EDS Reference Guide

In This Guide

The Rule Builder within AIQ's Expert Design Studio contains **Pre-built Routines** and **Pre-built Functions**. The following is a description of each.

Pre-built Routines

`[AcmDis down price up]`
AcmDis 21-day slope descending 21-day price slope ascending.

`[AcmDis up price down]`
AcmDis slope ascending 21-day price slope descending.

`[AcmDis nonconfirm new high]`
Price at 45-day high not confirmed by high in AcmDis.

`[AcmDis nonconfirm new low]`
Price at 45-day low not confirmed by 45 day low in AcmDis.

`[ADX Trend]`
ADX slope increasing (trend in place) and above the 25 level.

`[ADX no trend]`
ADX slope decreasing (no trend in place).

`[ADX rate up]`
ADX slope increasing (trend in place) and ADX rate is positive.

`[AIQ Upper Band cross down]`
Price crosses from above to below the Upper Band.

`[AIQ Lower Band cross up]`
Price crosses from below to above the Lower Band.

`[BBands tighten]`
Bollinger Bands tighten with low volatility.

`[BBands Upper cross down]`
Price crosses from above to below Upper BB.

`[BBands Lower cross up]`
Price crosses from below to above Lower BB.

`[Candlesticks - Belthold]`
Bullish Belthold pattern during a downtrend.

`[Candlesticks - Engulfing]`
Bullish Engulfing pattern with volume breakout during a downtrend.

`[Candlesticks - Hammer]`
Bullish Hammer pattern with volume breakout during a downtrend.

`[CCI cross below to above 100]`
CCI cuts from below to above 100.

`[CCI cross above to below 100]`
CCI cuts from above to below 100

`[CCI sell short]`
CCI cuts from above -100 to below –100.

`[CCI cover short]`
CCI cuts from below -100 to above -100.

`[Days Ago]`
Rule date minus report date.

`[Date]`
Date of the rule.

`[DIRMOV up and ADX trend]`
ADX slope increasing (trend in place) and DIRMOV is positive.

`[DIRMOV dn and ADX trend]`
ADX slope increasing (trend in place) and DIRMOV is negative.

`[ER up]`
Expert Rating of 95 or greater to the upside today.

`[ER down]`
Expert Rating of 95 or greater to the downside today.

`[Last ER Up]`
Show the date of the last ER to the upside of 95 or greater in the last 49 days.

`[Last ER Down]`
Show the date of the last ER to the downside of 95 or greater in the last 49 days.

`[ESA Upper cross down]`
Price crosses from above to below the Upper ESA.

`[ESA Lower Band cross up]`
Price crosses from below to above the Lower ESA.

`[ESA cross up]`
The ST ESA line crosses through the IT ESA to the upside today.

`[ESA cross down]`
The ST ESA crosses through the IT ESA to the downside today.

`[Gap Up]`
Price gap up – today's low is greater than yesterdays high.

`[Gap Down]`
Price gap down – today's high is lower than yesterdays low.

`[MACD cross up]`
MACDI cross over from below to above the signal line on the day of the report.

`[MACD cross down]`
MACDI cross over from above to below the signal line today.

`[ST MA price cross up]`
Price crosses from below to above the ST MA.

`[IT MA price cross up]`
Price crosses from below to above the IT MA.

`[LT MA price cross up]`
Price crosses from below to above the LT MA.

`[ST MA price cross down]`
Price crosses from above to below ST MA.

`[IT MA price cross down]`
Price crosses from above to below IT MA.

`[LT MA price cross down]`
Price crosses from above to below LT MA.

`[ST MA cross IT MA]`
ST MA crosses IT MA.

`[ST MA IT MA LT MA cross]`
ST MA crosses IT MA and LT MA.

`[ST MA cross LT MA]`
ST MA crosses LT MA.

`[MFRSI over 80]`
MFRSI is greater than 80.

[MFRSI under 20]
MFRSI is lower than 20.

[MFRSI rises above -90]
MFRSI crosses from below to above –90.

[MFRSI falls below 90]
MFRSI crosses from above to below 90.

[Moneyflow divergence dn]
Moneyflow slope is descending while price slope is ascending.

[Moneyflow divergence up]
Moneyflow slope is ascending while price slope is descending.

[Moneyflow at new high]
Moneyflow has reached a new 120-day high.

[Moneyflow at new low]
Moneyflow has reached new 120-day low.

[OBVPct turns positive]
OBVPct crosses from negative to positive.

[OBVPct turns negative]
OBVPct crosses from positive to negative.

[OBVPct divergence down]
OBVPct slope is descending while price slope is ascending.

[OBVPct divergence up]
OBVPct slope is descending while price slope is ascending.

[OBVPct nonconfirm new high]
Price at new 45-day high not confirmed by a 45-day high in OBVPct.

[OBVPct nonconfirm new low]
Price at 45-day low not confirmed by 45-day low in OBVPct.

[OBVPct and VAPct positive]
OBVPct and VAPct are positive.

[OBVPct and VAPct negative]
OBVPct and VAPct are negative.

[OBV down price up]
OBV slope descending while price slope is ascending.

[OBV up price down]
OBV slope ascending while price slope is descending.

[OBV nonconfirm new high]
Price reaches new high not confirmed by high in OBV.

[OBV nonconfirm new low]
Price reaches a new low not confirmed by low in OBV.

[OBV at new high]
OBV has reached a new high.

[OBV at new low]
OBV has reached new low.

[Phase up slope]
The Price Phase indicator today is positive and the 3-day slope is ascending.

[Phase down slope]
The Price Phase indicator today is negative and the 3-day slope is descending.

[Phase turns down]
Phase indicator turns down today (previously going up).

[Phase Turns Up]
Phase indicator turns up today (previously going down).

[Persistence Moneyflow]
VaPct is above zero 90% of the last 120 days.

[Price at new 21 day high]
Price reaches a new 21 day high.

[Price at new 21 day low]
Price reaches a new 21 day low.

[RSI Wilder over 70]
RSI Wilder is in overbought territory.

[RSI Wilder below 30]
RSI Wilder is in oversold territory.

[RSI Wilder falls below 70]
RSI Wilder falls from above to below 70.

[RSI Wilder rises above 30]
RSI Wilder rises from below to above 30.

[RSI AIQ over 70]
RSI AIQ is in overbought territory.

[RSI AIQ below 30]
RSI AIQ is in oversold territory.

[RSI AIQ falls below 70]
RSI AIQ falls from above to below 70.

[RSI AIQ rises above 30]
RSI AIQ rises from below to above 30.

[RSI Wilder Divergence dn]
RSI Wilder slope is descending while price slope is ascending.

[RSI Wilder Divergence up]
RSI Wilder slope is up while price slope is down.

[RSI AIQ Divergence dn]
RSI AIQ slope is descending while price slope is ascending.

[RSI AIQ Divergence up]
RSI AIQ slope is up while price slope is down.

[SK cross SD up]
SK crosses from below to above SD.

[SK cross SD down]
SK crosses from above to below SD.

[SK cross SD up, SD slope up]
SK crosses from below to above SD with SD slope up.

[SK cross SD dn, SD slope dn]
SK crosses from above to below SD with SD slope down.

[SplitVol up and price down]
SplitVol is positive for 5 last days and price is ascending.

[SplitVol down and price up]
SplitVol is negative for 5 days and price is descending.

[Stochastic above 80%]
Stochastic is in overbought territory.

[Stochastic below 20%]
Stochastic is in oversold territory.

[Stochastic dn from above 80%]
Stochastic cuts from above 80% to below 80%.

[Stochastic up from below 20%]
Stochastic cuts from below 20% to above 20%.

[SVMA sell signal]
SVMA moves from positive to negative.

[SVMA buy signal]
SVMA moves from negative to positive.

[SVMA divergence up]
SVMA slope is ascending while price slope is descending.

[SVMA divergence down]
SVMA slope is descending while price slope is ascending.

[Trend up 10 day]
The 10-day slope of the closing price of the ticker is positive.

[Trend down 10 day]
The 10-day slope of the closing price of the ticker is negative.

[Trend Change Down]
The 10-day slope of the closing price of the ticker 5 days ago was positive and the current 5-day slope of the closing price is negative.

[Trend Change Up]
The 10-day slope of the closing price of the ticker 5 days ago was negative and the current 5-day slope of the closing price is positive.

[VaPct turns positive]
VaPct crosses from negative to positive.

[VaPct turns negative]
VaPct crosses from positive to negative.

[VaPct divergence down]
45-day VaPct slope is descending while 45-day price slope is ascending.

[VaPct divergence up]
45-day VaPct slope is ascending while 45-day price slope is descending.

[VaPct nonconfirm new high]
Price at new 45-day high not confirmed by equivalent 45-day high in VaPct.

[VaPct nonconfirm new low]
Price at new 45-day low not confirmed by equivalent 45-day low in VaPct.

[Velocity above zero slope up]
Velocity is greater than zero and slope ascending.

[Velocity below zero slope dn]
Velocity is less than zero and slope descending.

[Volume]
Today’s volume is above the 21-day ESA average.

[Volume Spike]
Today’s volume is 25% higher than today’s ESA of volume.

[Volatility at 21 day lo]
Volatility reaches a 21 day low.

[Volatility at 21 day hi]
Volatility reaches a 21 day high.

[Volatility breakout]
Volatility cuts above it’s moving average after 21 days below.

`[VP Trend dn price up]`
VP Trend slope descending while price slope is ascending.

`[VP Trend up price dn]`
VP Trend slope ascending while price slope is descending.

`[VP Trend nonconfirm new hi]`
Price is at a 45-day high not confirmed by a 45-day high in VP Trend.

`[VP Trend nonconfirm new lo]`
Price is at a 45-day low not confirmed by a 45-day low in VP Trend.

AIQ Reports Duplicated

The Pre-built Routine section also includes routines which duplicate standard reports available in TradingExpert Pro's report area. They are as follows:

`[Reports Wt Action List up ER’s]`
ER to upside 95 or greater last 10 days confirmed by change in direction of the Price Phase indicator in the last 3 days.

`[Reports Wt Action List dn ER’s]`
ER to downside 95 or greater last 10 days confirmed by change in direction of the Price Phase indicator in the last 3 days.

`[Reports - Price chg % 5 days]`
Percentage price change last 5 days.

`[Reports new 21 day high]`
Price reaches a new 21 day high.

`[Reports new 21 day low]`
Price reaches a new 21 day low.

`[Reports Gap Up]`
Price gap up if today’s low is greater than yesterdays high.

`[Reports Gap Down]`
Price gap down if today’s high is less than yesterdays low.

`[Reports Volume Spike]`
Spike if volume is 100% above the ESA of volume.

`[Reports Volume Trend]`
5-day average of volume is twice the 21-day average of volume.

`[Reports cross 2 mov avgs]`
ST MA crosses LT MA sometime in last 5 days.

Abs(number)

Returns the absolute value of a number.

Example:	**num is Abs(-8).**
Explanation:	Return value is 8.

Ceiling(number)

Returns a number rounded to the next highest whole number.

Example:	**num is Ceiling(9.6).**
Explanation:	The return value is 10.

CountOf(rule, periods, [dateoffset]).

Counts the number of times a rule returned TRUE over the given period of time. DateOffset is an optional parameter that if given, will start dateoffset periods prior to the date of the report. Periods can be either weekly or daily depending on the Properties Settings.

Example 1:
Step 1 — define rule:
erup if [er up] >= 95.
Explanation: Base rule returns TRUE if the Up ER was >= 95.
Step 2 — Insert function:
MultipleERs if CountOf(erup,21) > 5.
Explanation: Count the number of ER's in the last 21 days. If more than 5, the rule returns TRUE.

Example 2:	**MultipleERs2 if CountOf(erup,21,5).**
Explanation:	Same as rule above but count starts 5 days ago.

Day()

Returns number corresponding to the day of the week.

Description()

This function returns the descriptive title <"string"> for the currently active ticker.

Exp(number)

Returns a number that is the exponential value of number.

Example:	**num is Exp(2.3026).**
Explanation:	The return value is 10.

ExpAvg(udf, periods, [dateoffset])

Computes the exponentially smoothed moving average of a user defined function (UDF).

Example:	**Eavg is ExpAvg(somefunc,20).**
Explanation:	Returns the exponential smoothed average over 20 day period.

FirstDataDate()

Returns the date of the first data for the current ticker.

Floor(number)

Returns a number rounded to the next lowest whole number.

Example:	**num is Floor(9.6).**
Explanation:	The return value is 9.

FundRank(number)

Returns a Fundamental Rank for the current ticker being tested.

Example:	**rank is FundRank().**
Explanation:	Returns the value of a Fundamental Ranking.

Futures()

Returns TRUE if the current ticker symbol being tested is a Future. Use this function to filter tickers based on a type.

Example:	**FuturesRule if Future() and [close] > 5.**
Explanation:	Close greater than 5 ONLY, for futures.

Group()

Returns TRUE if the current ticker being tested is a Group. Use this function to filter tickers based on a type.

Example:	**GroupRule if Group() and [er up] > 90.**
Explanation:	Er > 90 only for Groups.

`HasDataFor( periods )`

Checks if current ticker has data for the specified number of periods. If sufficient periods with data is found, the function returns the number specified. If the number of periods with data is less than the specified number, it returns the number available.

`HighResult( udf, periods, [dateoffset] )`

This function returns the highest value of an User Defined Function (UDF) over a given period of time. DateOffset is an optional parameter that if given, will start *dateoffset* periods prior to the date of the report. Periods may be weekly or daily depending on the Properties setting.

Step 1 — Define UDF:
AvgPrice is ([high] + [low] + [close]) / 3.
Step 2 — Insert funtion:
HighAvgPrice is HighResult(AvgPrice, 21).
Explanation: Returns the Highest Average Price over 21 days.

`HiVal( field, periods, [dateoffset] )`

This function returns the highest value of an indicator field over a given period of time. DateOffset is an optional parameter that if given, will start dateoffset periods prior to the date of the report. Periods may be weekly or daily depending on the Properties setting.

Example: **HighestER if HiVal([er up],21).**
Explanation: Returns the highest ER in the last 21 days.

`IFF( <rule>, <udf1>, <udf2> )`

This function will return result of udf1 if the rule is TRUE else it will return result of udf2 if the rule is FALSE.

Example: **val is IFF(myrule, myfunc1, myfunc2).**

Explanation: If myrule is TRUE, val equals result of myfunc1. If myrule is FALSE, val equals result of myfunc2..

`Index()`

Returns TRUE if the current ticker being tested is an Index. Use this function to filter tickers based on a type.

Example: **IndexRule if Index() and [close] > 800.**
Explanation: Price greater than 800 only for Indicies.

`Ln( number )`

This function returns the natural logarithm of number.

Example:	**num is Ln(10).**
Explanation:	Return the natural log of 10.

`Log10( number )`

This function returns the base 10 logarithm of number.

Example:	**num is Log10(10).**
Explanation:	Return the Log10 of 10.

`LoVal( field, periods, [dateoffset] )`

This function returns the lowest value of an indicator field over a given period of time. DateOffset is an optional parameter that if given, will start dateoffset periods prior to the date of the report. Periods may be weekly or daily depending on the Properties setting.

Example:	**LowestLow if LoVal([low],21).**
Explanation:	Returns the lowest low for the last 21 days.

`LowResult( udf, periods, [dateoffset] )`

This function returns the lowest value of an User Defined Function (UDF), over a given period of time. DateOffset is an optional parameter that if given, will start dateoffset periods prior to the date of the report. Periods may be weekly or daily depending on the Properties setting.

Step 1 — define UDF:
AvgPrice is ([high] + [low] + [close]) / 3.
Step 2 — insert:
LowAvgPrice is LowResult(AvgPrice, 21).
Explanation: Returns the Lowest Average Price over 21 days.

`MarketCode()`

Returns the code <marketcode> for the market on which the ticker trades.

`MakeDate(<month>,<day>,<year>)`

Returns a serial type date <date> compatible with other EDS functions. Required parameters are:

<month> the month 1-12
<day> the day 1-31
<year> the year 19xx - 20xx

`Market()`

Returns TRUE if the current ticker being tested is a Market. Use this function to filter tickers based on a type.

Step 1 — define UDF:
PositivePhase if [phase] > 0.
Step 2 — insert:
MarketER if Market() and [er up] >= 95 and PositivePhase.
Explanation: Screens only Market Ticker Types for ERs greater than 95 with phase greater than zero.

`Max( number1, number2 )`

This function returns the highest value of either number1 or number2.

Example:	**val is Max(10, 20).**
Explanation:	Returns 20 which is the highest number.

`Month()`

This function returns a number representing the current month.

`Min( number1, number2 )`

This function returns the lowest value of either number1 or number2.

Example:	**val is Min(10, 20).**
Explanation:	Returns 10 which is the lowest number.

`MutualFund()`

Returns TRUE if the current ticker being tested is a MutualFund. Use this function to filter tickers based on a type.

Example:	**MFundRule if MutualFund() and [close] > 5.**
Explanation:	Rule only applies to MutualFunds.

`MutualFundGroup()`

Returns TRUE if the current ticker being tested is a MutualFundGroup. Use this function to filter tickers based on a type.

Example:	**MFundGrpRule if MutualFundGroup() and [close] > 5.**
Explanation:	Rule only applies to MutualFundGroups.

`NoDate()`

This function returns the constant used internally by Expert Design Studio to signify an invalid date type. This number is -32768. Use this function instead of this number for future compatibility.

Example:	**rule if ScanAny(erup,21) <> NoDate().**
Explanation:	True if an ER up was found in the last 21 days.

`OffsetToDate(<month>,<day>,<year>)`

Returns the number of days from the date specified to the current date.

`PortfolioGroup()`

Returns TRUE if the current ticker being tested is a PortfolioGroup. Use this function to filter tickers based on a type.

Example:	**PortRule if PortfolioGroup() and [close] > 5.**
Explanation:	Rule only applies to PortfolioGroups.

`Power( x, y )`

This function returns x raised to the y power.

Example:	**val is Power(2, 10).**
Explanation:	2 to the 10th power

`ReportDate()`

This function returns the date that the report is being run on.

Example:	**Date is ReportDate().**
Explanation:	User Defined Function that returns the report date. Date can now be used as a column in the report.

ResetDate()

This function is used to reset the date back to ReportDate. This function is usefull to remove any date side effects from the ScanAny function described later. This function always returns TRUE.

Step 1 — define UDF:
erup if [er up] > 90.
Step 2 — insert:
Rule if ScanAny(erup,21) and ResetDate().
Explanation: Test for occurance of ER's then reset the date back to the report date.

Round(number)

This function rounds a number to the nearest whole number.

Example1: **val is Round(9.6).**
Explanation: returns 10

Example 2: **val is Round(9.4)**
Explanation2: returns 9

RSIndex()

This function returns the Relative Strength index <"ticker"> for the currently active ticker.

RSTicker()

This function returns the Relative Strength ticker <"ticker"> for the currently active ticker. If you set your RS tickers in Data Manager to your group structure pyramid you can do group analysis in EDS.

RuleDate()

This function returns the date that ScanAny found a rule to be TRUE on.

Step 1 — define rule:
rule if ScanAny(erup,21).
Step 2 — insert:
DaysAgo is ReportDate() - RuleDate().
Explanation: Finds out how many days ago the scanany function found the rule erup to be true.

`ScanAny( rule, periods, [dateoffset] )`

This function is used to perform a rule over a given period of time. If a rule is found to be TRUE during this period, the function returns the date that the rule was TRUE on. If a rule is not found to be true, this function returns -32768. Use the function NoDate() to determine if a valid date was found. The dateoffset parameter is optional and if specified, the scan will start dateoffset periods ago. This function will modifiy t

he internal Date inside of Expert Design Studio. All subsequent values retreived after this rule will be from the date the ScanAny found the rule to be TRUE on. If this is not the desired behavior, follow the ScanAny function with a ResetDate().

Step 1 — define rule:
erup if [er up] > 90.
Step 2 — insert:
AnyErs if ScanAny(erup,21) <> NoDate() and ResetDate().

Explanation:	See if any ER's occured over the last 21 days. Do not keep the date around. Reset it

`Sector()`

Returns TRUE if the current ticker being tested is a Sector. Use this function to filter tickers based on a type.

Example:	**SectorRule if Sector() and [er up] > 90.**
Explanation:	Er > 90 only for Sectors

`SetDate( periods )`

This function is used to set the internal date back the number of periods specified. All Subsequent rules and values obtained will be from this new date. This function always returns TRUE.

Example:	**funkyer if SetDate(5) and [er up] > 90.**
Explanation:	ER 5 days ago.

`SimpleAvg(udf, periods, [dateoffset])`

Computes the simple moving average of a user defined function (UDF).

Example:	**avg is SimpleAvg(somefunc,20).**
Explanation:	Computes the 20 simple average of a function.

`Slope( field, periods, [dateoffset] )`

This function will compute the slope of a given indicator field for the number of periods given. DateOffset is an optional parameter and if given, the slope will be computed starting dateoffset days ago.

Example:	**TrendChange if slope([close],10,5) > 0 and slope([close],5) < 0.**
Explanation:	Trend Changed if the 10 day slope 5 days ago is positive and the most recent 5 day slope is negative.

`Slope2( <udf>, periods, [dateoffset] )`

This function will compute the slope of the parameter defined by a User Defined Function for the number of periods given. DateOffset is an optional parameter and, if given, the slope will be computed starting dateoffset days ago.

Example:	**TrendChange if slope([<udf>],10,5) > 0 and slope([<udf>],5) < 0.**
Explanation:	Trend Change if the 10 day slope 5 days ago is positive and the most recent 5 day slope is negative.

`Sqrt( number )`

This function returns the square root of number.

Example:	**val is sqrt(5*5).**
Explanation:	returns 5

`Stock()`

Returns TRUE if the current ticker being tested is a Stock. Use this function to filter tickers based on a type.

Example:	**StockRule if Stock() and [er up] > 90.**
Explanation:	Er > 90 only for Stocks

`Sum( udf, periods, [dateoffset] )`

This function will sum the result of a given User Defined Function over the given period of time. DateOffset is an optional parameter and if given, the summation will start dateoffset days ago.

Step 1 — define UDF:
AvgPrice is ([high] + [low] + [close]) / 3.
Step 2 — insert:
val is Sum(AvgPrice,21)/21.
Explanation: Compute the 21 day average price.

`Symbol()`

This function returns the ticker symbol <"string"> for the currently active ticker.

`TickerRule(<"ticker">,<rule>)`

Evaluates a specified rule for a specified ticker and returns the boolean result (TRUE or FALSE).

Required parameters are:

<"ticker">	string specifying a ticker name
<rule>	rule name

`TickerUDF(<"ticker">,<udf>)`

Evaluates a specific UDF for a specified ticker and returns the value of the UDF <number or string or date>. The result of the UDF is the same as the UDF return.

Required parameters are:

<"ticker">	string specifying a ticker name
<UDF>	UDF name

`Val( field, periods, dateoffset )`

This function returns the value of an Indicator Field for periods ago. DateOffset is an optional parameter.

Example: **YesterdaysClose is Val([close],1).**
Explanation: Retrieve closing price 1 day ago.

`ValResult( udf, periods, dateoffset )`

This function will return the result of an User Defined Function n periods ago. DateOffset is an optional parameter.

Step 1 — define UDF:
AvgPrice is ([high] + [low] + [close]) / 3.
Step 2 — insert:
num is ValResult(AvgPrice, 5).
Explanation: Get the average price 5 days ago.

`Variance( udf, periods, [dateoffset] )`

This function returns the quantity Variance computed for a User Defined Function as follows:

$$\text{var} = \square\ (x_i\ \square\ x_i)^2 \qquad \text{summation: } i = 1,\ n$$

Where:

$$x_i = \square\ x_i \qquad \text{summation: } i = 1,\ n$$

n = number of periods in moving average

Example:	**StdDev is sqrt(variance([<udf>],10,5))**
Explanation:	Get the Standard Deviation of user defined function 5 days ago using previous 10 values.

`Warrant()`

Returns TRUE if the current ticker being tested is a Warrant. Use this function to filter tickers based on a type.

Example:	**WarrantRule if Warrant() and [er up] > 90.**
Explanation:	Er > 90 only for Warrants

`Year()`

This function returns a number representing the current year.