DBOpen

Initializes DLL. This needs to be called globally at every startup/application recycling.

Syntax

```
DBOpen(  ByVal AccountNumber as String,
         ByVal Password as String       ) as Integer
```

Parameters

AccountNumber required	ByVal	String	Account number for FT Lightning subscription
Password required	ByVal	String	Password for account

Return Value

Type: Integer
1 for successful initialization and -1 for failure

Exceptions

-5000000	Account number and password invalid. Please visit http://www.ftlightning.com for free trial
-5100000	Dll must be initialized by calling the DBOpen() method with valid a account number and password. Visit http://www.ftlightning.com for free trial.
General Exceptions	Error that apply to all methods

Comments

This method must be executed and return a 1 in order for any dll functions to return valid results and should be done upon every application start up.

Additionally, the dll needs to be re-initiallized every night at approximately 7PM Central Standard Time when FastTrack updaes its servers with new data. The dll will automatically re-initialize with the last entered Account Number and Password.

It is only necessary to execute this method once per applicatoin start up unless your Account Number and Password change.

Note: every execution of this method clears FTLightning's local cache.

Example

```
Dim int as Integer = DBOpen( "AccountNumber", "Password" )

Debug.Print("Result is: " + int.ToString)

#############

Result is: 0
```

LocalCacheUp2Date

Ensures all cached data is up to date and purges stale data.

Syntax

```
LocalCacheUp2Date() as Integer
```

Parameters

none

Return Value

Type: Integer
Returns a 1 if up to date 0 if update failed.

Exceptions

General Exceptions	Error that apply to all methods

Comments

For improved performance FT Lightning caches some data (mainly the results of DivAdjPrices()).

Since fund and stock data is updated with new prices daily, long running applications should call this before data intensive calculations to make sure the local cache has the most up to date data.

Example

```
Dim int as Integer = LocalCacheUp2Date()

Debug.Print("Result is: " + int.ToString)

#############

Result is: 1
```

ClearCache

Clears the local cache of all stale data.

Syntax

```
ClearCache() as Integer
```

Parameters

none

Return Value

Type: Integer
Returns 1 if cache is cleared and -1 if cache clearning failed.

Exceptions

General Exceptions	Error that apply to all methods

Comments

No Comment.

Example

```
Dim str() as String = ClearCache( )

Debug.Print("Result is: ")

For i = 0 to str.count - 1
  Debug.Print(str(i).ToString)
Next

#############

Result is: 0
```

DivAdjPrices

Returns an array that contains the dividend adjusted prices for all the ticker specified in strSym

Syntax

```
DivAdjPrices( ByVal strSym as String) as Single()
```

Parameters

strSym required	ByVal	String	Ticker symbol

Return Value

Type: Single()
Array of Singles containing dividend adjusted prices of strSym

Exceptions

-1500000	Missing/invalid ticker symbol
-1900000	Invalid strDelimiter
General Exceptions	Error that apply to all methods

Comments

1A value is returned for every market day in the FT Database. This means all prices preceding the ticker's start date are returned as the same value as the start date.

If ticker is invalid, -1500000 is returned instead of array/delimited string

Example

```
Dim sng() as Single = DivAdjPrices( "SPY" )

Debug.Print("Result is: " + sng.ToString)

For i = 0 to sng.count - 1
```

```
  Debug.Print(sng(i).ToString)
Next

#############

Result is: 27.657
27.657
27.657
27.657
27.657
...
244.01
244.42
245.56
245.53
245.66
246.99
```

MarketDay2Julian

Converts FT market date to Julian date

Syntax

```
MarketDay2Julian( ByVal vDay as vDay) as Integer
```

Parameters

vDay required	ByVal	vDay	Any form of Julian date, Market Day, or calendar date (according to your Windows Regional settings).

Return Value

Type: Integer
vDay converted to Julian date

Exceptions

General Exceptions	Error that apply to all methods

Comments

See vDay comments by hovering over the magnifying glass above.

Example

```
Dim str as String = MarketDay2Julian( "9/1/88" )

Debug.Print("Result is: " + str.ToString)

#############

Result is: 32387
```

MarketDay2Date

Converts FT market day to calendar date

Syntax

```
MarketDay2Date( ByVal vDay as vDay) as Date
```

Parameters

vDay required	ByVal	vDay	Any form of Julian date, Market Day, or calendar date (according to your Windows Regional settings).

Return Value

Type: Date
vDay converted to a calendar date

Exceptions

General Exceptions	Error that apply to all methods

Comments

See vDay comments by hovering over the magnifying glass above.

Example

```
Dim str as String = MarketDay2Date( "9/1/88" )

Debug.Print("Result is: " + str.ToString)

#############

Result is: 9/1/1988
```

DateMath

Performs mathematics on a date.

Syntax

```
DateMath(  ByVal interval as String,
           ByVal number as Double,
           ByVal vDay as vDay         ) as Date
```

Parameters

interval required	ByVal	String	"q" - quarter, "m" - month, "y" - day of year, "D" - day, "w"-weekday, "ww"-Week, "yyyy" - Year
number required	ByVal	Double	Number of intervals (can be positive or negative)
vDay required	ByVal	vDay	Any form of Julian date, Market Day, or calendar date (according to your Windows Regional settings).

Return Value

Type: Date
Returns the result as a Date in the form MM/DD/YYYY.

Exceptions

General Exceptions	Error that apply to all methods

Comments

If number is not an integer, it is rounded to the nearest whole number before being evaluated.

Note: The term "weekday" ("w") has nothing to do with skipping weekend days; i.e., it is NOT equivalent to "business days."

Example

```
Dim int as Integer = DateMath( "m", "24", "9/1/88" )
```

```
Debug.Print("Result is: " + int.ToString)

#############

Result is: 8/31/1990
```

Date2Julian

Converts any date to Julian date

Syntax

```
Date2Julian( ByVal vDay as vDay) as Integer
```

Parameters

vDay required	ByVal	vDay	Any form of Julian date, Market Day, or calendar date (according to your Windows Regional settings).

Return Value

Type: Integer
vDay converted to Julian date

Exceptions

General Exceptions	Error that apply to all methods

Comments

See vDay comments by hovering over the magnifying glass above.

Example

```
Dim int as Integer = Date2Julian( "9/1/88" )

Debug.Print("Result is: " + int.ToString)

#############

Result is: 32387
```

Date2Str

Converts any date to calendar date of format: MM/DD/YY

Syntax

```
Date2Str( ByVal vDay as vDay) as String
```

Parameters

vDay required	ByVal	vDay	Any form of Julian date, Market Day, or calendar date (according to your Windows Regional settings).

Return Value

Type: String
vDay converted to a calendar date formatted MM/DD/YYYY

Exceptions

General Exceptions	Error that apply to all methods

Comments

See vDay comments by hovering over the magnifying glass above.

Example

```
Dim str as String = date2str( "9/1/88" )

Debug.Print("Result is: " + str.ToString)

#############

Result is: 9/1/1988
```

Date2MarketDay

Converts any date format to a FastTrack Market Day.

Syntax

```
Date2MarketDay( ByVal vDay as vDay) as Integer
```

Parameters

vDay required	ByVal	vDay	Any form of Julian date, Market Day, or calendar date (according to your Windows Regional settings).

Return Value

Type: Integer
vDay converted to a Fasttrack Market Day.

Exceptions

General Exceptions	Error that apply to all methods

Comments

If vDay does not fall on a Market Day (for example a Saturday), then the Market Day prior to vDay is returned.

See vDay comments by hovering over the magnifying glass above.

Example

```
Dim int as Integer = Date2MarketDay( "9/1/88" )

Debug.Print("Result is: " + int.ToString)

#############

Result is: 1
```

StartMarketDay

Returns the ticker's starting day in the FastTrack Database.

Syntax

```
StartMarketDay( ByVal strSym as String) as Integer
```

Parameters

strSym required	ByVal	String	Ticker symbol

Return Value

Type: Integer
Market Date which the fund starts.

Exceptions

General Exceptions	Error that apply to all methods

Comments

This is the day before the first change in price in the data series. This often differse from the official inception date.

Example

```
Dim int as Integer = StartMarketDay( "SPY" )

Debug.Print("Result is: " + int.ToString)

#############

Result is: 1116
```

MaxNumDays

Returns the number of market days in the FT Database.

Syntax

```
MaxNumDays() as Integer
```

Parameters

none

Return Value

Type: Integer
Number of Market Days in FT DataBase

Exceptions

General Exceptions	Error that apply to all methods

Comments

No comment.

Example

```
Dim int as Integer = MaxNumDays()

Debug.Print("Result is: " + int.ToString)

#############

Result is: 7278
```

Families

Returns names of all Fund or Stock related families

Syntax

```
Families( ByVal FamType as String) as String
```

Parameters

FamType required	ByVal	String	FTDEF for Fund Families or SFTDEF for Stock Families

Return Value

Type: String
The result is a delimited string of family names for the specified FastTrack database.

Exceptions

-2100000	Invalid Family type. Only "FTDEF" or "SFTDEF" allowed.
General Exceptions	Error that apply to all methods

Comments

FTDEF - Fund FastTrack Defined Families.
SFTDEF - Stock FastTrack Defined Families.

The first item in string is # of families in delimted string.

Example

```
Dim temp as String = Families( "FTDEF" )
Dim str() as String = temp.Split("|")

Debug.Print("Result is: ")

For i = 0 to str.count - 1
  Debug.Print(str(i).ToString)
```

```
Next

#############

Result is:
1083
0-5
10+
10-YEAR
13D ACTIVIST
1919
...
YCG
YIELD
YIELDSHARES ETF
ZACKS
ZEO
ZWEIG
```

Family

Returns members (tickers symbols) of specified family

Syntax

```
Family( ByVal strFamily as String) as String
```

Parameters

strFamily required	ByVal	String	Family name

Return Value

Type: String
The result is a delimited string of ticker symbols for the specified FastTrack family

Exceptions

-2000000	Invalid family name
General Exceptions	Error that apply to all methods

Comments

The result is formatted as follows:
Index 0: Family Name
Index 1: Family Description
Index 2: A count of number of family members
Index 3+: The ticker symbols for each family member.

Example

```
Dim temp as String = Family( "SELECT" )
Dim str() as String = temp.Split("|")

Debug.Print("Result is: ")

For i = 0 to str.count - 1
```

```
  Debug.Print(str(i).ToString)
Next

#############

Result is:
SELECT
Fidelity Advisor Funds on 01/20/16(BUY)
40
FBIOX
FBMPX
FBSOX
...
FSRFX
FSRPX
FSTCX
FSUTX
FSVLX
FWRLX
```

GetAllAvailableTickers

Returns "|" delimited string of tickers available to current App ID

Syntax

```
GetAllAvailableTickers() as String
```

Parameters

none

Return Value

Type: String
The result is a "|" delimited string of ticker symbols available to the AppID specified in DBOpen()

Exceptions

General Exceptions	Error that apply to all methods

Comments

No comment

Example

```
Dim str() as String = GetAllAvailableTickers() )

Debug.Print("Result is: ")

For i = 0 to str.count - 1
  Debug.Print(str(i).ToString)
Next

#############

Result is: ticker1|ticker2|ticker3|etc...
```

IsIssue

Returns True/False if issue exists in the FastTrack database

Syntax

```
IsIssue( ByVal strSym as String) as Boolean
```

Parameters

strSym required	ByVal	String	Ticker symbol

Return Value

Type: Boolean
True if ticker exists in the FastTrack database, False if does not exist in FastTrack database.

Exceptions

General Exceptions	Error that apply to all methods

Comments

No comment.

Example

```
Dim bool as Boolean = IsIssue( "SPY" )

Debug.Print("Result is: " + bool.ToString)

#############

Result is: True
```

SymName

Returns name name of specified ticker symbol

Syntax

```
SymName( ByVal strSym as String) as String
```

Parameters

strSym required	ByVal	String	Ticker symbol

Return Value

Type: String
Name of ticker symbol.

Exceptions

General Exceptions	Error that apply to all methods

Comments

A ticker name of -4 mean no ticker name found.

Example

```
Dim str as String = GetSymbolName( "SPY" )

Debug.Print("Result is: " + str.ToString)

#############

Result is: StateSt ETF SPDR S&P 500
```

Correlation

Returns correlation for vPrices compared to the vBasis

Syntax

```
Correlation(  ByVal vPrices as Object,
              ByVal vBasis as Object,
              ByVal intCorLength as Integer,
              ByVal StartDate as vDay,
              ByVal EndDate as vDay          ) as Single
```

Parameters

vPrices required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
vBasis required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
intCorLength required	ByVal	Integer	The length, in days, of the period for which correlation is calculated
StartDate required	ByVal	vDay	Date to start analysis
EndDate required	ByVal	vDay	Date to end analysis

Return Value

Type: Single
Correlation of vPrices compared to the vBasis from StartDate to EndDate
If vPrices or vBasis is a ticker symbol, then dividend adjusted prices are used to calculate correlation.

Exceptions

-1500000	Missing/invalid ticker symbol
-1600000	Invalid array as vPrices. Length does not equal maxnumdays.
-1700000	Invalid smoothday
General Exceptions	Error that apply to all methods

Comments

Correlation measure the degree to which two different price series exhibit similar movement.

It is important to note that the calculation is based on the change of prices, not the actual prices themselves

FastTrack's correlation is computed according to an algorithm published in "Biometry", Robert Sokal and F. James Rohlf, W.H. Freeman, 1969, Page 509.

Example

```
Dim sng as Single = Correlation( "OAKMX", "SP-CP", "5", "9/1/88",
"1/1/11" )

Debug.Print("Result is: " + sng.ToString)

#############

Result is: 0.8307067
```

SD

Returns standard deviation of vPrices.

Syntax

```
SD(  ByVal vPrices as Object,
     ByVal StartDate as vDay,
     ByVal EndDate as vDay    ) as Single
```

Parameters

vPrices required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
StartDate required	ByVal	vDay	Date to start analysis
EndDate required	ByVal	vDay	Date to end analysis

Return Value

Type: Single
Standard Deviation of vPrices from StartDate to EndDate
If vPrices is a ticker symbol, then dividend adjusted prices are used to calculate standard deviation.

Exceptions

-1500000	Missing/invalid ticker symbol
-1600000	Invalid array as vPrices. Length does not equal maxnumdays.
General Exceptions	Error that apply to all methods

Comments

The FastTrack's SD algorithm is generally described as "The Difference between the Mean and the Square". There are other types of SD calculations which will differ from FastTrack's results. FastTrack's Standard Deviation is calculated on a daily basis and then adjusted to a monthly basis by multiplying by the square root of 21 (the average number of market days in a month.)

FastTrack's monthly SD can be converted (approximately) to Morningstar's annual SD by

multiplying the SD= value by the 3.4 (the square root of 12). This calculation is extremely sensitive to period measured. FastTrack performance values measure the period on screen. It is virtually impossible to determine the EXACT starting and ending period for other folks' calculations.

In 1992 when Morningstar choose to change from using monthly SD to annual SD, FastTrack choose to retain the original monthly formulation. FastTrackers commonly trade funds a few times per year. It makes no sense measure a period of several months with an annualized statistic.

Example

```
Dim sng as Single = SD( "OAKMX", "9/1/88", "1/1/11" )

Debug.Print("Result is: " + sng.ToString)

#############

Result is: 0.04626092
```

Yield1Y

Returns the 1 year yield (%) before EndDate.

Syntax

```
Yield1Y( ByVal strSym as String,
         ByVal EndDate as vDay   ) as Single
```

Parameters

strSym required	ByVal	String	Ticker symbol
EndDate required	ByVal	vDay	Date to end analysis

Return Value

Type: Single
Percentage yield of specified ticker symbol for 1 year prior to EndDate

Exceptions

-1500000	Missing/invalid ticker symbol
-1800000	No possible value. End date must be at least one year after ticker's start date.
General Exceptions	Error that apply to all methods

Comments

Yield1Y is calculated as follows:
Yield1Y = TotalIncome / (FDPrice + TotCapGains)
where
TotalIncome: The total of income distributions of all types, adjusted for reinvestments.
FDPrice: First Day's closing price (not dividend-adjusted)
TotCapCains: Total capital gains paid during the year.

Example

```
Dim sng as Single = Yield1Y( "SPY", "1/1/11" )

Debug.Print("Result is: " + sng.ToString)

#############

Result is: -1800000
```

MaxDrawDown

Returns the maximum drawdown for vPrices

Syntax

```
MaxDrawDown( ByVal vPrices as Object,
             ByVal StartDate as vDay,
             ByVal EndDate as vDay,
             Optional ByVal strFormat as String) as String
```

Parameters

vPrices required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
StartDate required	ByVal	vDay	Date to start analysis
EndDate required	ByVal	vDay	Date to end analysis
strFormat optional	ByVal	String	Format of MaxDrawdown Output "ND" for no dates, "D" for just dates, "B" for both

Return Value

Type: String
Maximum drawdown of vPrices from StartDate to EndDate

Exceptions

-1500000	Missing/invalid ticker symbol
-1600000	Invalid array as vPrices. Length does not equal maxnumdays.
General Exceptions	Error that apply to all methods

Comments

Maximum drawdown is the percentage loss from a ticker's peak value to its lowest value, over a specified time period. In general, Max Drawdown will be highest for issues whose Standard Deviation (SD=) is the highest.

Example

```
Dim sng as Single = MaxDrawDown( "OAKMX", "9/1/88", "1/1/11", "ND" )

Debug.Print("Result is: " + str.ToString)

#############

Result is: -0.5614578
```

UlcerIndex

Returns the Ulcer Index for vPrices.

Syntax

```
UlcerIndex(  ByVal vPrices as Object,
             ByVal StartDate as vDay,
             ByVal EndDate as vDay    ) as Single
```

Parameters

vPrices required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
StartDate required	ByVal	vDay	Date to start analysis
EndDate required	ByVal	vDay	Date to end analysis

Return Value

Type: Single
Ulcer Index of vPrices from StartDate to EndDate
If vPrices is a ticker symbol, then dividend adjusted prices are used to calculate ulcer index.

Exceptions

-1500000	Missing/invalid ticker symbol
-1600000	Invalid array as vPrices. Length does not equal maxnumdays.
General Exceptions	Error that apply to all methods

Comments

Martin & McCann's book entitled "The Investor's Guide to Fidelity Mutual Funds" originated the Ulcer Index. Standard Deviation is increased by both gains and losses in portfolio value, yet a real investor is only disconcerted by the downside. Rapid increases in price create profits, not risk. Standard Deviation also does not distinguish between randomly occurring gains and losses and very long sequences of losses. Clearly a risk measure is highly desirable that addresses these deficiencies.

Source: Brian Stocks "FastTools v2.07 for FastTrack"

Example

```
Dim sng as Single = UlcerIndex( "OAKMX", "9/1/88", "1/1/11" )

Debug.Print("Result is: " + sng.ToString)

#############

Result is: 11.88075
```

UPIndex

Returns Ulcer Performance Index for vPrices using vBasis as its basis.

Syntax

```
UPIndex(  ByVal vPrices as Object,
          ByVal vBasis as Object,
          ByVal StartDate as vDay,
          ByVal EndDate as vDay    ) as Single
```

Parameters

vPrices required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
vBasis required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
StartDate required	ByVal	vDay	Date to start analysis
EndDate required	ByVal	vDay	Date to end analysis

Return Value

Type: Single
Ulcer Performance Index of vPrices from StartDate to EndDate
If vPrices is a ticker symbol, then dividend adjusted prices are used to calculate ulcer performance index.

Exceptions

-1500000	Missing/invalid ticker symbol
-1600000	Invalid array as vPrices. Length does not equal maxnumdays.
General Exceptions	Error that apply to all methods

Comments

The Ulcer Performance Index is a measure of the Risk Adjusted Return of an investment. It measures how well an investment outperforms a vBasis compared with the amount of ulcers it

gives you. The higher the value, the better the investment.

Subtract the Annual Return of a vBasis from the Annual Return of vPrices. Divide the result by Ulcer Index of vPrices to get the UPI.
Ulcer Performance Index =
(Annualized Return(vPrices) - Annualized Return(vBasis)) / Ulcer Index

Example

```
Dim sng as Single = UPIndex( "OAKMX", "SP-CP", "9/1/88", "1/1/11" )

Debug.Print("Result is: " + sng.ToString)

#############

Result is: 0.2642331
```

SharesOutstanding

Returns the number of shares outstanding for the most recent month.

Syntax

```
SharesOutstanding( ByVal strSym as String) as Single
```

Parameters

strSym required	ByVal	String	Ticker symbol

Return Value

Type: Single
Number of shares outstanding for most recent month.

Exceptions

-1500000	Missing/invalid ticker symbol
General Exceptions	Error that apply to all methods

Comments

Updated aproximately on the 15th of each month for the prior month. Multiply by the current price to get total net assets for the fund or market cap of stock.

Example

```
Dim sng as Single = SharesOutstanding( "SPY" )

Debug.Print("Result is: " + sng.ToString)

#############

Result is: 0
```

MovingAvg

Returns an array containing the exponential moving average of vPrices.

Syntax

```
MovingAvg(  ByVal vPrices as Object,
            ByVal SmoothDays as Integer,
            ByVal StartDate as vDay,
            ByVal EndDate as vDay        ) as Single()
```

Parameters

vPrices required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
SmoothDays required	ByVal	Integer	Number of days to be averaged
StartDate required	ByVal	vDay	Date to start analysis
EndDate required	ByVal	vDay	Date to end analysis

Return Value

Type: Single()
Exponential moving average of vPrices. The result array is always sized to MaxNumDays()
If vPrices is a ticker symbol, then dividend adjusted prices are used to calculate the moving average.

Exceptions

-1500000	Missing/invalid ticker symbol
-1600000	Invalid array as vPrices. Length does not equal maxnumdays.
-1700000	Invalid smoothday
General Exceptions	Error that apply to all methods

Comments

Each day's moving average value is computed as follows:
MA = (Price * SFactor) + (MA of day before * (1 - SFactor))
SFactor = 2 / (SmoothDays + 1)

On the first day, MA is set to the first day's Price. The MA is carried forward to the next day to be used as the prior MA in the formula.

The result array is always sized to MaxNumDays() and all values before StartDate and after EndDate are returned as 0.

Example

```
Dim sng() as Single = MovingAvg( "OAKMX", "24", "9/1/88", "1/1/11" )

Debug.Print("Result is: " + sng.ToString)

For i = 0 to sng.count - 1
  Debug.Print(sng(i).ToString)
Next

#############

Result is:
3.648
3.648
3.648
3.648
3.648
3.648
...
0
0
0
0
0
0
```

SimpleMovingAvg

Returns an array of simple moving averages for vPrices.

Syntax

```
SimpleMovingAvg(  ByVal vPrices as Object,
                  ByVal SmoothDays as Integer,
                  ByVal StartDate as vDay,
                  ByVal EndDate as vDay        ) as Single()
```

Parameters

vPrices required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
SmoothDays required	ByVal	Integer	Number of days to be averaged
StartDate required	ByVal	vDay	Date to start analysis
EndDate required	ByVal	vDay	Date to end analysis

Return Value

Type: Single()
Simple moving average of vPrices. The result array is always sized to MaxNumDays()
If vPrices is a ticker symbol, then dividend adjusted prices are used to calculate the moving average

Exceptions

-1500000	Missing/invalid ticker symbol
-1600000	Invalid array as vPrices. Length does not equal maxnumdays.
-1700000	Invalid smoothday
General Exceptions	Error that apply to all methods

Comments

Each day is an arithmatic average of the preceeding SmoothDays.

The result array is always sized to MaxNumDays() and all values before StartDate and after EndDate are returned as 0.

Example

```
Dim sng() as Single = SimpleMovingAvg( "OAKMX", "24", "9/1/88",
"1/1/11" )

Debug.Print("Result is: " + sng.ToString)

For i = 0 to sng.count - 1
  Debug.Print(sng(i).ToString)
Next

#############

Result is:
3.648
3.648
3.648
3.648
3.648
...
0
0
0
0
0
0
```

RSI

Returns RSI values of vPrices. RSI is also popularly known as Relative Strength Index.

Syntax

```
RSI( ByVal vPrices as Object,
     ByVal Ndays as Integer,
     ByVal StartDate as vDay,
     ByVal EndDate as vDay    ) as Single()
```

Parameters

vPrices required	ByVal	Object	Ticker symbol as String or array of Singles. Array of singles must be of size MaxNumDays.
Ndays required	ByVal	Integer	Specifies the number of periods to include in the calculation
StartDate required	ByVal	vDay	Date to start analysis
EndDate required	ByVal	vDay	Date to end analysis

Return Value

Type: Single()
RSI values of vPrices.
If vPrices is a ticker symbol, then dividend adjusted prices are used to calculate RSI values.

Exceptions

-1500000	Missing/invalid ticker symbol
-1600000	Invalid array as vPrices. Length does not equal maxnumdays.
-1700000	Invalid smoothday
General Exceptions	Error that apply to all methods

Comments

RSI is a trading-range momentum indicator developed by Wells Wilder. RSI is also popularly known as Relative Strength Index. The RSI chart uses one adjustable parameter, Ndays (The number of days averaged). The calculations are based on day-to-day changes in Adjusted Price.

Step 1: Calculate an initial RSI using a simple average

a. Calculate the Simple Average of the Positive changes in value and another Simple Average of Negative changes in value. Both for P number of days. AvgDwn = Sum Positive / P AvgUp = Sum Negative / P
b. Calculate the initial RSI using the Simple Averages of the Positive and Negative changes (This will be the RSI value for day P). Initial RSI = 100 - [100 / (1 + (AvgUp / AvgDwn))]

Step 2: Calculate the remaining AvgUp and AvgDwn values using an Exponential Average for Days > P

a. AvgUp = [(Previous AvgUp * (P-1)) + Today's Negative Change (if there is one)] / P
b. AvgDwn = [(Previous AvgDwn * (P-1)) + Today's Positive Change (is there is one)] / P

Step 3: Calculate the RSI for Days < P

RSI = 100 - [100 / (1 + (AvgUp / AvgDwn))]

FastTrack sets any value of RSI that is less than 1 to 1. This is unique to FastTrack and has no impact on the practical interpretation of the chart.

Example

```
Dim sng() as Single = RSI( "OAKMX", "24", "9/1/88", "1/1/11" )

Debug.Print("Result is: " + sng.ToString)

For i = 0 to sng.count - 1
  Debug.Print(sng(i).ToString)
Next

#############

Result is:
13.19444
13.19444
13.19444
13.19444
13.19444
...
0
0
0
0
0
0
```