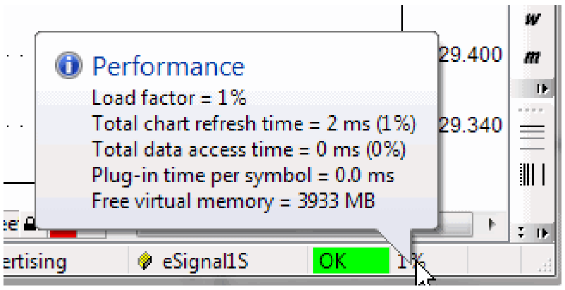
  The problem with the above code is that it repeats the same array-base moving average calculation with exactly same input over and over again (BarCount-times). The code like this is loop-invariant (does not depend on loop counter and does not have any input that is calculated depending on loop counter, so it can be calculated ONCE and as such should be moved outside (before) the loop, as shown below:

  ```
  x = MA( C, 10 ); // CORRECT ! loop invariant code located OUTSIDE of loop

  for( i = 0; i < BarCount; i++ )
  {
     y[ i ] = C[ i ]/x[ i ];
  }
  ```

- Use the AFL Formula Editor, **Tools->Code check & Profile** to find out which functions are called how many times and which ones take the most time. Start your code tuning with functions that are called the most often. Check if they are loop invariants and if so, move them OUTSIDE the loop.
- If you need to store array data in one formula and read it back in another, and were using AddToComposite/Foreign before, consider using array static variables (StaticVarGet/StaticVarSet) instead because static variables work faster. Note that static variable life is limited to program run lifetime (they are not persistent as AddToComposite tickers are), unless you set **Persistent** parameter to True in StaticVarSet.

Performance monitoring

In order to help you in real-time monitoring of program performance, AmiBroker provides two tools. First is Performance monitor window, second is Performance indicator that is located on the right most side of the AmiBroker status bar.


Performance
Load factor = 1%
Total chart refresh time = 2 ms (1%)
Total data access time = 0 ms (0%)
Plug-in time per symbol = 0.0 ms
Free virtual memory = 3933 MB

The status bar performance indicator shows:

- for real time databases: percentage load factor indicator
- for off-line intraday and eod databases: free virtual memory

Load Factor is a percentage value that shows relative 'snappiness' of the program. The load factor is calculated as (total chart refresh time in milliseconds)/2 + (total data access time in milliseconds)/2 + (free virtual memory below 20% of total memory available). So it will reach 100% if any of situations listed below happen:

- total chart refresh time is higher than 200ms
- total data access time is higher than 200ms
- free virtual memory drops below 10% of total memory (or combination of above factors)

Total chart refresh time is a sum of times needed to redraw completely all charts displayed on screen, it includes AFL execution time for each chart pane and GDI (graphics) output on screen. (It does not include data access)

Total data access time is sum of times required to access fresh data via plugin for all displayed symbols plus time required to apply time filtering, and time compression from base interval to displayed interval.

Plug-in time per symbol is time spent in plug-in GetQuotes call per *single* symbol. If you display 10 symbol charts at once AmiBroker will call GetQuotes 10 times so this time gets multiplied by number of symbols displayed (this total plugin time *is* included in total data access time figure - listed above)
If Plug-in time per symbol exceeds 10ms it means that plugin is slow (or does not use new ADK 2.0), if this is the case you should contact plugin vendor to get updated plugin that uses ADK 2.0.

Recommended is to keep this load factor below 100%. When load factor is 100% AmiBroker is able to keep updating all charts in real-time (more frequently than 5 times per second) and maintain responsive and smooth user-interface. With load factor of 200% AMiBroker is still able to keep updating all charts as frequenty as 2.5 times per second, but user interface reaction time may be impaired a bit. Keeping load at 100% or less is recommended.
200% is maximum value that allows more or less "normal" operation.

When load factor rises above 100% the warning tooltip will pop up once informing what is the reason of poor performance When load factor rises above 300% the above tooltip will reappear every minute.

AmiBroker will continue working with loads even above 1000% however the performance will be bad (one update per 5 seconds or more).

Multithreading performance

For more information about optimum use of multithreading and other general remarks regarding performance see Efficient Use of Multithreading.

How to purchase AmiBroker ?

AmiBroker is a trialware. This means that you **SHOULD** evaluate the trial version of the program for a period of 30 days before buying it.

If you like the program and want to use it for more than 30 days evaluation period - you have to buy the license to use it. We assume that you installed AmiBroker before ordering and checked if it fits your needs.

AmiBroker software is currently available in 2 editions: Standard and Professional (RT). To learn about the differences between these two versions click here.

PRICING

Licensing fees are as follows for NEW SINGLE-USER LICENSES

Product	Price
AmiBroker Standard Edition New User License	**$299**
AmiBroker Professional Edition New User License	**$399**
AmiQuote (add-on quote downloader) New User License	**$99**
AFL Code Wizard (add-on formula builder) New User License	**$99**
AmiBroker Ultimate Pack New User License (bundle of AmiBroker Professional + AmiQuote + AFL Code Wizard)	**$499**

Licensing fees are as follows for UPGRADES (only for licensed users of previous versions after 2 years of free maintenance period)

Product	Price
AmiBroker Standard Edition Upgrade License (existing user)	**$159**
AmiBroker Professional Edition Upgrade License (existing user)	**$199**

All purchases can be done securely via our web shop:

https://www.amibroker.com/order.html

BENEFITS:

Here is what YOU gain purchasing AmiBroker:

- **the keyfile** enabling all features of the program (database saving, no more annoying requesters)
- **free upgrades for TWO years from the date of purchase (with minimum of two official major feature upgrades)**
- access to **members-only zone** featuring

    - ♦ **AmiBroker Developer Kit** (for the developers of plugin DLLs)
    - ♦ newest issues **AmiBroker Tips weekly newsletter**
    - ♦ monthly **Stocks&Commodities® Traders' Tips for AmiBroker**
    - ♦ newest, private versions of AmiBroker
    - ♦ extra AFL formulas for indicators, commentaries, trading systems
- ability to influence the **future of AmiBroker** because your proposals of new features are much more likely to be implemented
- **50% discount** on next year of upgrade and maintenance pack
- 24 months **technical support** via e-mail
- other bonuses

DELIVERY

After paying registration fee you will receive the **personalized keyfile by e-mail**. No other delivery methods are supported. **When purchasing please supply your e-mail address**.

HOW TO ORDER AMIBROKER?

ORDERING ON-LINE
If you would like to buy AmiBroker, you can do the purchase online securely using links below. Payment methods include all major credit cards as well as cheques and wire transfers.

To place an order-on line, please visit: **https://www.amibroker.com/order.html**

AmiBroker online ordering is provided by FastSpring, ShareIt Digital River a well established shareware registration and credit card processing agents. Their server uses your browser's powerful built in encryption and security, along with VeriSign/Thawte authentication, to encrypt your personal information and credit card details so that they cannot be intercepted by hackers or other third parties.

All credit card data are transmitted using the secure (encrypted) HTTP protocol according to the current SSL (Secure Socket Layer) 128-bit strong cryptography standard. We have all heard a lot of talk about whether shopping on the internet is safe. The fact is that this year on-line shoppers will spend over $5.7 billion dollars according to International Data Corp. The main concern of on-line shoppers is that their credit card information will somehow end up in the wrong hands. FastSpring, ShareIt registration services use Secure Server technology, which encrypts your order information, keeping it private and protected. This technology is used by all the major commercial shopping sites. It is actually safer to transmit your credit card info over the Internet than it is to use your credit card around town.

For more information on security matters, please consult your browser's documentation. Also please note that all information submitted in the online shop is 100% confidential - we won't sell or give away your email address or other details!

ADDITIONAL INFORMATION

On-line purchasing is the fastest way to obtain your personal registration code(s). Once you complete your registration, you will receive your personal data within 24 hours.

It's of main importance that you give us a complete and correct Internet e-mail address. Entering an incorrect e-mail address (or an e-mail address that doesn't work correctly), you won't be able to register your software.

Further questions regarding registration are answered via support page http://www.amibroker.com/support.html

Credits

Thanks

Many people make significant contributions to the development and testing of the AmiBroker. Thank you all. Thanks to Herman van den Bergen, Bill Barack, Bruce Robinson, Howard B. Bandy, Rick Perkins, Ken Close, Dimitris Tsokakis, Marek Ch³opek, Ed Winters, Patrick Hargus, Mark Leon, Dan Clark, Rick Parsons, Charlie Hooper, Steve Wiser, Jim Ellis, David Holzgrefe, Carlton McEachern, Geoff Mulhall, Richard Cloonan, Peter Gialames, Stephane Carrasset, Dale Wingo, Fred Tonetti, Chuck Rademacher, Gary A. Serkhoshian, Rick Perkins, Tom Supera, Michael Robb, Mark Allen, Geo Singleman, Anthony Faragasso, Jayson Casavant, Al Holzwarth, Sidney Kaiser, William Peters, Ara Kaloustian and all the other AmiBroker users for giving me valuable feedback, comments, ideas, suggestions, test results and all the support. Special thanks to eSignal, myTrack, IQFeed for their generous support and co-operation. Thanks to Jerry Medved (QuoteTracker) for co-operation. Thanks to Mark Jurik of Jurik Research for providing his tools. Special thanks to Gary Lyben of Quotes Plus for the help with interfacing to Quotes Plus database. Many thanks to all the contributors to the AFL formula library for sharing their work. Many thanks to Sharenet (Robin and Steve) and all South African users for their continuous support. Thanks to Jordan Russell and Martijn Laan for their InnoSetup/ISX. Thanks to Neil Hodgson for Scintilla control. Thanks to Donald Dalley for extensive support he provided for Amiga version of AmiBroker. The deepest thanks and love to my wife Elizabeth and kisses for our children Julia and Jacob and Nico for bringing so much joy to my life everyday. And thank You, Dear User, for purchasing AmiBroker! With your kind support we can make dreams come true.

AmiBroker on the Web

For latest news, patches and updates please check out AmiBroker WWW site at: http://www.amibroker.com.

Visit support section of AmiBroker web page at: http://www.amibroker.com/support.html

Knowledge Base:

http://www.amibroker.com/kb/

DevLog:

http://www.amibroker.com/devlog/

Official AmiBroker Community Forum at:

https://forum.amibroker.com

AFL on-line library:

http://www.amibroker.com/library/

AFL on-line reference:

http://www.amibroker.com/guide/afl/

Third-part area (plugins, documentation):

http://www.amibroker.net/3rdparty.php