**AUTOMATED ORDER FLOW (*all order conditions/criteria is listed in detail in this document)***

* Buy Long – this order triggers the following orders
  + 5% Stop Loss
  + Sell 25% of position (to lock in some profit)
* Once the 25% sell order is triggered this triggers
  + Move stop loss to entry price
  + Sell 100% of position (when criteria is met)
* Short position flow will be the same (in reversed direction of course)

**Criteria for trading Auto Bot**

* Automatically execute orders when a stock on a watchlist **AND** scan meets the conditional criteria of my studies/indicators.
* The same (reversed direction) process will be used for short trading
* Each stock may only have 1 open position at a time, so once a entry order (long or short) is executed no other entry signals will be traded until that position is completely closed.
* I trade both sides on most stocks so I need the ability to select if a watchlist/scan should be traded only long, or only short or in both directions.
* I also want the ability to tell the bot to only trade in the direction of the last daily and/or hourly reversal signal (if it was bearish, then only trade short, or bullish, only trade long.) I would like to try using or not using this feature to see which results in higher profitability
* Just to reiterate…The Bot needs the ability to trade a watchlist AND a scanner
* The ability to specify trading hours (for example, If I only want the bot to submit trades between 10am and 3:30pm or whatever times I choose)
* Ability to close all open positions at a certain time of day if I choose to do so.
* I will also need instructions on how to use the bot and how to change the conditions/study indicators it's using.
* I would like the bot with my custom order criteria to be thoroughly back tested for reliability and success rate.
* Orders should execute immediately once all criteria is met (not on candle close)

**BACK TESTING**

Please Provide % success rate of the indicators/strategy listed below. If you feel it can be improved, please also test any additions that you feel increase the success rate. Please do a large volume of trades for each stage of the strategy (preferably on the same stocks on the same day/days) In order to get the most accurate comparison. Weed out any indicators that do not add much value to the strategies success rate/profitability. Most of the scripts for the indicators are listed below. By the end of the back testing we should know which indicators/conditions *combined* together result in the highest level of success (and what that success and profitability rate is.)

**Basic Scanner Criteria to use** *(there’s a lot of criteria to meet in the order conditions so I wouldn’t narrow the stock selection too conservatively otherwise number of trades may be too limited)*

* Stocks with a high enough volatility/range to be profitable day trading.
* Enough volume where buying/selling shares isn’t a problem and there isn’t much difference between the ask & bid
* No trash pump n dump stocks
* Ability to select a min/max share price for the scan
* Ability to weed out stocks that are moving sideways
* Any other criteria you may suggest that would work well with my strategy.

**DEMA TESTING FOR BASELINE SUCCESS & PROFITABILITY RATE**

* 1. DEMA cross orders. This first test should include the following and should trade both long and short with the direction of the trade triggered by the first entry signal received (buy long or sell short) I do not expect these to be highly successful on their own but as we figure out which indicators are most complementary with the DEMA studies the success rate should increase significantly.

***DEMA Buy Long*** *(A**buy order is only submitted if* ***BOTH*** *of these indicators trigger with the specified criteria listed below. Also, please use a stop loss of 10%)*

* + 1. **ABBotBuyLong (*Last signal was/is Bullish) 1 Min TF Condition is also that this is the last signal triggered out of the selection of this indicator and “ABBotSellLong” or ABBotShort Entry”)***

input price1 = LOW;

input length1 = 25;

input displacedEMA1 = 10;

input price2 = HIGH;

input length2 = 70;

input displacedEMA2 = 0;

input averageType1 = AverageType.EXPONENTIAL;

input averageType2 = AverageType.EXPONENTIAL;

input crossingType = {default above, below};

def avg1 = ExpAverage(price1 [-displacedEMA1], length1);

def avg2 = ExpAverage(price2 [-displacedEMA2], length2);

plot signal = Crosses(avg1, avg2, crossingType == crossingType.above)[1];

* + 1. Same indicator as above **EXCEPT on the 5min TF AND must have occurred in 3 bars in order to be considered met criteria for entry.**

***Sell Long 25%***

**AB Ribbon** (*sell 25% triggered by a bearish signal on the* ***3min*** *TF* ***ONLY if the sell price is greater than the entry price!)***

input length = 5;

def abtClose\_org = (open + high + low + close) / 4;

def abtOpen\_org = (abtOpen\_org[1] + abtClose\_org[1]) / 2;

def abtHigh\_org\_temp = Max(abtOpen\_org, abtClose\_org);

def abtHigh\_org = Max(high, abtHigh\_org\_temp);

def abtLow\_org\_temp = Min(abtOpen\_org, abtClose\_org);

def abtLow\_org = Min(low, abtLow\_org\_temp);

plot abtOpen = MovingAverage(AverageType.SIMPLE, abtOpen\_org, length);

def abtHigh = MovingAverage(AverageType.SIMPLE, abtHigh\_org, length);

def abtLow = MovingAverage(AverageType.SIMPLE, abtLow\_org, length);

plot abtClose = MovingAverage(AverageType.SIMPLE, abtClose\_org, length);

abtOpen.SetDefaultColor(Color.GRAY);

abtClose.SetDefaultColor(Color.GRAY);

AddCloud(abtOpen, abtClose, Color.RED, Color.GREEN, yes);

def flag\_green = abtClose > abtOpen;

def flag\_red = abtClose < abtOpen;

def flag\_green\_1 = abtClose > abtOpen and abtClose[1] <= abtOpen[1];

def flag\_red\_1 = abtClose < abtOpen and abtClose[1] >= abtOpen[1];

def bull\_pre\_1;

def bearish\_pre\_1;

def bull\_pre\_2;

def bearish\_pre\_2;

def bull\_pre\_3;

def bearish\_pre\_3;

bull\_pre\_1 = if flag\_green\_1 then Close else bull\_pre\_1[1];

bull\_pre\_2 = if flag\_green\_1 then bull\_pre\_1[1] else bull\_pre\_2[1];

bull\_pre\_3 = if flag\_green\_1 then bull\_pre\_2[1] else bull\_pre\_3[1];

def bull = if flag\_green then Close > bull\_pre\_2 else 0;

bearish\_pre\_1 = if flag\_red\_1 then Close else bearish\_pre\_1[1];

bearish\_pre\_2 = if flag\_red\_1 then bearish\_pre\_1[1] else bearish\_pre\_2[1];

bearish\_pre\_3 = if flag\_red\_1 then bearish\_pre\_2[1] else bearish\_pre\_3[1];

def bearish = if flag\_red then Close < bearish\_pre\_2 else 0;

# Trend Change Signals (Boolean Options)

def bullTrendChange = flag\_green\_1 and !flag\_green\_1[1]; # Trend changes from bearish to bullish

def bearishTrendChange = flag\_red\_1 and !flag\_red\_1[1]; # Trend changes from bullish to bearish

# Trend Change Plot

plot BullTrendSignal = bullTrendChange;

BullTrendSignal.SetDefaultColor(Color.GREEN);

BullTrendSignal.SetStyle(Curve.POINTS);

BullTrendSignal.SetLineWeight(3);

plot BearishTrendSignal = bearishTrendChange;

BearishTrendSignal.SetDefaultColor(Color.RED);

BearishTrendSignal.SetStyle(Curve.POINTS);

BearishTrendSignal.SetLineWeight(3);

# Optional boolean flags to show trend changes on the chart

input showBullTrend = yes;

input showBearishTrend = yes;

# Conditionally show trend change signals

BullTrendSignal.SetHiding(!showBullTrend);

BearishTrendSignal.SetHiding(!showBearishTrend);

def n = 5;

def showLines = no;

def showValues = no;

def showBarNumbers = no;

def ExtensionLengthBars = 20;

def UpperExtensionPercentLimit = 5;

def LowerExtensionPercentLimit = 5;

def vHigh = high;

def vLow = low;

def vOpen = open;

def vClose = close;

def vVolume = volume;

def nan = Double.NaN;

def bn = BarNumber();

def currentBar = HighestAll(if !IsNaN(vHigh) then bn else nan);

def x\_AxisLastExpansionBar = BarNumber() == HighestAll(BarNumber());

# Additional code for pivot high/low, extension lines, etc., as in the original script.

**WHEN THE SELL 25% ORDER HAS BEEN TRIGGERED, THE STOP LOSS ORDER SHOULD BE MOVED TO THE LONG POSITION ENTRY PRICE**

***DEMA Sell 100% Long*** *(a sell order for 100% of the position should be placed when* ***ONE*** *of the following 2 indicators is triggered bearish on the* ***5 min TF****)*

* + 1. **ABBOTSellLong**

input price1 = HIGH;

input length1 = 9;

input displacedEMA1 = 22;

input price2 = LOW;

input length2 = 20;

input displacedEMA2 = 0;

input averageType1 = AverageType.EXPONENTIAL;

input averageType2 = AverageType.EXPONENTIAL;

input crossingType = {default above, below};

def avg1 = ExpAverage(price1 [-displacedEMA1], length1);

def avg2 = ExpAverage(price2 [-displacedEMA2], length2);

plot signal = Crosses(avg1, avg2, crossingType == crossingType.below)[1];

* + 1. **DEMAShortEntry**  (the *second indicator that when triggered bearish would submit a sell 100% of long position order )*

input price1 = LOW;

input length1 = 25;

input displacedEMA1 = 10;

input price2 = HIGH;

input length2 = 70;

input displacedEMA2 = 0;

input averageType1 = AverageType.EXPONENTIAL;

input averageType2 = AverageType.EXPONENTIAL;

input crossingType = {default above, below};

def avg1 = ExpAverage(price1 [-displacedEMA1], length1);

def avg2 = ExpAverage(price2 [-displacedEMA2], length2);

plot signal = Crosses(avg1, avg2, crossingType == crossingType.below)[1];

***DEMA Short Entry***

*(same indicator and parameters as the* **DEMAShortEntry** *script above also triggered bearish on the* ***5 min*** *TF, However, If this indicator triggers a sell 100% long order can it also immediately open a short position? If not, then a VERY minimal offset of one or the other would be acceptable in order to give it a second to sell the long position and as quickly as possible open a short position.*

***DEMA Short Cover 25%***

**AB Ribbon** *cover 25% of short position when this indicator is triggered bullish on the* ***3min*** *TF. (This is the same script as the cover long 25% except the signal triggered is bullish instead of bearish.)* ***ONLY if the price is less than the entry price!)***

***DEMA Short Cover 100%***

*(Triggers on the 5 min TF)*

input price1 = HIGH;

input length1 = 9;

input displacedEMA1 = 22;

input price2 = LOW;

input length2 = 20;

input displacedEMA2 = 0;

input averageType1 = AverageType.EXPONENTIAL;

input averageType2 = AverageType.EXPONENTIAL;

input crossingType = {default above, below};

def avg1 = ExpAverage(price1 [-displacedEMA1], length1);

def avg2 = ExpAverage(price2 [-displacedEMA2], length2);

plot signal = Crosses(avg1, avg2, crossingType == crossingType.above)[1];

**STRATEGIES TO BACK TEST FOR LONG ENTRY**

*(#1 is the* ***DEMA Buy Long*** *script in the above baseline testing.)*

* 1. #1 + Only trade in direction of the last daily signal (bullish or bearish) **AB Trend Reversal**
  2. #1 + Only trade in the direction of the last hourly signal (bullish or bearish) **AB Trend Reversal**
  3. Is using #1-3 Together taking into account the daily AND hourly direction signal the most profitable? **AB Trend Reversal**
  4. Using the most successful combination of #1-4 test to see if only trading in the direction of the overall market on the day improves success **(please use a prebuilt script in trading view for this one)**
  5. Using the most successful combination of #1-5 Is it more profitable to trade 9am-4pm only, 10am-3:30pm only, or (if the software allows) 4am-8pm? **please use a prebuilt script in trading view for this one)**

**STRATEGIES TO BACK TEST FOR ALL ORDER TYPES #7-15**

***(The scripts for #7-14 listed below are all based on a LONG ENTRY, please change/reverse***

***parameters for testing “Sell Long” “Short Entry” and “Short Close” where .)***

* 1. Using the most successful combination of #1-6 add the alignment of the 1, 3, & 5 minute TF using the **ABFibAvg** indicator *(Last signal was/is “upPt” bullish)*
  2. Using the most successful combination of #1-7 add in the **DMI Reversal** signals on the 3min TF (*last signal was bullish)*
  3. Using the most successful combination of #1-8 add in the **DMI Reversal** signals on the 5min TF *(last signal was bullish)* ***this trigger must occur within 3 bars in order to be considered a “met criteria” for a long entry order.***
  4. Using the most successful combination of #1-9 add in the **AB Ribbon** indicator on the 1min, 3, & 5 min TF *(Last signal was/is bullish)*
  5. Using the most successful combination of #1-10 and add in the **MovingAvgCross** on the 3 min TF *(Last signal was cross above)*
  6. Using the most successful combination of #1-11 and add in the **MovingAvgCross** on the 5 min TF ***this trigger must occur within 3 bars in order to be considered a “met criteria” for a long entry order.***
  7. Using the most successful combination of #1-12 add in the **Parabolic SAR Crossover 3 min** on the 3min TF *(Last signal crossed under (Bullish))*
  8. Using the most successful combination of #1-14 add in the **Parabolic SAR Crossover 5 min** on the 5min TF ***please note that there are different parameters for this script (more than just the TF) it is NOT the same script as the 3min SAR cross***
  9. Using the most successful combination of #1-15 add in a requirement for the **MACD** to have crossed for the 1, 3, & 5 min TFs to see if that improves the success rate. **(please use a prebuilt script in trading view for this one)**

**# 2, 3, & 4 script AB TREND REVERSAL DAILY (*HOURLY IS 4,3,1,no)***

#Automatically show the bullish(buy) / bearish(sell) signals

input length = 8;

input threshold = 3;

input scale = 1;

input show\_all\_signals=yes;

def abtClose\_org = (open + high + low + close) / 4;

def abtOpen\_org = (abtOpen\_org[1] + abtClose\_org[1]) / 2;

def abtHigh\_org\_temp = Max(abtOpen\_org, abtClose\_org);

def abtHigh\_org = Max(high, abtHigh\_org\_temp);

def abtLow\_org\_temp = Min(abtOpen\_org, abtClose\_org);

def abtLow\_org = Min(low, abtLow\_org\_temp);

def abtOpen = MovingAverage(AverageType.SIMPLE, abtOpen\_org, length);

def abtHigh = MovingAverage(AverageType.SIMPLE, abtHigh\_org, length);

def abtLow = MovingAverage(AverageType.SIMPLE, abtLow\_org, length);

def abtClose = MovingAverage(AverageType.SIMPLE, abtClose\_org, length);

def flag\_green = abtClose > abtOpen;

def flag\_red = abtClose < abtOpen;

def flag\_green\_1 = abtClose > abtOpen and abtClose[1] <= abtOpen[1];

def flag\_red\_1 = abtClose < abtOpen and abtClose[1] >= abtOpen[1];

def bull\_pre\_1;

def bearish\_pre\_1;

def bull\_pre\_2;

def bearish\_pre\_2;

def bull\_pre\_3;

def bearish\_pre\_3;

bull\_pre\_1 = if flag\_green\_1 then close else bull\_pre\_1[1];

bull\_pre\_2 = if flag\_green\_1 then bull\_pre\_1[1] else bull\_pre\_2[1];

bull\_pre\_3 = if flag\_green\_1 then bull\_pre\_2[1] else bull\_pre\_3[1];

def bull = if flag\_green then close > bull\_pre\_2 \* (1 + threshold / (1000 \* scale)) else 0;

bearish\_pre\_1 = if flag\_red\_1 then close else bearish\_pre\_1[1];

bearish\_pre\_2 = if flag\_red\_1 then bearish\_pre\_1[1] else bearish\_pre\_2[1];

bearish\_pre\_3 = if flag\_red\_1 then bearish\_pre\_2[1] else bearish\_pre\_3[1];

def bearish = if flag\_red then close < bearish\_pre\_2 \* (1 - threshold / (1000 \* scale)) else 0;

def isAbove\_temp = if bull and !bull[1] then bull else 0;

def isBelow\_temp = if bearish and !bearish[1] then bearish else 0;

def direction = if isAbove\_temp then 1 else (if isBelow\_temp then 0 else direction[1]);

def isAbove\_temp\_2 = if direction then 1 else 0;

def isBelow\_temp\_2 = if !direction then 1 else 0;

def isAbove\_temp\_3 = if isAbove\_temp\_2 and isBelow\_temp\_2[1] then 1 else 0;

def isBelow\_temp\_3 = if isBelow\_temp\_2 and isAbove\_temp\_2[1] then 1 else 0;

plot isAbove = if show\_all\_signals then isAbove\_temp else isAbove\_temp\_3;

plot isBelow = if show\_all\_signals then isBelow\_temp else isBelow\_temp\_3;

isAbove.SetPaintingStrategy(PaintingStrategy.BOOLEAN\_ARROW\_UP);

isBelow.SetPaintingStrategy(PaintingStrategy.BOOLEAN\_ARROW\_DOWN);

isAbove.SetDefaultColor(Color.GREEN);

isBelow.SetDefaultColor(Color.RED);

isAbove.SetLineWeight(5);

isBelow.SetLineWeight(5);

def n = 5;

def showLines = no;

def showValues = no;

def showBarNumbers = no;

def ExtensionLengthBars = 20;

def UpperExtensionPercentLimit = 5;

def LowerExtensionPercentLimit = 5;

def vHigh = high;

def vLow = low;

def vOpen = open;

def vClose = close;

def vVolume = volume;

def nan = Double.NaN;

def bn = BarNumber();

def currentBar = HighestAll(if !IsNaN(vHigh) then bn else nan);

def x\_AxisLastExpansionBar = BarNumber() == HighestAll(BarNumber());

def PH;

def PL;

def hh = fold i = 1 to n + 1

with p = 1

while p

do vHigh > getValue(vHigh, -i);

PH = if (bn > n and

vHigh == highest(vHigh, n) and

hh)

then vHigh

else double.NaN;

def ll = fold j = 1 to n + 1

with q = 1

while q

do vLow < getValue(low, -j);

PL = if (bn > n and

vLow == lowest(vLow, n) and

ll)

then vLow

else double.NaN;

def PHBar = if !isNaN(PH)

then bn

else PHBar[1];

def PLBar = if !isNaN(PL)

then bn

else PLBar[1];

def PHL = if !isNaN(PH)

then PH

else PHL[1];

def priorPHBar = if PHL != PHL[1]

then PHBar[1]

else priorPHBar[1];

def PLL = if !isNaN(PL)

then PL

else PLL[1];

def priorPLBar = if PLL != PLL[1]

then PLBar[1]

else priorPLBar[1];

def HighPivots = bn >= highestAll(priorPHBar);

def LowPivots = bn >= highestAll(priorPLBar);

def FirstRpoint = if HighPivots

then bn - PHBar

else 0;

def PriorRpoint = if HighPivots

then bn - PriorPHBar

else 0;

def RSlope = (getvalue(PH, FirstRpoint) - getvalue(PH, PriorRpoint))

/ (PHBar - PriorPHBar);

def FirstSpoint = if LowPivots

then bn - PLBar

else 0;

def PriorSpoint = if LowPivots

then bn - PriorPLBar

else 0;

def SSlope = (getvalue(PL, FirstSpoint) - getvalue(PL, PriorSpoint))

/ (PLBar - PriorPLBar);

def RExtend = if bn == highestall(PHBar)

then 1

else RExtend[1];

def SExtend = if bn == highestall(PLBar)

then 1

else SExtend[1];

def pivotHigh = if HighPivots

then PH

else double.NaN;

# pivotHigh.SetDefaultColor(GetColor(1));

# pivotHigh.setPaintingStrategy(PaintingStrategy.VALUES\_ABOVE);

# pivotHigh.setHiding(!showValues);

def pivotHighLine = if PHL > 0 and

HighPivots

then PHL

else double.NaN;

# pivotHighLine.SetPaintingStrategy(PaintingStrategy.DASHES);

# pivotHighLine.setDefaultColor(color.uptick);

# pivotHighLine.setHiding(!showLines);

def RLine = pivotHigh;

# RLine.enableApproximation();

# RLine.SetDefaultColor(Color.LIGHT\_GRAY);

# RLine.SetStyle(Curve.Short\_DASH);

def calc\_ResistanceExtension = if RExtend

then (bn - PHBar) \* RSlope + PHL

else double.NaN;

def line\_ResistanceExtension = if bn <= (Currentbar + ExtensionLengthBars)

and calc\_ResistanceExtension[1] >= (lowestall(vLow) \* (1-(lowerExtensionPercentLimit/100)))

and calc\_ResistanceExtension[1] <= (Highestall(vHigh) \* (1 + (upperExtensionPercentLimit/100)))

then calc\_ResistanceExtension else double.nan;

# line\_ResistanceExtension.SetStyle(Curve.Short\_DASH);

# line\_ResistanceExtension.SetDefaultColor(color.LIGHT\_GRAY);

# line\_ResistanceExtension.setLineWeight(1);

def pivotLow = if LowPivots

then PL

else double.NaN;

# pivotLow.setDefaultColor(GetColor(4));

# pivotLow.setPaintingStrategy(PaintingStrategy.VALUES\_BELOW);

# pivotLow.setHiding(!showValues);

def pivotLowLine = if PLL > 0 and

LowPivots

then PLL

else double.NaN;

# pivotLowLine.SetPaintingStrategy(PaintingStrategy.DASHES);

# pivotLowLine.setDefaultColor(color.DOWNTICK);

# pivotLowLine.setHiding(!showLines);

def SupportLine = pivotLow;

# SupportLine.enableApproximation();

# SupportLine.SetDefaultColor(color.LIGHT\_GRAY);

# SUpportLine.SetStyle(Curve.Short\_DASH);

def calc\_SupportExtension = if SExtend

then (bn - PLBar) \* SSlope + PLL

else double.NaN;

def line\_SupportExtension = if bn <= (Currentbar + ExtensionLengthBars)

and calc\_SupportExtension[1] >= (lowestall(vLow) \* (1-(lowerExtensionPercentLimit/100)))

and calc\_SupportExtension[1] <= (Highestall(vHigh) \* (1 + (upperExtensionPercentLimit/100)))

then calc\_supportExtension else double.nan;

# line\_SupportExtension.SetDefaultColor(color.LIGHT\_GRAY); #was 7

# line\_SupportExtension.SetStyle(Curve.Short\_DASH);

# line\_SupportExtension.setLineWeight(1);

def BarNumbersBelow = bn;

# BarNumbersBelow.SetDefaultColor(GetColor(0));

# BarNumbersBelow.setHiding(!showBarNumbers);

# BarNumbersBelow.SetPaintingStrategy(PaintingStrategy.VALUES\_BELOW);

def PivotDot = if !isNaN(pivotHigh)

then pivotHigh

else if !isNaN(pivotLow)

then pivotLow

else double.NaN;

# pivotDot.SetDefaultColor(GetColor(7));

# pivotDot.SetPaintingStrategy(PaintingStrategy.POINTS);

# pivotDot.SetLineWeight(3);

**#7 Script ABFibAvg**

***THIS STUDY MUST BE BULLISH ON THE 1, 3, AND 5 MIN TFs IN ORDER FOR IT TO MEET THE LONG BUY ORDER CRITERIA. .***T*he following parameters in the images below need to be used in this script (not the default parameters that are currently in the script below the images.) This will be 3 different indicators. One for the 1, one for the 3, and one for the 5min TFs.*

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

input agg1 = aggregationPeriod.four\_hours;

def Source = close(period = agg1);

input AverageTypes = AverageType.EXPONENTIAL;

def AveragePrices = Source;

input AverageLength1 = 144;

input AverageLength2 = 126;

input AverageLength3 = 108;

input AverageLength4 = 90;

input AverageLength5 = 72;

input AverageLength6 = 54;

input AverageLength7 = 27;

input AverageLength8 = 18;

def avg1 = MovingAverage(AverageTypes, AveragePrices, AverageLength1) \* 34;

def avg2 = MovingAverage(AverageTypes, AveragePrices, AverageLength2) \* 21;

def avg3 = MovingAverage(AverageTypes, AveragePrices, AverageLength3) \* 13;

def avg4 = MovingAverage(AverageTypes, AveragePrices, AverageLength4) \* 8;

def avg5 = MovingAverage(AverageTypes, AveragePrices, AverageLength5) \* 5;

def avg6 = MovingAverage(AverageTypes, AveragePrices, AverageLength6) \* 3;

def avg7 = MovingAverage(AverageTypes, AveragePrices, AverageLength7) \* 2;

def avg8 = MovingAverage(AverageTypes, AveragePrices, AverageLength8);

def typfibavg = (avg1 + avg2 + avg3 + avg4 + avg5 + avg6 + avg7 + avg8) / 87;

plot TypicalFibAverage = typfibavg;

TypicalFibAverage.SetLineWeight(3);

# Input to control the visibility of trend change signals

input showBuySignal = yes; # Boolean to toggle buy signals

input showSellSignal = yes; # Boolean to toggle sell signals

# Supertrend and other functions

input maType = {SMA, default EMA, HMA, WMA, VWMA, DEMA, ZLSMA, ZLEMA, VAR, TILL, TSF};

input maLength = 135; #100 " Length MA"

input changeATR = yes;

input atrLength = 10; # " Length Atr"

input atrMultiplier = 0.5; # "Band Mult"

input ShowCloud = no;

input ShowCloud2 = yes;

input showsignals = no;

def na = Double.NaN;

def avPlot = ohlc4;

#// ] -------------- FUNCTIONS : Moving Avg ------------------ [

#pine\_linreg(src, len, offset=0) =>

Script linreg {

input src = close;

input len = 100;

input offset = 0;

def na = Double.NaN;

def bar\_index = isNaN(close);

def x\_sum = if bar\_index then na else

fold i = 0 to len with p do

p + i;

def xx\_sum = if bar\_index then na else

fold ii = 0 to len with pp do

pp + ii \* ii;

def y\_sum = sum(src, len);

def xy\_sum = fold j = 0 to len with q do

q + j \* GetValue(src, len - j - 1);

def slope = (len \* xy\_sum - x\_sum \* y\_sum) / (len \* xx\_sum - x\_sum \* x\_sum);

def intercept = (y\_sum - slope \* x\_sum) / len;

def linreg = intercept + slope \* (len - offset - 1);

plot out = linreg;

}

Script TSF {

input src = close;

input length = 100;

def lrc = Inertia(src, length);

def lrc1 = linreg(src, length, 1);

def lrs = lrc - lrc1;

def TSF = Inertia(src, length) + lrs;

plot out = TSF;

}

script TILL {

input src = close;

input length = 100;

input T3a1 = 0.7;

def T3e1 = ExpAverage(src, length);

def T3e2 = ExpAverage(T3e1, length);

def T3e3 = ExpAverage(T3e2, length);

def T3e4 = ExpAverage(T3e3, length);

def T3e5 = ExpAverage(T3e4, length);

def T3e6 = ExpAverage(T3e5, length);

def T3c1 = -T3a1 \* T3a1 \* T3a1;

def T3c2 = 3 \* T3a1 \* T3a1 + 3 \* T3a1 \* T3a1 \* T3a1;

def T3c3 = -6 \* T3a1 \* T3a1 - 3 \* T3a1 - 3 \* T3a1 \* T3a1 \* T3a1;

def T3c4 = 1 + 3 \* T3a1 + T3a1 \* T3a1 \* T3a1 + 3 \* T3a1 \* T3a1;

def T3 = T3c1 \* T3e6 + T3c2 \* T3e5 + T3c3 \* T3e4 + T3c4 \* T3e3;

plot out = T3;

}

script VAR {

input src = close;

input length = 100;

def curClose = src;

def prevClose = src[1];

def valpha = 2 / (length + 1);

def vud1 = if curClose > prevClose then curClose - prevClose else 0;

def vdd1 = if prevClose > curClose then prevClose - curClose else 0;

def vUD = sum(vud1, 9);

def vDD = sum(vdd1, 9);

def vCMO = if vUD + vDD == 0 then 0 else (vUD - vDD) / (vUD + vDD);

def cmo = valpha \* AbsValue(vCMO);

def VAR = CompoundValue(1, cmo \* curClose + (1 - cmo) \* VAR[1], src);

plot out = VAR;

}

#export zlSma(float src, simple int len) =>

script ZLSMA {

input src = close;

input len = 14;

def lsma = Inertia(src, len);

def lsma2 = Inertia(lsma, len);

def eq = lsma - lsma2;

def zlsma = lsma + eq;

plot return = zlsma;

}

#export zlSma(float src, simple int len) =>

script ZLEMA {

input src = close;

input len = 14;

def lenR = Round(len / 2, 0);

def zxLag = if len / 2 == lenR then len / 2 else (len - 1) / 2;

def zxEMAData = src + src - src[zxLag];

def ZLEMA = ExpAverage(zxEMAData, len);

plot out = ZLEMA;

}

#vwma(source, length)

script VWMA {

input src = close;

input len = 15;

def v = volume;

def VWMA = SimpleMovingAvg(src \* v, len) / SimpleMovingAvg(v, len);

plot result = VWMA;

}

#export multiMa(float source, simple int length, string type) =>

def mov;

Switch (maType) {

Case SMA : mov = Average(source, maLength);

Case VAR : mov = VAR(source, maLength);

Case WMA : mov = WMA(source, maLength);

Case VWMA : mov = VWMA(source, maLength);

Case DEMA : mov = DEMA(source, maLength);

Case ZLSMA : mov = ZLSMA(source, maLength);

Case ZLEMA : mov = ZLEMA(source, maLength);

Case TILL : mov = TILL(source, maLength);

Case TSF : mov = TSF(source, maLength);

Case HMA : mov = HullMovingAvg(source, maLength);

Default : mov = ExpAverage(source, maLength);

}

script Supertrend {

input src = hl2;

input atrLength = 14;

input mult = 2;

input changeATR = yes;

def tr = TrueRange(high, close, low);

def nATR = if changeATR then ATR(Length = atrLength) else SimpleMovingAvg(tr, atrLength);

def up = src - mult \* nATR;

def dn = src + mult \* nATR;

def lower;

def upper;

def up2 = if isNaN(upper[1]) then up else upper[1];

def dn2 = if isNaN(lower[1]) then dn else lower[1];

def up1 = if up2 == 0 then up else up2;

def dn1 = if dn2 == 0 then dn else dn2;

upper = if (close[1] > up1) then if up > up1 then up else up1 else up;

lower = if (close[1] < dn1) then if dn < dn1 then dn else dn1 else dn;

def trend;

def trend2 = if isNaN(trend[1]) then 0 else trend[1];

def trend1 = if trend2 == 0 then 1 else trend2;

if trend1 == -1 and close > dn1 {

trend = 1;

} else if trend1 == 1 and close < up1 {

trend = -1;

} else {

trend = trend1;

}

def superTrend = if trend == -1 then lower else upper;

plot ST = superTrend;

plot dir = trend;

}

def ma = mov;

def ST = Supertrend(ma, atrLength, atrMultiplier, changeATR).ST;

def Trend = Supertrend(ma, atrLength, atrMultiplier, changeATR).dir;

def change = Trend != Trend[1];

def buySignal = Trend > 0 and change;

def sellSignal = Trend < 0 and change;

# Plots for Trend

plot upPlot = if Trend > 0 then ST else na; # 'Up Band'

plot loPlot = if Trend < 0 then ST else na; # 'Low Band'

upPlot.SetDefaultColor(Color.Green);

loPlot.SetDefaultColor(Color.Red);

upPlot.SetLineWeight(3);

loPlot.SetLineWeight(3);

# Conditional plotting for buy and sell signals

plot upPt = if showBuySignal and buySignal then ST else na;

plot dnPt = if showSellSignal and sellSignal then ST else na;

upPt.SetLineWeight(2);

dnPt.SetLineWeight(2);

upPt.SetDefaultColor(Color.Green);

dnPt.SetDefaultColor(Color.Red);

upPt.SetPaintingStrategy(PaintingStrategy.POINTS);

dnPt.SetPaintingStrategy(PaintingStrategy.POINTS);

TypicalFibAverage.SetDefaultColor(Color.Green);

TypicalFibAverage.AssignValueColor(if TypicalFibAverage < loPlot then Color.Orange else Color.Current);

# Signals and Cloud

AddChartBubble(showsignals and buySignal and showBuySignal, ST, "B", Color.Green, no);

AddChartBubble(showsignals and sellSignal and showSellSignal, ST, "S", Color.Red, no);

AddCloud(If(TypicalFibAverage < loPlot, TypicalFibAverage, Double.NaN), loPlot, Color.Red, Color.Red);

AddCloud(If(TypicalFibAverage > upPlot, TypicalFibAverage, Double.NaN), upPlot, Color.Green, Color.Green);

AddCloud(if ShowCloud then avPlot else na, upPlot, CreateColor(0, 35, 0)); # 'Upper Area'

AddCloud(if ShowCloud then loPlot else na, avPlot, CreateColor(55, 0, 0)); # 'Lower Area'

**#8 & 9 Script DMI Reversal Alerts** *(Last signal was/is bullish)*

input length = 200;

input highLowLength = 200;

input sumLength = 14;

input averageLength = 5;

def stoch = DMI\_StochasticExtreme(length, highLowLength, sumLength);

def avgStoch = Average(stoch, averageLength);

plot Above = stoch crosses above avgStoch;

plot Below = stoch crosses below avgStoch;

Below.SetDefaultColor(Color.DOWNTICK);

Below.SetPaintingStrategy(PaintingStrategy.BOOLEAN\_ARROW\_DOWN);

Above.SetDefaultColor(Color.UPTICK);

Above.SetPaintingStrategy(PaintingStrategy.BOOLEAN\_ARROW\_UP);

**#10 Script**  **AB Ribbon** *(THIS STUDY MUST BE BULLISH ON THE 1, 3, AND 5 MIN TFs IN ORDER FOR IT TO MEET THE LONG BUY ORDER CRITERIA.*

input length = 10;

def abtClose\_org = (open + high + low + close) / 4;

def abtOpen\_org = (abtOpen\_org[1] + abtClose\_org[1]) / 2;

def abtHigh\_org\_temp = Max(abtOpen\_org, abtClose\_org);

def abtHigh\_org = Max(high, abtHigh\_org\_temp);

def abtLow\_org\_temp = Min(abtOpen\_org, abtClose\_org);

def abtLow\_org = Min(low, abtLow\_org\_temp);

plot abtOpen = MovingAverage(AverageType.SIMPLE, abtOpen\_org, length);

def abtHigh = MovingAverage(AverageType.SIMPLE, abtHigh\_org, length);

def abtLow = MovingAverage(AverageType.SIMPLE, abtLow\_org, length);

plot abtClose = MovingAverage(AverageType.SIMPLE, abtClose\_org, length);

abtOpen.SetDefaultColor(Color.GRAY);

abtClose.SetDefaultColor(Color.GRAY);

AddCloud(abtOpen, abtClose, Color.RED, Color.GREEN, yes);

def flag\_green = abtClose > abtOpen;

def flag\_red = abtClose < abtOpen;

def flag\_green\_1 = abtClose > abtOpen and abtClose[1] <= abtOpen[1];

def flag\_red\_1 = abtClose < abtOpen and abtClose[1] >= abtOpen[1];

def bull\_pre\_1;

def bearish\_pre\_1;

def bull\_pre\_2;

def bearish\_pre\_2;

def bull\_pre\_3;

def bearish\_pre\_3;

bull\_pre\_1 = if flag\_green\_1 then Close else bull\_pre\_1[1];

bull\_pre\_2 = if flag\_green\_1 then bull\_pre\_1[1] else bull\_pre\_2[1];

bull\_pre\_3 = if flag\_green\_1 then bull\_pre\_2[1] else bull\_pre\_3[1];

def bull = if flag\_green then Close > bull\_pre\_2 else 0;

bearish\_pre\_1 = if flag\_red\_1 then Close else bearish\_pre\_1[1];

bearish\_pre\_2 = if flag\_red\_1 then bearish\_pre\_1[1] else bearish\_pre\_2[1];

bearish\_pre\_3 = if flag\_red\_1 then bearish\_pre\_2[1] else bearish\_pre\_3[1];

def bearish = if flag\_red then Close < bearish\_pre\_2 else 0;

# Trend Change Signals (Boolean Options)

def bullTrendChange = flag\_green\_1 and !flag\_green\_1[1]; # Trend changes from bearish to bullish

def bearishTrendChange = flag\_red\_1 and !flag\_red\_1[1]; # Trend changes from bullish to bearish

# Trend Change Plot

plot BullTrendSignal = bullTrendChange;

BullTrendSignal.SetDefaultColor(Color.GREEN);

BullTrendSignal.SetStyle(Curve.POINTS);

BullTrendSignal.SetLineWeight(3);

plot BearishTrendSignal = bearishTrendChange;

BearishTrendSignal.SetDefaultColor(Color.RED);

BearishTrendSignal.SetStyle(Curve.POINTS);

BearishTrendSignal.SetLineWeight(3);

# Optional boolean flags to show trend changes on the chart

input showBullTrend = yes;

input showBearishTrend = yes;

# Conditionally show trend change signals

BullTrendSignal.SetHiding(!showBullTrend);

BearishTrendSignal.SetHiding(!showBearishTrend);

def n = 5;

def showLines = no;

def showValues = no;

def showBarNumbers = no;

def ExtensionLengthBars = 20;

def UpperExtensionPercentLimit = 5;

def LowerExtensionPercentLimit = 5;

def vHigh = high;

def vLow = low;

def vOpen = open;

def vClose = close;

def vVolume = volume;

def nan = Double.NaN;

def bn = BarNumber();

def currentBar = HighestAll(if !IsNaN(vHigh) then bn else nan);

def x\_AxisLastExpansionBar = BarNumber() == HighestAll(BarNumber());

**#11 & 12 Script MovingAvgCross**

*(Last signal was cross above on the 3 min TF****) For the 5 min TF (NOT the 3 min) this trigger must occur within 3 bars in order to be considered a “met criteria” for a long entry order.***

#wizard input: length1

#wizard text: -period

#wizard input: averageType1

#wizard text: crosses

#wizard input: crossingType

#wizard input: length2

#wizard text: -period

#wizard input: averageType2

#wizard text: Price:

#wizard input: price

input price = High;

input length1 = 2;

input length2 = 20;

input averageType1 = AverageType.EXPONENTIAL;

input averageType2 = AverageType.EXPONENTIAL;

input crossingType = {default above, below};

def avg1 = MovingAverage(averageType1, price, length1);

def avg2 = MovingAverage(averageType2, price, length2);

plot signal = crosses(avg1, avg2, crossingType == CrossingType.above);

signal.DefineColor("Above", GetColor(6));

signal.DefineColor("Below", GetColor(7));

signal.AssignValueColor(if crossingType == CrossingType.above then signal.color("Above") else signal.color("Below"));

signal.SetPaintingStrategy(if crossingType == CrossingType.above

then PaintingStrategy.BOOLEAN\_ARROW\_UP

else PaintingStrategy.BOOLEAN\_ARROW\_DOWN);

**#13 Script Parabolic SAR Crossover 3min** *(Last signal crossed under (Bullish))*

#wizard input: crossingType

#wizard text: Inputs: acceleration factor:

#wizard input: accelerationFactor

#wizard text: acceleration limit:

#wizard input: accelerationLimit

input accelerationFactor = 0.01;

input accelerationLimit = 0.1;

input crossingType = {default Bearish, Bullish};

def sar = ParabolicSAR(accelerationFactor=accelerationFactor, accelerationLimit=accelerationLimit);

plot signal = crosses(sar, close, CrossingType == CrossingType.Bearish);

signal.DefineColor("Bullish", GetColor(5));

signal.DefineColor("Bearish", GetColor(6));

signal.AssignValueColor(if crossingType == CrossingType.Bullish then signal.color("Bullish") else signal.color("Bearish"));

signal.SetPaintingStrategy(if crossingType == CrossingType.bullish

then PaintingStrategy.BOOLEAN\_ARROW\_UP

else PaintingStrategy.BOOLEAN\_ARROW\_DOWN);

**#14 Script Parabolic SAR Crossover 5min** *(Last signal crossed under (Bullish))* ***(Settings are different for this study compared to the 3min TF SAR cross)***

#wizard input: crossingType

#wizard text: Inputs: acceleration factor:

#wizard input: accelerationFactor

#wizard text: acceleration limit:

#wizard input: accelerationLimit

input accelerationFactor = 0.001;

input accelerationLimit = 0.01;

input crossingType = {default Bearish, Bullish};

def sar = ParabolicSAR(accelerationFactor=accelerationFactor, accelerationLimit=accelerationLimit);

plot signal = crosses(sar, close, CrossingType == CrossingType.Bearish);

signal.DefineColor("Bullish", GetColor(5));

signal.DefineColor("Bearish", GetColor(6));

signal.AssignValueColor(if crossingType == CrossingType.Bullish then signal.color("Bullish") else signal.color("Bearish"));

signal.SetPaintingStrategy(if crossingType == CrossingType.bullish

then PaintingStrategy.BOOLEAN\_ARROW\_UP

else PaintingStrategy.BOOLEAN\_ARROW\_DOWN);

**OTHERS TO OPTIMIZE/EXPLORE**

**ABTrendProbReversal**

**A screenshot of a computer

AI-generated content may be incorrect.**

declare zerobase;

Declare lower;

input timeframe = AggregationPeriod.MIN;

input colorBars = no;

input showProbabilityLabel = yes;

input EnableProbabilityLevels = yes; # "Enable Probability Levels"

input rsiLength = 20; # "Oscillator Length"

input movAvgType = AverageType.SIMPLE;

input source = FundamentalType.HL2;

input fastLength = 5;

input slowLength = 34;

def na = Double.NaN;

def last = isNaN(Close);

def cap = GetAggregationPeriod();

def tf = Max(cap, timeframe);

def src = Fundamental(source, Period = tf);

#--

DefineGlobalColor("Green1" , CreateColor(0, 255, 187));

DefineGlobalColor("Green2" , CreateColor(0, 216, 158));

DefineGlobalColor("Green3" , CreateColor(0, 177, 129));

DefineGlobalColor("Green4" , CreateColor(0, 137, 101));

DefineGlobalColor("Green5" , CreateColor(0, 98, 72));

DefineGlobalColor("Red1" , CreateColor(255, 17, 0));

DefineGlobalColor("Red2" , CreateColor(216, 14, 0));

DefineGlobalColor("Red3" , CreateColor(177, 12, 0));

DefineGlobalColor("Red4" , CreateColor(137, 9, 0));

DefineGlobalColor("Red5" , CreateColor(98, 7, 0));

DefineGlobalColor("dev1" , CreateColor(209, 209, 255));

DefineGlobalColor("dev2" , CreateColor(189, 189, 255));

DefineGlobalColor("dev3" , CreateColor(170, 170, 255));

DefineGlobalColor("dev4" , CreateColor(130, 130, 255));

DefineGlobalColor("dev5" , CreateColor(111, 111, 255));

#// Amazing Oscillator Calculation

def shortSMA = MovingAverage(movAvgType, src, fastLength);

def longSMA = MovingAverage(movAvgType, src, slowLength);

def amazingOsc = shortSMA - longSMA;

#// RSI-like Calculation

def cRSI = rsi(Price = amazingOsc, Length = rsiLength);

def customRSI = cRSI - 50;

def col = customRSI > 0;

def cross = if (customRSI > 0 and customRSI[1] <= 0) then yes else

if (customRSI < 0 and customRSI[1] >= 0) then yes else no;

def cut = if cross then 0 else cut[1] + 1;

def volatility = stdev(cut, 100) / 3;

def cutCnt;

def cutVal;

def sqrVal;

if (cut == 0 and cut[1]) {

cutCnt = cutCnt[1] + 1;

cutVal = cutVal[1] + cut[1];

sqrVal = sqrVal[1] + Sqr(cut[1]);

} else {

cutCnt = cutCnt[1];

cutVal = cutVal[1];

sqrVal = sqrVal[1];

}

def avg = if cutCnt!=0 then cutVal / cutCnt else na;

def StDv = if cutCnt!=0 then Sqrt((sqrVal / cutCnt) - Sqr(avg)) else na;

def up3 = avg + StDv \* 3;

def up2 = avg + StDv \* 2;

def up1 = avg + StDv \* 1;

def lo1 = avg - StDv \* 1;

def chg1 = cut \* 0.80;

def chg2 = cut \* 0.60;

def chg3 = cut \* 0.40;

def chg4 = cut \* 0.20;

plot upper3 = if !last and EnableProbabilityLevels then up3 else na; # "Upper 3 SD"

plot upper2 = if !last and EnableProbabilityLevels then up2 else na; # "Upper 2 SD"

plot upper1 = if !last and EnableProbabilityLevels then up1 else na;

plot basis = if !last and EnableProbabilityLevels then avg else na;

plot lower1 = if !last and EnableProbabilityLevels then lo1 else na; # "Lower 1 SD"

plot sigDuration5 = if last then na else chg4; # "Signal Duration 20%"

plot sigDuration4 = if last then na else chg3; # "Signal Duration 40%"

plot sigDuration3 = if last then na else chg2; # "Signal Duration 60%"

plot sigDuration2 = if last then na else chg1; # "Signal Duration 80%"

plot sigDuration1 = if last then na else cut; # "Signal Duration"

upper3.SetDefaultColor(GlobalColor("dev1"));

upper2.SetDefaultColor(GlobalColor("dev2"));

upper1.SetDefaultColor(GlobalColor("dev3"));

basis.SetDefaultColor(GlobalColor("dev4"));

lower1.SetDefaultColor(GlobalColor("dev5"));

sigDuration1.SetPaintingStrategy(PaintingStrategy.SQUARED\_HISTOGRAM);

sigDuration2.SetPaintingStrategy(PaintingStrategy.SQUARED\_HISTOGRAM);

sigDuration3.SetPaintingStrategy(PaintingStrategy.SQUARED\_HISTOGRAM);

sigDuration4.SetPaintingStrategy(PaintingStrategy.SQUARED\_HISTOGRAM);

sigDuration5.SetPaintingStrategy(PaintingStrategy.SQUARED\_HISTOGRAM);

sigDuration1.AssignValueColor(if col then GlobalColor("Green5") else GlobalColor("Red5"));

sigDuration2.AssignValueColor(if col then GlobalColor("Green4") else GlobalColor("Red4"));

sigDuration3.AssignValueColor(if col then GlobalColor("Green3") else GlobalColor("Red3"));

sigDuration4.AssignValueColor(if col then GlobalColor("Green2") else GlobalColor("Red2"));

sigDuration5.AssignValueColor(if col then GlobalColor("Green1") else GlobalColor("Red1"));

plot extreme = if cut > (avg + StDv \* 3) then cut + volatility else na; # "Extreme Reversal Probability"

extreme.SetPaintingStrategy(PaintingStrategy.LINE\_VS\_TRIANGLES);

extreme.SetDefaultColor(Color.YELLOW);

#// Function to approximate cumulative probability using error function

Script f\_cdf {

input z = 0;

def a1 = 0.254829592;

def a2 = -0.284496736;

def a3 = 1.421413741;

def a4 = -1.453152027;

def a5 = 1.061405429;

def p = 0.3275911;

def sign = if z < 0 then -1 else 1;

def x = AbsValue(z) / sqrt(2);

def t = 1 / (1 + p \* x);

def erf\_approx = 1 - (((((a5 \* t + a4) \* t) + a3) \* t + a2) \* t + a1) \* t \* exp(-x \* x);

def f\_cdf = 0.5 \* (1 + sign \* erf\_approx);

plot out = if isNaN(f\_cdf) then 0 else Min(Max(f\_cdf, 0), 1);

}

def z = (cut - avg) / StDv;

def probability = f\_cdf(z);

def prop = probability \* 100;

AddLabel(showProbabilityLabel,(if prop > up3 then "Extreme Reversal Probability (" else

if prop > up2 then "Very High Reversal Probability (" else

if prop > up1 then "High Reversal Probability (" else

if prop > avg then "Avg Reversal Probability (" else

if prop > lo1 then "Low Reversal Probability (" else "Very Low Reversal Probability (") +

AsPercent(probability) + ")", CreateColor(prop\*2.55, 255 - prop\*2.55, 255));

#-- Bar Color

##// ~~ Gradient Coloring {

Script gradient\_color {

input src = close;

input minVal = 10;

input maxVal = 400;

input loR = 173;

input loG = 216;

input loB = 230;

input hiR = 41;

input hiG = 98;

input hiB = 255;

def value = if isNaN(src) then 0 else src;

def clamped\_value = max(min(value, maxVal), minVal);

def normalized\_value = (clamped\_value - minVal) / (maxVal - minVal);

def re = floor(loR + (hiR - loR) \* normalized\_value);

def gr = floor(loG + (hiG - loG) \* normalized\_value);

def bl = floor(loB + (hiB - loB) \* normalized\_value);

plot r = re;

plot g = gr;

plot b = bl;

}

def r = gradient\_color(cut, avg - StDv \* 2, up2, 0, 255, 0, 255, 0, 0).r;

AssignPriceColor(if !colorBars then Color.CURRENT else

if col > 0 then CreateColor(0, r, r) else

CreateColor(r, 0, r));

#AssignPriceColor(if !colorBars then Color.CURRENT else CreateColor(255 - cRSI\*2.55, cRSI\*2.55, cRSI\*2.55));

#-- END of CODe

**ABMomentumSMA (open, 9, 9)**

**declare lower;**

**input price = close;**

**input momentumLength = 28;**

**input smaLength = 28;**

**assert(momentumLength > 0, "'momentum length' must be positive: " + momentumLength);**

**plot Momentum = price - price[momentumLength];**

**plot Avg = Average(data = Momentum, length = smaLength);**

**plot ZeroLine = 0;**

**Momentum.SetDefaultColor(GetColor(1));**

**Avg.SetDefaultColor(GetColor(5));**

**ZeroLine.SetDefaultColor(GetColor(0));**

**ABMultiTimeFrameTrend (Exp 5, 21)**

input MAtype = AverageType.EXPONENTIAL;

input ShortAverage = 5;

input LongAverage = 21;

def MA\_month\_close;

def MA\_month\_shortAvg;

def MA\_month\_longAvg;

def MA\_month\_higher;

def MA\_month\_lower;

def MA\_month\_same;

def MA\_month\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.MONTH {

MA\_month\_close = close(period="Month");

MA\_month\_shortAvg = MovingAverage(MAtype, MA\_month\_close, ShortAverage);

MA\_month\_longAvg = MovingAverage(MAtype,MA\_month\_close, LongAverage);

MA\_month\_nowcrossing = if Crosses(MA\_month\_shortAvg, MA\_month\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_month\_same = if MA\_month\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_month\_same == 1 {

MA\_month\_higher = Double.NaN;

MA\_month\_lower = Double.NaN;

} else {

MA\_month\_higher = if MA\_month\_shortAvg >= MA\_month\_longAvg[1] then 1 else Double.NaN;

MA\_month\_lower = if MA\_month\_shortAvg < MA\_month\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_month\_close = 0;

MA\_month\_shortAvg = 0;

MA\_month\_longAvg = 0;

MA\_month\_higher = Double.NaN;

MA\_month\_lower = Double.NaN;

MA\_month\_same = Double.NaN;

MA\_month\_nowcrossing = 0;

}

AddLabel(MA\_month\_higher, "M", Color.DARK\_GREEN);

AddLabel(MA\_month\_lower, "M", Color.DARK\_RED);

AddLabel(MA\_month\_same, "M", Color.WHITE);

def MA\_week\_close;

def MA\_week\_shortAvg;

def MA\_week\_longAvg;

def MA\_week\_higher;

def MA\_week\_lower;

def MA\_week\_same;

def MA\_week\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.WEEK {

MA\_week\_close = close(period="Week");

MA\_week\_shortAvg = MovingAverage(MAtype, MA\_week\_close, ShortAverage);

MA\_week\_longAvg = MovingAverage(MAtype,MA\_week\_close, LongAverage);

MA\_week\_nowcrossing = if Crosses(MA\_week\_shortAvg, MA\_week\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_week\_same = if MA\_week\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_week\_same == 1 {

MA\_week\_higher = Double.NaN;

MA\_week\_lower = Double.NaN;

} else {

MA\_week\_higher = if MA\_week\_shortAvg >= MA\_week\_longAvg[1] then 1 else Double.NaN;

MA\_week\_lower = if MA\_week\_shortAvg < MA\_week\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_week\_close = 0;

MA\_week\_shortAvg = 0;

MA\_week\_longAvg = 0;

MA\_week\_higher = Double.NaN;

MA\_week\_lower = Double.NaN;

MA\_week\_same = Double.NaN;

MA\_week\_nowcrossing = 0;

}

AddLabel(MA\_week\_higher, "W", Color.DARK\_GREEN);

AddLabel(MA\_week\_lower, "W", Color.DARK\_RED);

AddLabel(MA\_week\_same, "W", Color.WHITE);

def MA\_4day\_close;

def MA\_4day\_shortAvg;

def MA\_4day\_longAvg;

def MA\_4day\_higher;

def MA\_4day\_lower;

def MA\_4day\_same;

def MA\_4day\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.FOUR\_DAYS {

MA\_4day\_close = close(period="4 days");

MA\_4day\_shortAvg = MovingAverage(MAtype, MA\_4day\_close, ShortAverage);

MA\_4day\_longAvg = MovingAverage(MAtype,MA\_4day\_close, LongAverage);

MA\_4day\_nowcrossing = if Crosses(MA\_4day\_shortAvg, MA\_4day\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_4day\_same = if MA\_4day\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_4day\_same == 1 {

MA\_4day\_higher = Double.NaN;

MA\_4day\_lower = Double.NaN;

} else {

MA\_4day\_higher = if MA\_4day\_shortAvg >= MA\_4day\_longAvg[1] then 1 else Double.NaN;

MA\_4day\_lower = if MA\_4day\_shortAvg < MA\_4day\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_4day\_close = 0;

MA\_4day\_shortAvg = 0;

MA\_4day\_longAvg = 0;

MA\_4day\_higher = Double.NaN;

MA\_4day\_lower = Double.NaN;

MA\_4day\_same = Double.NaN;

MA\_4day\_nowcrossing = 0;

}

AddLabel(MA\_4day\_higher, "4D", Color.DARK\_GREEN);

AddLabel(MA\_4day\_lower, "4D", Color.DARK\_RED);

AddLabel(MA\_4day\_same, "4D", Color.WHITE);

def MA\_3day\_close;

def MA\_3day\_shortAvg;

def MA\_3day\_longAvg;

def MA\_3day\_higher;

def MA\_3day\_lower;

def MA\_3day\_same;

def MA\_3day\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.THREE\_DAYS {

MA\_3day\_close = close(period="3 days");

MA\_3day\_shortAvg = MovingAverage(MAtype, MA\_3day\_close, ShortAverage);

MA\_3day\_longAvg = MovingAverage(MAtype,MA\_3day\_close, LongAverage);

MA\_3day\_nowcrossing = if Crosses(MA\_3day\_shortAvg, MA\_3day\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_3day\_same = if MA\_3day\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_3day\_same == 1 {

MA\_3day\_higher = Double.NaN;

MA\_3day\_lower = Double.NaN;

} else {

MA\_3day\_higher = if MA\_3day\_shortAvg >= MA\_3day\_longAvg[1] then 1 else Double.NaN;

MA\_3day\_lower = if MA\_3day\_shortAvg < MA\_3day\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_3day\_close = 0;

MA\_3day\_shortAvg = 0;

MA\_3day\_longAvg = 0;

MA\_3day\_higher = Double.NaN;

MA\_3day\_lower = Double.NaN;

MA\_3day\_same = Double.NaN;

MA\_3day\_nowcrossing = 0;

}

AddLabel(MA\_3day\_higher, "3D", Color.DARK\_GREEN);

AddLabel(MA\_3day\_lower, "3D", Color.DARK\_RED);

AddLabel(MA\_3day\_same, "3D", Color.WHITE);

def MA\_2day\_close;

def MA\_2day\_shortAvg;

def MA\_2day\_longAvg;

def MA\_2day\_higher;

def MA\_2day\_lower;

def MA\_2day\_same;

def MA\_2day\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.TWO\_DAYS {

MA\_2day\_close = close(period="2 days");

MA\_2day\_shortAvg = MovingAverage(MAtype, MA\_2day\_close, ShortAverage);

MA\_2day\_longAvg = MovingAverage(MAtype,MA\_2day\_close, LongAverage);

MA\_2day\_nowcrossing = if Crosses(MA\_2day\_shortAvg, MA\_2day\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_2day\_same = if MA\_2day\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_2day\_same == 1 {

MA\_2day\_higher = Double.NaN;

MA\_2day\_lower = Double.NaN;

} else {

MA\_2day\_higher = if MA\_2day\_shortAvg >= MA\_2day\_longAvg[1] then 1 else Double.NaN;

MA\_2day\_lower = if MA\_2day\_shortAvg < MA\_2day\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_2day\_close = 0;

MA\_2day\_shortAvg = 0;

MA\_2day\_longAvg = 0;

MA\_2day\_higher = Double.NaN;

MA\_2day\_lower = Double.NaN;

MA\_2day\_same = Double.NaN;

MA\_2day\_nowcrossing = 0;

}

AddLabel(MA\_2day\_higher, "2D", Color.DARK\_GREEN);

AddLabel(MA\_2day\_lower, "2D", Color.DARK\_RED);

AddLabel(MA\_2day\_same, "2D", Color.WHITE);

def MA\_1day\_close;

def MA\_1day\_shortAvg;

def MA\_1day\_longAvg;

def MA\_1day\_higher;

def MA\_1day\_lower;

def MA\_1day\_same;

def MA\_1day\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.DAY {

MA\_1day\_close = close(period="Day");

MA\_1day\_shortAvg = MovingAverage(MAtype, MA\_1day\_close, ShortAverage);

MA\_1day\_longAvg = MovingAverage(MAtype,MA\_1day\_close, LongAverage);

MA\_1day\_nowcrossing = if Crosses(MA\_1day\_shortAvg, MA\_1day\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_1day\_same = if MA\_1day\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_1day\_same == 1 {

MA\_1day\_higher = Double.NaN;

MA\_1day\_lower = Double.NaN;

} else {

MA\_1day\_higher = if MA\_1day\_shortAvg >= MA\_1day\_longAvg[1] then 1 else Double.NaN;

MA\_1day\_lower = if MA\_1day\_shortAvg < MA\_1day\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_1day\_close = 0;

MA\_1day\_shortAvg = 0;

MA\_1day\_longAvg = 0;

MA\_1day\_higher = Double.NaN;

MA\_1day\_lower = Double.NaN;

MA\_1day\_same = Double.NaN;

MA\_1day\_nowcrossing = 0;

}

AddLabel(MA\_1day\_higher, "1D", Color.DARK\_GREEN);

AddLabel(MA\_1day\_lower, "1D", Color.DARK\_RED);

AddLabel(MA\_1day\_same, "1D", Color.WHITE);

def MA\_4hour\_close;

def MA\_4hour\_shortAvg;

def MA\_4hour\_longAvg;

def MA\_4hour\_higher;

def MA\_4hour\_lower;

def MA\_4hour\_same;

def MA\_4hour\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.FOUR\_HOURS {

MA\_4hour\_close = close(period="4 hours");

MA\_4hour\_shortAvg = MovingAverage(MAtype, MA\_4hour\_close, ShortAverage);

MA\_4hour\_longAvg = MovingAverage(MAtype,MA\_4hour\_close, LongAverage);

MA\_4hour\_nowcrossing = if Crosses(MA\_4hour\_shortAvg, MA\_4hour\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_4hour\_same = if MA\_4hour\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_4hour\_same == 1 {

MA\_4hour\_higher = Double.NaN;

MA\_4hour\_lower = Double.NaN;

} else {

MA\_4hour\_higher = if MA\_4hour\_shortAvg >= MA\_4hour\_longAvg[1] then 1 else Double.NaN;

MA\_4hour\_lower = if MA\_4hour\_shortAvg < MA\_4hour\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_4hour\_close = 0;

MA\_4hour\_shortAvg = 0;

MA\_4hour\_longAvg = 0;

MA\_4hour\_higher = Double.NaN;

MA\_4hour\_lower = Double.NaN;

MA\_4hour\_same = Double.NaN;

MA\_4hour\_nowcrossing = 0;

}

AddLabel(MA\_4hour\_higher, "4h", Color.DARK\_GREEN);

AddLabel(MA\_4hour\_lower, "4h", Color.DARK\_RED);

AddLabel(MA\_4hour\_same, "4h", Color.WHITE);

def MA\_2hour\_close;

def MA\_2hour\_shortAvg;

def MA\_2hour\_longAvg;

def MA\_2hour\_higher;

def MA\_2hour\_lower;

def MA\_2hour\_same;

def MA\_2hour\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.TWO\_HOURS {

MA\_2hour\_close = close(period="2 hours");

MA\_2hour\_shortAvg = MovingAverage(MAtype, MA\_2hour\_close, ShortAverage);

MA\_2hour\_longAvg = MovingAverage(MAtype,MA\_2hour\_close, LongAverage);

MA\_2hour\_nowcrossing = if Crosses(MA\_2hour\_shortAvg, MA\_2hour\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_2hour\_same = if MA\_2hour\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_2hour\_same == 1 {

MA\_2hour\_higher = Double.NaN;

MA\_2hour\_lower = Double.NaN;

} else {

MA\_2hour\_higher = if MA\_2hour\_shortAvg >= MA\_2hour\_longAvg[1] then 1 else Double.NaN;

MA\_2hour\_lower = if MA\_2hour\_shortAvg < MA\_2hour\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_2hour\_close = 0;

MA\_2hour\_shortAvg = 0;

MA\_2hour\_longAvg = 0;

MA\_2hour\_higher = Double.NaN;

MA\_2hour\_lower = Double.NaN;

MA\_2hour\_same = Double.NaN;

MA\_2hour\_nowcrossing = 0;

}

AddLabel(MA\_2hour\_higher, "2h", Color.DARK\_GREEN);

AddLabel(MA\_2hour\_lower, "2h", Color.DARK\_RED);

AddLabel(MA\_2hour\_same, "2h", Color.WHITE);

def MA\_1hour\_close;

def MA\_1hour\_shortAvg;

def MA\_1hour\_longAvg;

def MA\_1hour\_higher;

def MA\_1hour\_lower;

def MA\_1hour\_same;

def MA\_1hour\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.HOUR {

MA\_1hour\_close = close(period="1 hour");

MA\_1hour\_shortAvg = MovingAverage(MAtype, MA\_1hour\_close, ShortAverage);

MA\_1hour\_longAvg = MovingAverage(MAtype,MA\_1hour\_close, LongAverage);

MA\_1hour\_nowcrossing = if Crosses(MA\_1hour\_shortAvg, MA\_1hour\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_1hour\_same = if MA\_1hour\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_1hour\_same == 1 {

MA\_1hour\_higher = Double.NaN;

MA\_1hour\_lower = Double.NaN;

} else {

MA\_1hour\_higher = if MA\_1hour\_shortAvg >= MA\_1hour\_longAvg[1] then 1 else Double.NaN;

MA\_1hour\_lower = if MA\_1hour\_shortAvg < MA\_1hour\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_1hour\_close = 0;

MA\_1hour\_shortAvg = 0;

MA\_1hour\_longAvg = 0;

MA\_1hour\_higher = Double.NaN;

MA\_1hour\_lower = Double.NaN;

MA\_1hour\_same = Double.NaN;

MA\_1hour\_nowcrossing = 0;

}

AddLabel(MA\_1hour\_higher, "1h", Color.DARK\_GREEN);

AddLabel(MA\_1hour\_lower, "1h", Color.DARK\_RED);

AddLabel(MA\_1hour\_same, "1h", Color.WHITE);

def MA\_30min\_close;

def MA\_30min\_shortAvg;

def MA\_30min\_longAvg;

def MA\_30min\_higher;

def MA\_30min\_lower;

def MA\_30min\_same;

def MA\_30min\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.THIRTY\_MIN {

MA\_30min\_close = close(period="30 min");

MA\_30min\_shortAvg = MovingAverage(MAtype, MA\_30min\_close, ShortAverage);

MA\_30min\_longAvg = MovingAverage(MAtype,MA\_30min\_close, LongAverage);

MA\_30min\_nowcrossing = if Crosses(MA\_30min\_shortAvg, MA\_30min\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_30min\_same = if MA\_30min\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_30min\_same == 1 {

MA\_30min\_higher = Double.NaN;

MA\_30min\_lower = Double.NaN;

} else {

MA\_30min\_higher = if MA\_30min\_shortAvg >= MA\_30min\_longAvg[1] then 1 else Double.NaN;

MA\_30min\_lower = if MA\_30min\_shortAvg < MA\_30min\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_30min\_close = 0;

MA\_30min\_shortAvg = 0;

MA\_30min\_longAvg = 0;

MA\_30min\_higher = Double.NaN;

MA\_30min\_lower = Double.NaN;

MA\_30min\_same = Double.NaN;

MA\_30min\_nowcrossing = 0;

}

AddLabel(MA\_30min\_higher, "30m", Color.DARK\_GREEN);

AddLabel(MA\_30min\_lower, "30m", Color.DARK\_RED);

AddLabel(MA\_30min\_same, "30m", Color.WHITE);

def MA\_15min\_close;

def MA\_15min\_shortAvg;

def MA\_15min\_longAvg;

def MA\_15min\_higher;

def MA\_15min\_lower;

def MA\_15min\_same;

def MA\_15min\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.FIFTEEN\_MIN {

MA\_15min\_close = close(period="15 min");

MA\_15min\_shortAvg = MovingAverage(MAtype, MA\_15min\_close, ShortAverage);

MA\_15min\_longAvg = MovingAverage(MAtype,MA\_15min\_close, LongAverage);

MA\_15min\_nowcrossing = if Crosses(MA\_15min\_shortAvg, MA\_15min\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_15min\_same = if MA\_15min\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_15min\_same == 1 {

MA\_15min\_higher = Double.NaN;

MA\_15min\_lower = Double.NaN;

} else {

MA\_15min\_higher = if MA\_15min\_shortAvg >= MA\_15min\_longAvg[1] then 1 else Double.NaN;

MA\_15min\_lower = if MA\_15min\_shortAvg < MA\_15min\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_15min\_close = 0;

MA\_15min\_shortAvg = 0;

MA\_15min\_longAvg = 0;

MA\_15min\_higher = Double.NaN;

MA\_15min\_lower = Double.NaN;

MA\_15min\_same = Double.NaN;

MA\_15min\_nowcrossing = 0;

}

AddLabel(MA\_15min\_higher, "15m", Color.DARK\_GREEN);

AddLabel(MA\_15min\_lower, "15m", Color.DARK\_RED);

AddLabel(MA\_15min\_same, "15m", Color.WHITE);

def MA\_10min\_close;

def MA\_10min\_shortAvg;

def MA\_10min\_longAvg;

def MA\_10min\_higher;

def MA\_10min\_lower;

def MA\_10min\_same;

def MA\_10min\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.TEN\_MIN {

MA\_10min\_close = close(period="10 min");

MA\_10min\_shortAvg = MovingAverage(MAtype, MA\_10min\_close, ShortAverage);

MA\_10min\_longAvg = MovingAverage(MAtype,MA\_10min\_close, LongAverage);

MA\_10min\_nowcrossing = if Crosses(MA\_10min\_shortAvg, MA\_10min\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_10min\_same = if MA\_10min\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_10min\_same == 1 {

MA\_10min\_higher = Double.NaN;

MA\_10min\_lower = Double.NaN;

} else {

MA\_10min\_higher = if MA\_10min\_shortAvg >= MA\_10min\_longAvg[1] then 1 else Double.NaN;

MA\_10min\_lower = if MA\_10min\_shortAvg < MA\_10min\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_10min\_close = 0;

MA\_10min\_shortAvg = 0;

MA\_10min\_longAvg = 0;

MA\_10min\_higher = Double.NaN;

MA\_10min\_lower = Double.NaN;

MA\_10min\_same = Double.NaN;

MA\_10min\_nowcrossing = 0;

}

AddLabel(MA\_10min\_higher, "10m", Color.DARK\_GREEN);

AddLabel(MA\_10min\_lower, "10m", Color.DARK\_RED);

AddLabel(MA\_10min\_same, "10m", Color.WHITE);

def MA\_shortAvgmin\_close;

def MA\_shortAvgmin\_shortAvg;

def MA\_shortAvgmin\_longAvg;

def MA\_shortAvgmin\_higher;

def MA\_shortAvgmin\_lower;

def MA\_shortAvgmin\_same;

def MA\_shortAvgmin\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.FIVE\_MIN {

MA\_shortAvgmin\_close = close(period="5 min");

MA\_shortAvgmin\_shortAvg = MovingAverage(MAtype, MA\_shortAvgmin\_close, ShortAverage);

MA\_shortAvgmin\_longAvg = MovingAverage(MAtype,MA\_shortAvgmin\_close, LongAverage);

MA\_shortAvgmin\_nowcrossing = if Crosses(MA\_shortAvgmin\_shortAvg, MA\_shortAvgmin\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_shortAvgmin\_same = if MA\_shortAvgmin\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_shortAvgmin\_same == 1 {

MA\_shortAvgmin\_higher = Double.NaN;

MA\_shortAvgmin\_lower = Double.NaN;

} else {

MA\_shortAvgmin\_higher = if MA\_shortAvgmin\_shortAvg >= MA\_shortAvgmin\_longAvg[1] then 1 else Double.NaN;

MA\_shortAvgmin\_lower = if MA\_shortAvgmin\_shortAvg < MA\_shortAvgmin\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_shortAvgmin\_close = 0;

MA\_shortAvgmin\_shortAvg = 0;

MA\_shortAvgmin\_longAvg = 0;

MA\_shortAvgmin\_higher = Double.NaN;

MA\_shortAvgmin\_lower = Double.NaN;

MA\_shortAvgmin\_same = Double.NaN;

MA\_shortAvgmin\_nowcrossing = 0;

}

AddLabel(MA\_shortAvgmin\_higher, "5m", Color.DARK\_GREEN);

AddLabel(MA\_shortAvgmin\_lower, "5m", Color.DARK\_RED);

AddLabel(MA\_shortAvgmin\_same, "5m", Color.WHITE);

def MA\_4min\_close;

def MA\_4min\_shortAvg;

def MA\_4min\_longAvg;

def MA\_4min\_higher;

def MA\_4min\_lower;

def MA\_4min\_same;

def MA\_4min\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.FOUR\_MIN {

MA\_4min\_close = close(period="4 min");

MA\_4min\_shortAvg = MovingAverage(MAtype, MA\_4min\_close, ShortAverage);

MA\_4min\_longAvg = MovingAverage(MAtype,MA\_4min\_close, LongAverage);

MA\_4min\_nowcrossing = if Crosses(MA\_4min\_shortAvg, MA\_4min\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_4min\_same = if MA\_4min\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_4min\_same == 1 {

MA\_4min\_higher = Double.NaN;

MA\_4min\_lower = Double.NaN;

} else {

MA\_4min\_higher = if MA\_4min\_shortAvg >= MA\_4min\_longAvg[1] then 1 else Double.NaN;

MA\_4min\_lower = if MA\_4min\_shortAvg < MA\_4min\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_4min\_close = 0;

MA\_4min\_shortAvg = 0;

MA\_4min\_longAvg = 0;

MA\_4min\_higher = Double.NaN;

MA\_4min\_lower = Double.NaN;

MA\_4min\_same = Double.NaN;

MA\_4min\_nowcrossing = 0;

}

AddLabel(MA\_4min\_higher, "4m", Color.DARK\_GREEN);

AddLabel(MA\_4min\_lower, "4m", Color.DARK\_RED);

AddLabel(MA\_4min\_same, "4m", Color.WHITE);

def MA\_3min\_close;

def MA\_3min\_shortAvg;

def MA\_3min\_longAvg;

def MA\_3min\_higher;

def MA\_3min\_lower;

def MA\_3min\_same;

def MA\_3min\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.THREE\_MIN {

MA\_3min\_close = close(period="3 min");

MA\_3min\_shortAvg = MovingAverage(MAtype, MA\_3min\_close, ShortAverage);

MA\_3min\_longAvg = MovingAverage(MAtype,MA\_3min\_close, LongAverage);

MA\_3min\_nowcrossing = if Crosses(MA\_3min\_shortAvg, MA\_3min\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_3min\_same = if MA\_3min\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_3min\_same == 1 {

MA\_3min\_higher = Double.NaN;

MA\_3min\_lower = Double.NaN;

} else {

MA\_3min\_higher = if MA\_3min\_shortAvg >= MA\_3min\_longAvg[1] then 1 else Double.NaN;

MA\_3min\_lower = if MA\_3min\_shortAvg < MA\_3min\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_3min\_close = 0;

MA\_3min\_shortAvg = 0;

MA\_3min\_longAvg = 0;

MA\_3min\_higher = Double.NaN;

MA\_3min\_lower = Double.NaN;

MA\_3min\_same = Double.NaN;

MA\_3min\_nowcrossing = 0;

}

AddLabel(MA\_3min\_higher, "3m", Color.DARK\_GREEN);

AddLabel(MA\_3min\_lower, "3m", Color.DARK\_RED);

AddLabel(MA\_3min\_same, "3m", Color.WHITE);

def MA\_2min\_close;

def MA\_2min\_shortAvg;

def MA\_2min\_longAvg;

def MA\_2min\_higher;

def MA\_2min\_lower;

def MA\_2min\_same;

def MA\_2min\_nowcrossing;

if GetAggregationPeriod() <= AggregationPeriod.TWO\_MIN {

MA\_2min\_close = close(period="2 min");

MA\_2min\_shortAvg = MovingAverage(MAtype, MA\_2min\_close, ShortAverage);

MA\_2min\_longAvg = MovingAverage(MAtype,MA\_2min\_close, LongAverage);

MA\_2min\_nowcrossing = if Crosses(MA\_2min\_shortAvg, MA\_2min\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_2min\_same = if MA\_2min\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_2min\_same == 1 {

MA\_2min\_higher = Double.NaN;

MA\_2min\_lower = Double.NaN;

} else {

MA\_2min\_higher = if MA\_2min\_shortAvg >= MA\_2min\_longAvg[1] then 1 else Double.NaN;

MA\_2min\_lower = if MA\_2min\_shortAvg < MA\_2min\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_2min\_close = 0;

MA\_2min\_shortAvg = 0;

MA\_2min\_longAvg = 0;

MA\_2min\_higher = Double.NaN;

MA\_2min\_lower = Double.NaN;

MA\_2min\_same = Double.NaN;

MA\_2min\_nowcrossing = 0;

}

AddLabel(MA\_2min\_higher, "2m", Color.DARK\_GREEN);

AddLabel(MA\_2min\_lower, "2m", Color.DARK\_RED);

AddLabel(MA\_2min\_same, "2m", Color.WHITE);

def MA\_1min\_close;

def MA\_1min\_shortAvg;

def MA\_1min\_longAvg;

def MA\_1min\_higher;

def MA\_1min\_lower;

def MA\_1min\_same;

def MA\_1min\_nowcrossing;

if GetAggregationPeriod() == AggregationPeriod.MIN {

MA\_1min\_close = close(period="1 min");

MA\_1min\_shortAvg = MovingAverage(MAtype, MA\_1min\_close, ShortAverage);

MA\_1min\_longAvg = MovingAverage(MAtype,MA\_1min\_close, LongAverage);

MA\_1min\_nowcrossing = if Crosses(MA\_1min\_shortAvg, MA\_1min\_longAvg, CrossingDirection.ANY) then 1 else 0;

MA\_1min\_same = if MA\_1min\_nowcrossing == 1 then 1 else Double.NaN;

if MA\_1min\_same == 1 {

MA\_1min\_higher = Double.NaN;

MA\_1min\_lower = Double.NaN;

} else {

MA\_1min\_higher = if MA\_1min\_shortAvg >= MA\_1min\_longAvg[1] then 1 else Double.NaN;

MA\_1min\_lower = if MA\_1min\_shortAvg < MA\_1min\_longAvg[1] then 1 else Double.NaN;

}

} else {

MA\_1min\_close = 0;

MA\_1min\_shortAvg = 0;

MA\_1min\_longAvg = 0;

MA\_1min\_higher = Double.NaN;

MA\_1min\_lower = Double.NaN;

MA\_1min\_same = Double.NaN;

MA\_1min\_nowcrossing = 0;

}

AddLabel(MA\_1min\_higher, "1m", Color.DARK\_GREEN);

AddLabel(MA\_1min\_lower, "1m", Color.DARK\_RED);

AddLabel(MA\_1min\_same, "1m", Color.WHITE);

**ABStochMFI** (lines, yes, yes, no, close, 4, 1, 90, 200, yes, no, no)

declare lower;

#// Calculate the Stochastic RSI

input displayOptions = {"Heikin-Ashi",Default "Lines"};

input ColorBar = yes;

input useLog = yes; # title="Log")

input UseAverageKAndD = no; # title="Use Average of both K & D")

input source = close;#, title="Source")

input smoothK = 3;#, minval=1, title="Stochastic %K Smoothing")

input smoothD = 3;#, minval=1, title="Stochastic %D Smoothing")

input lengthRsi = 14;#, minval=1, title="RSI Length")

input lengthStoch = 14;#, minval=1, title="Stochastic Length")

input showRegularDivergences = yes; # title="Show Divergences")

input ShowHiddenDivergences = no; #, title="Show Hidden Divergences")

input ShowDivergencesChannel = no; # title="Show Divergences Channel")

def na = Double.NaN;

def last = IsNaN(close);

def ha = displayOptions == displayOptions."Heikin-Ashi";

def n = n[1] + 1;

def bar = if !last then n else bar[1];

def src = if useLog then Log(source) else source;

#--- Color -----

DefineGlobalColor("up" , CreateColor(0, 127, 255));

DefineGlobalColor("dn" , CreateColor(255, 127, 0));

DefineGlobalColor("upBtm" , CreateColor(137, 196, 255));

DefineGlobalColor("dnTop" , CreateColor(255, 196, 137));

# stoch(source, high, low, length) =>

script stoch {

input src = close;

input h = high;

input l = low;

input len = 14;

def hh = Highest(h, len);

def ll = Lowest(l, len);

def stoch = 100 \* (src - ll) / (hh - ll);

plot return = stoch;

}

def rsi1 = RSI(PRICE = src, LENGTH = lengthRSI);

def sto = stoch(rsi1, rsi1, rsi1, lengthStoch);

def kk = Average(sto, smoothK);

def d = Average(kk, smoothD);

def k = if UseAverageKAndD then (kk + d) / 2 else kk;

#// Calculate the Stochastic Heikin-Ashi candles

def cl = k;

def op = cl[1];

def hi = Max(cl, cl[1]);

def lo = Min(cl, cl[1]);

def ha\_close = if ha then (hi + lo + cl + op) / 4 else na; #(kk + d) / 2 else na;

def haOpen = CompoundValue(1, (haOpen[1] + ha\_close[1]) / 2, (op + cl) / 2);

def ha\_open = if haOpen then haOpen else na;

def ha\_high = Max(hi, Max(ha\_open, ha\_close));

def ha\_low = Min(lo, Min(ha\_close, ha\_open));

def price = (ha\_open + ha\_close + ha\_high + ha\_low) / 4;

def col = if ha\_close >= ha\_open then if ha\_close[1] >= ha\_open[1] then 2 else 1 else

if ha\_close[1] < ha\_open[1] then -2 else -1;

#// Plot the Heikin-Ashi Stochastic RSI

AddChart(high = if col == 2 then ha\_high else na, low = ha\_low , open = ha\_close, close = ha\_open,

type = ChartType.CANDLE, growcolor = GlobalColor("up"));

AddChart(high = if col == 1 then ha\_high else na, low = ha\_low , open = ha\_close, close = ha\_open,

type = ChartType.CANDLE, growcolor = GlobalColor("upBtm"));

AddChart(high = if col == -2 then ha\_high else na, low = ha\_low , open = ha\_open, close = ha\_close,

type = ChartType.CANDLE, growcolor = GlobalColor("dn"));

AddChart(high = if col == -1 then ha\_high else na, low = ha\_low , open = ha\_open, close = ha\_close,

type = ChartType.CANDLE, growcolor = GlobalColor("dnTop"));

#// Plot

plot kLine = if !ha and !last then kk else na;

plot dLine = if !ha and !last then d else na;

kLine.SetDefaultColor(GlobalColor("up"));

dLine.SetDefaultColor(GlobalColor("dn"));

AddCloud(kLine, dLine, GlobalColor("up"), GlobalColor("dn"));

#// Use the up and down colors in your script

plot mindline = if last then na else 50;

plot lineUp = if last then na else 80;

plot lineDn = if last then na else 20;

mindline.SetStyle(Curve.SHORT\_DASH);

lineUp.SetPaintingStrategy(PaintingStrategy.DASHES);

lineDn.SetPaintingStrategy(PaintingStrategy.DASHES);

mindline.SetDefaultColor(Color.DARK\_GRAY);

lineUp.SetDefaultColor(Color.DARK\_GRAY);

lineDn.SetDefaultColor(Color.DARK\_GRAY);

AddCloud(lineUp, lineDn, Color.DARK\_GRAY);

AssignPriceColor( if !ColorBar then Color.CURRENT else

if col > 0 and price > 80 then Color.GREEN else

if col > 0 then Color.DARK\_GREEN else

if col < 0 and price < 20 then GlobalColor("dn") else GlobalColor("dnTop"));

#//@RicardoSantos' Divergence Script

def f\_top\_fractal = k[4] < k[2] and k[3] < k[2] and k[2] > k[1] and k[2] > k[0];

def f\_bot\_fractal = k[4] > k[2] and k[3] > k[2] and k[2] < k[1] and k[2] < k[0];

def f\_fractalize = if f\_top\_fractal then 1 else if f\_bot\_fractal then -1 else 0;

#//-------------------------

def fractal\_top = if f\_fractalize > 0 then k[2] else 0;

def fractal\_bot = if f\_fractalize < 0 then k[2] else 0;

def highBarPrev = if fractal\_top then bar[2] else highBarPrev[1];

def high\_prev1 = if fractal\_top then k[2] else high\_prev1[1];

def high\_price1 = if fractal\_top then high[2] else high\_price1[1];

def lowBarPrev = if fractal\_bot then bar[2] else lowBarPrev[1];

def low\_prev1 = if fractal\_bot then k[2] else low\_prev1[1];

def low\_price1 = if fractal\_bot then low[2] else low\_price1[1];

def high\_prev = high\_prev1[2];

def high\_price = high\_price1[2];

def low\_prev = low\_prev1[2];

def low\_price = low\_price1[2];

def regular\_bearish\_div = showRegularDivergences and fractal\_top and high[2] > high\_price and k[2] < high\_prev;

def hidden\_bearish\_div = ShowHiddenDivergences and fractal\_top and high[2] < high\_price and k[2] > high\_prev;

def regular\_bullish\_div = showRegularDivergences and fractal\_bot and low[2] < low\_price and k[2] > low\_prev;

def hidden\_bullish\_div = ShowHiddenDivergences and fractal\_bot and low[2] > low\_price and k[2] < low\_prev;

def regBarBear = if regular\_bearish\_div[-2] then highBarPrev[1] else 0;

def lastRegBear = if regular\_bearish\_div[-2] then bar else 0;

def regBarBull = if regular\_bullish\_div[-2] then lowBarPrev[1] else 0;

def lastRegBull = if regular\_bullish\_div[-2] then bar else 0;

def hidBarBear = if hidden\_bearish\_div[-2] then highBarPrev[1] else 0;

def lasthidBear = if hidden\_bearish\_div[-2] then bar else 0;

def hidBarBull = if hidden\_bullish\_div[-2] then lowBarPrev[1] else 0;

def lasthidBull = if hidden\_bullish\_div[-2] then bar else 0;

#//-------------------------

plot topFract = if ShowDivergencesChannel and fractal\_top[-2] then fractal\_top[-2] else na;

plot botFract = if ShowDivergencesChannel and fractal\_bot[-2] then fractal\_bot[-2] else na;

plot topDiv = if ShowDivergencesChannel and fractal\_top[-2] then fractal\_top[-2] else na; # "Top Div Channel"

plot botDiv = if ShowDivergencesChannel and fractal\_bot[-2] then fractal\_bot[-2] else na; # "Bottom Div Channel"

topDiv.EnableApproximation();

botDiv.EnableApproximation();

topDiv.SetDefaultColor(Color.GRAY);

botDiv.SetDefaultColor(Color.GRAY);

topFract.SetDefaultColor(Color.GRAY);

botFract.SetDefaultColor(Color.GRAY);

topFract.SetPaintingStrategy(3);

botFract.SetPaintingStrategy(2);

AddChartBubble(regular\_bearish\_div[-2], k, "R", Color.RED);

AddChartBubble(regular\_bullish\_div[-2], k, "R", Color.GREEN, no);

AddChartBubble(hidden\_bearish\_div[-2], k, "H", Color.DARK\_RED);

AddChartBubble(hidden\_bullish\_div[-2], k, "H", Color.DARK\_GREEN, no);

plot regBearLine = if bar == HighestAll(regBarBear) then k else

if bar == HighestAll(lastRegBear) then k else na;

plot regBullLine = if bar == HighestAll(regBarBull) then k else

if bar == HighestAll(lastRegBull) then k else na;

plot hidBearLine = if bar == HighestAll(hidBarBear) then k else

if bar == HighestAll(lasthidBear) then k else na;

plot hidBullLine = if bar == HighestAll(hidBarBull) then k else

if bar == HighestAll(lasthidBull) then k else na;

regBearLine.EnableApproximation();

regBullLine.EnableApproximation();

hidBearLine.EnableApproximation();

hidBullLine.EnableApproximation();

regBearLine.SetDefaultColor(Color.RED);

regBullLine.SetDefaultColor(Color.GREEN);

hidBearLine.SetDefaultColor(Color.DARK\_RED);

hidBullLine.SetDefaultColor(Color.DARK\_GREEN);

#---- END CODE

**ABMomentum**

#Momentum Indicator

input length = 13;

input AvgMomentumLength = 13;

input price = close;

input AverageType = AverageType.EXPONENTIAL;

input showBreakoutSignals = no;

assert(length > 0, "'length' must be positive: " + length);

def Momentum = price - price[length];

plot AvgMomentum = movavgExponential(Momentum, AvgMomentumLength);

def bullishMomentum = AvgMomentum > 0;

def bearishMomentum = AvgMomentum < 0;

AvgMomentum.SetPaintingStrategy(paintingstrategy.LINE);

AvgMomentum.SetLineWeight(3);

AvgMomentum.AssignValueColor(if bullishmomentum then color.blue else if bearishMomentum then color.red else color.black);

# Squeeze Checklist Signals

def EMA8 = movavgExponential(close,8);

def EMA21 = movavgExponential(close,21);

def bullishStack = EMA8 > EMA21;

def bearishStack = EMA8 < EMA21;

#RSI

def RSIlength = 14;

def RSIover\_Bought = 70;

def RSIover\_Sold = 30;

def RSIprice = close;

def RSIaverageType = AverageType.WILDERS;

def NetChgAvg = MovingAverage(RSIaverageType, RSIprice - RSIprice[1], RSIlength);

def TotChgAvg = MovingAverage(RSIaverageType, AbsValue(RSIprice - RSIprice[1]), RSIlength);

def ChgRatio = if TotChgAvg != 0 then NetChgAvg / TotChgAvg else 0;

def RSI = 50 \* (ChgRatio + 1);

def RSIOverSold = RSIover\_Sold;

def RSIOverBought = RSIover\_Bought;

def RSIUpSignal = if RSI crosses above RSIOverSold then RSIOverSold else Double.NaN;

def RSIDownSignal = if RSI crosses below RSIOverBought then RSIOverBought else Double.NaN;

def RSImidline = 50;

def bullishRSI = RSI >= 50;

def bearishRSI = RSI < 50;

#Stochastics

def over\_bought = 80;

def over\_sold = 20;

def KPeriod = 14;

def DPeriod = 3;

def priceH = high;

def priceL = low;

def priceC = close;

def slowing\_period = 3;

def averageTypeStoch = AverageType.SIMPLE;

def lowest\_k = Lowest(priceL, KPeriod);

def c1 = priceC - lowest\_k;

def c2 = Highest(priceH, KPeriod) - lowest\_k;

def FastK = if c2 != 0 then c1 / c2 \* 100 else 0;

def FullK = MovingAverage(averageTypeStoch, FastK, slowing\_period);

def FullD = MovingAverage(averageTypeStoch, FullK, DPeriod);

def OverBought = over\_bought;

def OverSold = over\_sold;

def MidLine = 50;

def upK = FullK crosses above OverSold;

def upD = FullD crosses above OverSold;

def downK = FullK crosses below OverBought;

def downD = FullD crosses below OverBought;

def bullishStoch = FullK > FullD;

def bearishStoch = FullK < FullD;

# SqueezeHistogram

def squeezehistogram = TTM\_Squeeze(close, 20, 1.5, 2.0, 1.0).Histogram;

def squeezehistogramUP = squeezehistogram > squeezehistogram[1] and squeezehistogram[1] > squeezehistogram[2];

def squeezehistogramDOWN = squeezehistogram < squeezehistogram[1] and squeezehistogram[1] < squeezehistogram[2];

# Squeeze Signal

def normalSqueeze = if TTM\_Squeeze(close, 20, 1.5, 2.0, 1.0).SqueezeAlert == 0 then 1 else 0;

def aggroSqueeze = if TTM\_Squeeze(close, 20, 2.0, 2.0, 1.0).SqueezeAlert == 0 then 1 else 0;

plot squeezeSignal = normalSqueeze or aggroSqueeze;

squeezeSignal.setPaintingStrategy(paintingStrategy.POINTS);

squeezeSignal.setLineWeight(3);

squeezeSignal.AssignValueColor(if normalSqueeze then color.yellow else if aggroSqueeze then color.yellow else color.dark\_gray);

def BullishSignal = squeezeSignal and bullishstack and bullishRSI and bullishStoch and squeezehistogramUP;

def BearishSignal = squeezeSignal and bearishstack and bearishRSI and bearishStoch and squeezehistogramDOWN;

plot BullishSignal2 = if bullishsignal and !bullishsignal[1] then avgmomentum else double.nan;

plot BearishSignal2 = if bearishsignal and !bearishsignal[1] then avgmomentum else double.nan;

BullishSignal2.SetpaintingStrategy(PaintingStrategy.ARROW\_UP);

BearishSignal2.SetpaintingStrategy(PaintingStrategy.ARROW\_DOWN);

BullishSignal2.SetLineWeight(2);

BearishSignal2.SetlineWeight(2);

BullishSignal2.SetdefaultColor(color.green);

BearishSignal2.SetdefaultColor(color.red);

# End Code

**ABDynamicSR**

input labelOptions = {"Left", "Right", Default "Center", "Don't Show"};

input dynamic = yes; # "Use Dynamic Look-Back"

input lookback = 200;

input Resolution = 40; #, "Resolution"

input ZoneTolerance = 0.93; #,"Zone Tolerance",toolti

def na = Double.NaN;

def last = isNaN(close);

def n = if BarNumber() < 1 then 1 else BarNumber();

def bar = if !last then n else Bar[1];

def res = Min(Max(Resolution, 1), 50);

def zone = Min(Max(ZoneTolerance, 0.01), 1);

Script Pivot {

input series = close;

input leftBars = 10;

input rightBars = 10;

input isHigh = yes;

def na = Double.NaN;

def HH = series == Highest(series, leftBars + 1);

def LL = series == Lowest(series, leftBars + 1);

def pivotRange = (leftBars + rightBars + 1);

def leftEdgeValue = if series[pivotRange] ==0 then na else series[pivotRange];

def pvtCond = !isNaN(series) and leftBars > 0 and rightBars > 0 and !isNaN(leftEdgeValue);

def barIndexH = if pvtCond then

fold i = 1 to rightBars + 1 with p=1 while p do

series > GetValue(series, - i) else na;

def barIndexL = if pvtCond then

fold j = 1 to rightBars + 1 with q=1 while q do

series < GetValue(series, - j) else na;

def PivotPoint;

if isHigh {

PivotPoint = if HH and barIndexH then series else na;

} else {

PivotPoint = if LL and barIndexL then series else na;

}

plot pvt = PivotPoint;

}

def ph = pivot(high[lookback], lookback, lookback, yes);

def pl = pivot(low[lookback], lookback, lookback, no);

def nPh = !isNaN(ph);

def nPl = !isNaN(pl);

def x1 = if (nPh or nPl) then lookback else x1[1] + 1;

def x = if x1 > lookback then x1 else lookback;

def len1 = if dynamic then x else lookback;

def sartCond = bar >= highestAll(bar - len1);

def highs = if sartCond then Max(highs[1], high) else highs[1];

def lows = if sartCond then Min(if(lows[1], lows[1], low), low) else lows[1];

def maxx = highestAll(InertiaAll(highs, 2));

def minn = highestAll(InertiaAll(lows, 2));

def step = (maxx - minn) / res;

plot hh = if sartCond and !last then maxx else na;

plot ll = if sartCond and !last then minn else na;

hh.SetDefaultColor(Color.RED);

ll.SetDefaultColor(Color.GREEN);

AddCloud(hh, ll, CreateColor(26,26,26));

def lvl00 = minn;

def lvl01 = if res < 1 then na else minn + ((01)\*step);

def lvl02 = if res < 2 then na else minn + ((02)\*step);

def lvl03 = if res < 3 then na else minn + ((03)\*step);

def lvl04 = if res < 4 then na else minn + ((04)\*step);

def lvl05 = if res < 5 then na else minn + ((05)\*step);

def lvl06 = if res < 6 then na else minn + ((06)\*step);

def lvl07 = if res < 7 then na else minn + ((07)\*step);

def lvl08 = if res < 8 then na else minn + ((08)\*step);

def lvl09 = if res < 9 then na else minn + ((09)\*step);

def lvl10 = if res < 10 then na else minn + ((10)\*step);

def lvl11 = if res < 11 then na else minn + ((11)\*step);

def lvl12 = if res < 12 then na else minn + ((12)\*step);

def lvl13 = if res < 13 then na else minn + ((13)\*step);

def lvl14 = if res < 14 then na else minn + ((14)\*step);

def lvl15 = if res < 15 then na else minn + ((15)\*step);

def lvl16 = if res < 16 then na else minn + ((16)\*step);

def lvl17 = if res < 17 then na else minn + ((17)\*step);

def lvl18 = if res < 18 then na else minn + ((18)\*step);

def lvl19 = if res < 19 then na else minn + ((19)\*step);

def lvl20 = if res < 20 then na else minn + ((20)\*step);

def lvl21 = if res < 21 then na else minn + ((21)\*step);

def lvl22 = if res < 22 then na else minn + ((22)\*step);

def lvl23 = if res < 23 then na else minn + ((23)\*step);

def lvl24 = if res < 24 then na else minn + ((24)\*step);

def lvl25 = if res < 25 then na else minn + ((25)\*step);

def lvl26 = if res < 26 then na else minn + ((26)\*step);

def lvl27 = if res < 27 then na else minn + ((27)\*step);

def lvl28 = if res < 28 then na else minn + ((28)\*step);

def lvl29 = if res < 29 then na else minn + ((29)\*step);

def lvl30 = if res < 30 then na else minn + ((30)\*step);

def lvl31 = if res < 31 then na else minn + ((31)\*step);

def lvl32 = if res < 32 then na else minn + ((32)\*step);

def lvl33 = if res < 33 then na else minn + ((33)\*step);

def lvl34 = if res < 34 then na else minn + ((34)\*step);

def lvl35 = if res < 35 then na else minn + ((35)\*step);

def lvl36 = if res < 36 then na else minn + ((36)\*step);

def lvl37 = if res < 37 then na else minn + ((37)\*step);

def lvl38 = if res < 38 then na else minn + ((38)\*step);

def lvl39 = if res < 39 then na else minn + ((39)\*step);

def lvl40 = if res < 40 then na else minn + ((40)\*step);

def lvl41 = if res < 41 then na else minn + ((41)\*step);

def lvl42 = if res < 42 then na else minn + ((42)\*step);

def lvl43 = if res < 43 then na else minn + ((43)\*step);

def lvl44 = if res < 44 then na else minn + ((44)\*step);

def lvl45 = if res < 45 then na else minn + ((45)\*step);

def lvl46 = if res < 46 then na else minn + ((46)\*step);

def lvl47 = if res < 47 then na else minn + ((47)\*step);

def lvl48 = if res < 48 then na else minn + ((48)\*step);

def lvl49 = if res < 49 then na else minn + ((49)\*step);

def lvl50 = if res < 50 then na else minn + ((50)\*step);

#Script nz {

#input src = close;

# def nzL = if isNaN(src) then 0 else src;

# plot out = highestAll(inertiaAll(nzL, 2));

#}

Script vol {

input lvl1 = low;

input lvl2 = high;

input len = 1;

def x = if isNaN(len) then 1 else len;

def bin0 = fold i = 0 to x with p=1 do

p + (if !(GetValue(low,i)>lvl1 or GetValue(high,i)<lvl2) then GetValue(volume,i) else 0);

def nzL = if isNaN(bin0) then 1 else bin0;

def bin = highestAll(inertiaAll(nzL, 2));

plot out = bin;

}

def bin00 = vol(lvl01, lvl00, x);

def bin01 = vol(lvl02, lvl01, x);

def bin02 = vol(lvl03, lvl02, x);

def bin03 = vol(lvl04, lvl03, x);

def bin04 = vol(lvl05, lvl04, x);

def bin05 = vol(lvl06, lvl05, x);

def bin06 = vol(lvl07, lvl06, x);

def bin07 = vol(lvl08, lvl07, x);

def bin08 = vol(lvl09, lvl08, x);

def bin09 = vol(lvl10, lvl09, x);

def bin10 = vol(lvl11, lvl10, x);

def bin11 = vol(lvl12, lvl11, x);

def bin12 = vol(lvl13, lvl12, x);

def bin13 = vol(lvl14, lvl13, x);

def bin14 = vol(lvl15, lvl14, x);

def bin15 = vol(lvl16, lvl15, x);

def bin16 = vol(lvl17, lvl16, x);

def bin17 = vol(lvl18, lvl17, x);

def bin18 = vol(lvl19, lvl18, x);

def bin19 = vol(lvl20, lvl19, x);

def bin20 = vol(lvl21, lvl20, x);

def bin21 = vol(lvl22, lvl21, x);

def bin22 = vol(lvl23, lvl22, x);

def bin23 = vol(lvl24, lvl23, x);

def bin24 = vol(lvl25, lvl24, x);

def bin25 = vol(lvl26, lvl25, x);

def bin26 = vol(lvl27, lvl26, x);

def bin27 = vol(lvl28, lvl27, x);

def bin28 = vol(lvl29, lvl28, x);

def bin29 = vol(lvl30, lvl29, x);

def bin30 = vol(lvl31, lvl30, x);

def bin31 = vol(lvl32, lvl31, x);

def bin32 = vol(lvl33, lvl32, x);

def bin33 = vol(lvl34, lvl33, x);

def bin34 = vol(lvl35, lvl34, x);

def bin35 = vol(lvl36, lvl35, x);

def bin36 = vol(lvl37, lvl36, x);

def bin37 = vol(lvl38, lvl37, x);

def bin38 = vol(lvl39, lvl38, x);

def bin39 = vol(lvl40, lvl39, x);

def bin40 = vol(lvl41, lvl40, x);

def bin41 = vol(lvl42, lvl41, x);

def bin42 = vol(lvl43, lvl42, x);

def bin43 = vol(lvl44, lvl43, x);

def bin44 = vol(lvl45, lvl44, x);

def bin45 = vol(lvl46, lvl45, x);

def bin46 = vol(lvl47, lvl46, x);

def bin47 = vol(lvl48, lvl47, x);

def bin48 = vol(lvl49, lvl48, x);

def bin49 = vol(lvl50, lvl49, x);

def iskey00 = (bin00 < bin01\*zone);

def iskey01 = (bin01 < bin00\*zone) and (if res == 2 then yes else bin01 < bin02\*zone);

def iskey02 = (bin02 < bin01\*zone) and (if res == 3 then yes else bin02 < bin03\*zone);

def iskey03 = (bin03 < bin02\*zone) and (if res == 4 then yes else bin03 < bin04\*zone);

def iskey04 = (bin04 < bin03\*zone) and (if res == 5 then yes else bin04 < bin05\*zone);

def iskey05 = (bin05 < bin04\*zone) and (if res == 6 then yes else bin05 < bin06\*zone);

def iskey06 = (bin06 < bin05\*zone) and (if res == 7 then yes else bin06 < bin07\*zone);

def iskey07 = (bin07 < bin06\*zone) and (if res == 8 then yes else bin07 < bin08\*zone);

def iskey08 = (bin08 < bin07\*zone) and (if res == 9 then yes else bin08 < bin09\*zone);

def iskey09 = (bin09 < bin08\*zone) and (if res == 10 then yes else bin09 < bin10\*zone);

def iskey10 = (bin10 < bin09\*zone) and (if res == 11 then yes else bin10 < bin11\*zone);

def iskey11 = (bin11 < bin10\*zone) and (if res == 12 then yes else bin11 < bin12\*zone);

def iskey12 = (bin12 < bin11\*zone) and (if res == 13 then yes else bin12 < bin13\*zone);

def iskey13 = (bin13 < bin12\*zone) and (if res == 14 then yes else bin13 < bin14\*zone);

def iskey14 = (bin14 < bin13\*zone) and (if res == 15 then yes else bin14 < bin15\*zone);

def iskey15 = (bin15 < bin14\*zone) and (if res == 16 then yes else bin15 < bin16\*zone);

def iskey16 = (bin16 < bin15\*zone) and (if res == 17 then yes else bin16 < bin17\*zone);

def iskey17 = (bin17 < bin16\*zone) and (if res == 18 then yes else bin17 < bin18\*zone);

def iskey18 = (bin18 < bin17\*zone) and (if res == 19 then yes else bin18 < bin19\*zone);

def iskey19 = (bin19 < bin18\*zone) and (if res == 20 then yes else bin19 < bin20\*zone);

def iskey20 = (bin20 < bin19\*zone) and (if res == 21 then yes else bin20 < bin21\*zone);

def iskey21 = (bin21 < bin20\*zone) and (if res == 22 then yes else bin21 < bin22\*zone);

def iskey22 = (bin22 < bin21\*zone) and (if res == 23 then yes else bin22 < bin23\*zone);

def iskey23 = (bin23 < bin22\*zone) and (if res == 24 then yes else bin23 < bin24\*zone);

def iskey24 = (bin24 < bin23\*zone) and (if res == 25 then yes else bin24 < bin25\*zone);

def iskey25 = (bin25 < bin24\*zone) and (if res == 26 then yes else bin25 < bin26\*zone);

def iskey26 = (bin26 < bin25\*zone) and (if res == 27 then yes else bin26 < bin27\*zone);

def iskey27 = (bin27 < bin26\*zone) and (if res == 28 then yes else bin27 < bin28\*zone);

def iskey28 = (bin28 < bin27\*zone) and (if res == 29 then yes else bin28 < bin29\*zone);

def iskey29 = (bin29 < bin28\*zone) and (if res == 30 then yes else bin29 < bin30\*zone);

def iskey30 = (bin30 < bin29\*zone) and (if res == 31 then yes else bin30 < bin31\*zone);

def iskey31 = (bin31 < bin30\*zone) and (if res == 32 then yes else bin31 < bin32\*zone);

def iskey32 = (bin32 < bin31\*zone) and (if res == 33 then yes else bin32 < bin33\*zone);

def iskey33 = (bin33 < bin32\*zone) and (if res == 34 then yes else bin33 < bin34\*zone);

def iskey34 = (bin34 < bin33\*zone) and (if res == 35 then yes else bin34 < bin35\*zone);

def iskey35 = (bin35 < bin34\*zone) and (if res == 36 then yes else bin35 < bin36\*zone);

def iskey36 = (bin36 < bin35\*zone) and (if res == 37 then yes else bin36 < bin37\*zone);

def iskey37 = (bin37 < bin36\*zone) and (if res == 38 then yes else bin37 < bin38\*zone);

def iskey38 = (bin38 < bin37\*zone) and (if res == 39 then yes else bin38 < bin39\*zone);

def iskey39 = (bin39 < bin38\*zone) and (if res == 40 then yes else bin39 < bin40\*zone);

def iskey40 = (bin40 < bin39\*zone) and (if res == 41 then yes else bin40 < bin41\*zone);

def iskey41 = (bin41 < bin40\*zone) and (if res == 42 then yes else bin41 < bin42\*zone);

def iskey42 = (bin42 < bin41\*zone) and (if res == 43 then yes else bin42 < bin43\*zone);

def iskey43 = (bin43 < bin42\*zone) and (if res == 44 then yes else bin43 < bin44\*zone);

def iskey44 = (bin44 < bin43\*zone) and (if res == 45 then yes else bin44 < bin45\*zone);

def iskey45 = (bin45 < bin44\*zone) and (if res == 46 then yes else bin45 < bin46\*zone);

def iskey46 = (bin46 < bin45\*zone) and (if res == 47 then yes else bin46 < bin47\*zone);

def iskey47 = (bin47 < bin46\*zone) and (if res == 48 then yes else bin47 < bin48\*zone);

def iskey48 = (bin48 < bin47\*zone) and (if res == 49 then yes else bin48 < bin49\*zone);

def iskey49 = (bin49 < bin48\*zone);

def up00 = if iskey00 and lvl01 then lvl01 else na;

def up01 = if iskey01 and lvl02 then lvl02 else na;

def up02 = if iskey02 and lvl03 then lvl03 else na;

def up03 = if iskey03 and lvl04 then lvl04 else na;

def up04 = if iskey04 and lvl05 then lvl05 else na;

def up05 = if iskey05 and lvl06 then lvl06 else na;

def up06 = if iskey06 and lvl07 then lvl07 else na;

def up07 = if iskey07 and lvl08 then lvl08 else na;

def up08 = if iskey08 and lvl09 then lvl09 else na;

def up09 = if iskey09 and lvl10 then lvl10 else na;

def up10 = if iskey10 and lvl11 then lvl11 else na;

def up11 = if iskey11 and lvl12 then lvl12 else na;

def up12 = if iskey12 and lvl13 then lvl13 else na;

def up13 = if iskey13 and lvl14 then lvl14 else na;

def up14 = if iskey14 and lvl15 then lvl15 else na;

def up15 = if iskey15 and lvl16 then lvl16 else na;

def up16 = if iskey16 and lvl17 then lvl17 else na;

def up17 = if iskey17 and lvl18 then lvl18 else na;

def up18 = if iskey18 and lvl19 then lvl19 else na;

def up19 = if iskey19 and lvl20 then lvl20 else na;

def up20 = if iskey20 and lvl21 then lvl21 else na;

def up21 = if iskey21 and lvl22 then lvl22 else na;

def up22 = if iskey22 and lvl23 then lvl23 else na;

def up23 = if iskey23 and lvl24 then lvl24 else na;

def up24 = if iskey24 and lvl25 then lvl25 else na;

def up25 = if iskey25 and lvl26 then lvl26 else na;

def up26 = if iskey26 and lvl27 then lvl27 else na;

def up27 = if iskey27 and lvl28 then lvl28 else na;

def up28 = if iskey28 and lvl29 then lvl29 else na;

def up29 = if iskey29 and lvl30 then lvl30 else na;

def up30 = if iskey30 and lvl31 then lvl31 else na;

def up31 = if iskey31 and lvl32 then lvl32 else na;

def up32 = if iskey32 and lvl33 then lvl33 else na;

def up33 = if iskey33 and lvl34 then lvl34 else na;

def up34 = if iskey34 and lvl35 then lvl35 else na;

def up35 = if iskey35 and lvl36 then lvl36 else na;

def up36 = if iskey36 and lvl37 then lvl37 else na;

def up37 = if iskey37 and lvl38 then lvl38 else na;

def up38 = if iskey38 and lvl39 then lvl39 else na;

def up39 = if iskey39 and lvl40 then lvl40 else na;

def up40 = if iskey40 and lvl41 then lvl41 else na;

def up41 = if iskey41 and lvl42 then lvl42 else na;

def up42 = if iskey42 and lvl43 then lvl43 else na;

def up43 = if iskey43 and lvl44 then lvl44 else na;

def up44 = if iskey44 and lvl45 then lvl45 else na;

def up45 = if iskey45 and lvl46 then lvl46 else na;

def up46 = if iskey46 and lvl47 then lvl47 else na;

def up47 = if iskey47 and lvl48 then lvl48 else na;

def up48 = if iskey48 and lvl49 then lvl49 else na;

def up49 = if iskey49 and lvl50 then lvl50 else na;

#-----

def dn00 = if iskey00 and lvl00 then lvl00 else na;

def dn01 = if iskey01 and lvl01 then lvl01 else na;

def dn02 = if iskey02 and lvl02 then lvl02 else na;

def dn03 = if iskey03 and lvl03 then lvl03 else na;

def dn04 = if iskey04 and lvl04 then lvl04 else na;

def dn05 = if iskey05 and lvl05 then lvl05 else na;

def dn06 = if iskey06 and lvl06 then lvl06 else na;

def dn07 = if iskey07 and lvl07 then lvl07 else na;

def dn08 = if iskey08 and lvl08 then lvl08 else na;

def dn09 = if iskey09 and lvl09 then lvl09 else na;

def dn10 = if iskey10 and lvl10 then lvl10 else na;

def dn11 = if iskey11 and lvl11 then lvl11 else na;

def dn12 = if iskey12 and lvl12 then lvl12 else na;

def dn13 = if iskey13 and lvl13 then lvl13 else na;

def dn14 = if iskey14 and lvl14 then lvl14 else na;

def dn15 = if iskey15 and lvl15 then lvl15 else na;

def dn16 = if iskey16 and lvl16 then lvl16 else na;

def dn17 = if iskey17 and lvl17 then lvl17 else na;

def dn18 = if iskey18 and lvl18 then lvl18 else na;

def dn19 = if iskey19 and lvl19 then lvl19 else na;

def dn20 = if iskey20 and lvl20 then lvl20 else na;

def dn21 = if iskey21 and lvl21 then lvl21 else na;

def dn22 = if iskey22 and lvl22 then lvl22 else na;

def dn23 = if iskey23 and lvl23 then lvl23 else na;

def dn24 = if iskey24 and lvl24 then lvl24 else na;

def dn25 = if iskey25 and lvl25 then lvl25 else na;

def dn26 = if iskey26 and lvl26 then lvl26 else na;

def dn27 = if iskey27 and lvl27 then lvl27 else na;

def dn28 = if iskey28 and lvl28 then lvl28 else na;

def dn29 = if iskey29 and lvl29 then lvl29 else na;

def dn30 = if iskey30 and lvl30 then lvl30 else na;

def dn31 = if iskey31 and lvl31 then lvl31 else na;

def dn32 = if iskey32 and lvl32 then lvl32 else na;

def dn33 = if iskey33 and lvl33 then lvl33 else na;

def dn34 = if iskey34 and lvl34 then lvl34 else na;

def dn35 = if iskey35 and lvl35 then lvl35 else na;

def dn36 = if iskey36 and lvl36 then lvl36 else na;

def dn37 = if iskey37 and lvl37 then lvl37 else na;

def dn38 = if iskey38 and lvl38 then lvl38 else na;

def dn39 = if iskey39 and lvl39 then lvl39 else na;

def dn40 = if iskey40 and lvl40 then lvl40 else na;

def dn41 = if iskey41 and lvl41 then lvl41 else na;

def dn42 = if iskey42 and lvl42 then lvl42 else na;

def dn43 = if iskey43 and lvl43 then lvl43 else na;

def dn44 = if iskey44 and lvl44 then lvl44 else na;

def dn45 = if iskey45 and lvl45 then lvl45 else na;

def dn46 = if iskey46 and lvl46 then lvl46 else na;

def dn47 = if iskey47 and lvl47 then lvl47 else na;

def dn48 = if iskey48 and lvl48 then lvl48 else na;

def dn49 = if iskey49 and lvl49 then lvl49 else na;

def cond00 = if close > up00 then 1 else

if close < dn00 then 0 else cond00[1];

def cond01 = if close > up01 then 1 else

if close < dn01 then 0 else cond01[1];

def cond02 = if close > up02 then 1 else

if close < dn02 then 0 else cond02[1];

def cond03 = if close > up03 then 1 else

if close < dn03 then 0 else cond03[1];

def cond04 = if close > up04 then 1 else

if close < dn04 then 0 else cond04[1];

def cond05 = if close > up05 then 1 else

if close < dn05 then 0 else cond05[1];

def cond06 = if close > up06 then 1 else

if close < dn06 then 0 else cond06[1];

def cond07 = if close > up07 then 1 else

if close < dn07 then 0 else cond07[1];

def cond08 = if close > up08 then 1 else

if close < dn08 then 0 else cond08[1];

def cond09 = if close > up09 then 1 else

if close < dn09 then 0 else cond09[1];

def cond10 = if close > up10 then 1 else

if close < dn10 then 0 else cond10[1];

def cond11 = if close > up11 then 1 else

if close < dn11 then 0 else cond11[1];

def cond12 = if close > up12 then 1 else

if close < dn12 then 0 else cond12[1];

def cond13 = if close > up13 then 1 else

if close < dn13 then 0 else cond13[1];

def cond14 = if close > up14 then 1 else

if close < dn14 then 0 else cond14[1];

def cond15 = if close > up15 then 1 else

if close < dn15 then 0 else cond15[1];

def cond16 = if close > up16 then 1 else

if close < dn16 then 0 else cond16[1];

def cond17 = if close > up17 then 1 else

if close < dn17 then 0 else cond17[1];

def cond18 = if close > up18 then 1 else

if close < dn18 then 0 else cond18[1];

def cond19 = if close > up19 then 1 else

if close < dn19 then 0 else cond19[1];

def cond20 = if close > up20 then 1 else

if close < dn20 then 0 else cond20[1];

def cond21 = if close > up21 then 1 else

if close < dn21 then 0 else cond21[1];

def cond22 = if close > up22 then 1 else

if close < dn22 then 0 else cond22[1];

def cond23 = if close > up23 then 1 else

if close < dn23 then 0 else cond23[1];

def cond24 = if close > up24 then 1 else

if close < dn24 then 0 else cond24[1];

def cond25 = if close > up25 then 1 else

if close < dn25 then 0 else cond25[1];

def cond26 = if close > up26 then 1 else

if close < dn26 then 0 else cond26[1];

def cond27 = if close > up27 then 1 else

if close < dn27 then 0 else cond27[1];

def cond28 = if close > up28 then 1 else

if close < dn28 then 0 else cond28[1];

def cond29 = if close > up29 then 1 else

if close < dn29 then 0 else cond29[1];

def cond30 = if close > up30 then 1 else

if close < dn30 then 0 else cond30[1];

def cond31 = if close > up31 then 1 else

if close < dn31 then 0 else cond31[1];

def cond32 = if close > up32 then 1 else

if close < dn32 then 0 else cond32[1];

def cond33 = if close > up33 then 1 else

if close < dn33 then 0 else cond33[1];

def cond34 = if close > up34 then 1 else

if close < dn34 then 0 else cond34[1];

def cond35 = if close > up35 then 1 else

if close < dn35 then 0 else cond35[1];

def cond36 = if close > up36 then 1 else

if close < dn36 then 0 else cond36[1];

def cond37 = if close > up37 then 1 else

if close < dn37 then 0 else cond37[1];

def cond38 = if close > up38 then 1 else

if close < dn38 then 0 else cond38[1];

def cond39 = if close > up39 then 1 else

if close < dn39 then 0 else cond39[1];

def cond40 = if close > up40 then 1 else

if close < dn40 then 0 else cond40[1];

def cond41 = if close > up41 then 1 else

if close < dn41 then 0 else cond41[1];

def cond42 = if close > up42 then 1 else

if close < dn42 then 0 else cond42[1];

def cond43 = if close > up43 then 1 else

if close < dn43 then 0 else cond43[1];

def cond44 = if close > up44 then 1 else

if close < dn44 then 0 else cond44[1];

def cond45 = if close > up45 then 1 else

if close < dn45 then 0 else cond45[1];

def cond46 = if close > up46 then 1 else

if close < dn46 then 0 else cond46[1];

def cond47 = if close > up47 then 1 else

if close < dn47 then 0 else cond47[1];

def cond48 = if close > up48 then 1 else

if close < dn48 then 0 else cond48[1];

def cond49 = if close > up49 then 1 else

if close < dn49 then 0 else cond49[1];

def con00 = if sartCond and !last then cond00 else na;

def con01 = if sartCond and !last then cond01 else na;

def con02 = if sartCond and !last then cond02 else na;

def con03 = if sartCond and !last then cond03 else na;

def con04 = if sartCond and !last then cond04 else na;

def con05 = if sartCond and !last then cond05 else na;

def con06 = if sartCond and !last then cond06 else na;

def con07 = if sartCond and !last then cond07 else na;

def con08 = if sartCond and !last then cond08 else na;

def con09 = if sartCond and !last then cond09 else na;

def con10 = if sartCond and !last then cond10 else na;

def con11 = if sartCond and !last then cond11 else na;

def con12 = if sartCond and !last then cond12 else na;

def con13 = if sartCond and !last then cond13 else na;

def con14 = if sartCond and !last then cond14 else na;

def con15 = if sartCond and !last then cond15 else na;

def con16 = if sartCond and !last then cond16 else na;

def con17 = if sartCond and !last then cond17 else na;

def con18 = if sartCond and !last then cond18 else na;

def con19 = if sartCond and !last then cond19 else na;

def con20 = if sartCond and !last then cond20 else na;

def con21 = if sartCond and !last then cond21 else na;

def con22 = if sartCond and !last then cond22 else na;

def con23 = if sartCond and !last then cond23 else na;

def con24 = if sartCond and !last then cond24 else na;

def con25 = if sartCond and !last then cond25 else na;

def con26 = if sartCond and !last then cond26 else na;

def con27 = if sartCond and !last then cond27 else na;

def con28 = if sartCond and !last then cond28 else na;

def con29 = if sartCond and !last then cond29 else na;

def con30 = if sartCond and !last then cond30 else na;

def con31 = if sartCond and !last then cond31 else na;

def con32 = if sartCond and !last then cond32 else na;

def con33 = if sartCond and !last then cond33 else na;

def con34 = if sartCond and !last then cond34 else na;

def con35 = if sartCond and !last then cond35 else na;

def con36 = if sartCond and !last then cond36 else na;

def con37 = if sartCond and !last then cond37 else na;

def con38 = if sartCond and !last then cond38 else na;

def con39 = if sartCond and !last then cond39 else na;

def con40 = if sartCond and !last then cond40 else na;

def con41 = if sartCond and !last then cond41 else na;

def con42 = if sartCond and !last then cond42 else na;

def con43 = if sartCond and !last then cond43 else na;

def con44 = if sartCond and !last then cond44 else na;

def con45 = if sartCond and !last then cond45 else na;

def con46 = if sartCond and !last then cond46 else na;

def con47 = if sartCond and !last then cond47 else na;

def con48 = if sartCond and !last then cond48 else na;

def con49 = if sartCond and !last then cond49 else na;

def level00 = if !isNaN(con00) then if con00 then up00 else dn00 else na;

def level01 = if !isNaN(con01) then if con01 then up01 else dn01 else na;

def level02 = if !isNaN(con02) then if con02 then up02 else dn02 else na;

def level03 = if !isNaN(con03) then if con03 then up03 else dn03 else na;

def level04 = if !isNaN(con04) then if con04 then up04 else dn04 else na;

def level05 = if !isNaN(con05) then if con05 then up05 else dn05 else na;

def level06 = if !isNaN(con06) then if con06 then up06 else dn06 else na;

def level07 = if !isNaN(con07) then if con07 then up07 else dn07 else na;

def level08 = if !isNaN(con08) then if con08 then up08 else dn08 else na;

def level09 = if !isNaN(con09) then if con09 then up09 else dn09 else na;

def level10 = if !isNaN(con10) then if con10 then up10 else dn10 else na;

def level11 = if !isNaN(con11) then if con11 then up11 else dn11 else na;

def level12 = if !isNaN(con12) then if con12 then up12 else dn12 else na;

def level13 = if !isNaN(con13) then if con13 then up13 else dn13 else na;

def level14 = if !isNaN(con14) then if con14 then up14 else dn14 else na;

def level15 = if !isNaN(con15) then if con15 then up15 else dn15 else na;

def level16 = if !isNaN(con16) then if con16 then up16 else dn16 else na;

def level17 = if !isNaN(con17) then if con17 then up17 else dn17 else na;

def level18 = if !isNaN(con18) then if con18 then up18 else dn18 else na;

def level19 = if !isNaN(con19) then if con19 then up19 else dn19 else na;

def level20 = if !isNaN(con20) then if con20 then up20 else dn20 else na;

def level21 = if !isNaN(con21) then if con21 then up21 else dn21 else na;

def level22 = if !isNaN(con22) then if con22 then up22 else dn22 else na;

def level23 = if !isNaN(con23) then if con23 then up23 else dn23 else na;

def level24 = if !isNaN(con24) then if con24 then up24 else dn24 else na;

def level25 = if !isNaN(con25) then if con25 then up25 else dn25 else na;

def level26 = if !isNaN(con26) then if con26 then up26 else dn26 else na;

def level27 = if !isNaN(con27) then if con27 then up27 else dn27 else na;

def level28 = if !isNaN(con28) then if con28 then up28 else dn28 else na;

def level29 = if !isNaN(con29) then if con29 then up29 else dn29 else na;

def level30 = if !isNaN(con30) then if con30 then up30 else dn30 else na;

def level31 = if !isNaN(con31) then if con31 then up31 else dn31 else na;

def level32 = if !isNaN(con32) then if con32 then up32 else dn32 else na;

def level33 = if !isNaN(con33) then if con33 then up33 else dn33 else na;

def level34 = if !isNaN(con34) then if con34 then up34 else dn34 else na;

def level35 = if !isNaN(con35) then if con35 then up35 else dn35 else na;

def level36 = if !isNaN(con36) then if con36 then up36 else dn36 else na;

def level37 = if !isNaN(con37) then if con37 then up37 else dn37 else na;

def level38 = if !isNaN(con38) then if con38 then up38 else dn38 else na;

def level39 = if !isNaN(con39) then if con39 then up39 else dn39 else na;

def level40 = if !isNaN(con40) then if con40 then up40 else dn40 else na;

def level41 = if !isNaN(con41) then if con41 then up41 else dn41 else na;

def level42 = if !isNaN(con42) then if con42 then up42 else dn42 else na;

def level43 = if !isNaN(con43) then if con43 then up43 else dn43 else na;

def level44 = if !isNaN(con44) then if con44 then up44 else dn44 else na;

def level45 = if !isNaN(con45) then if con45 then up45 else dn45 else na;

def level46 = if !isNaN(con46) then if con46 then up46 else dn46 else na;

def level47 = if !isNaN(con47) then if con47 then up47 else dn47 else na;

def level48 = if !isNaN(con48) then if con48 then up48 else dn48 else na;

def level49 = if !isNaN(con49) then if con49 then up49 else dn49 else na;

AddCloud(level00, if con00 then dn00 else up00, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level01, if con01 then dn01 else up01, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level02, if con02 then dn02 else up02, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level03, if con03 then dn03 else up03, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level04, if con04 then dn04 else up04, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level05, if con05 then dn05 else up05, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level06, if con06 then dn06 else up06, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level07, if con07 then dn07 else up07, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level08, if con08 then dn08 else up08, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level09, if con09 then dn09 else up09, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level10, if con10 then dn10 else up10, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level11, if con11 then dn11 else up11, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level12, if con12 then dn12 else up12, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level13, if con13 then dn13 else up13, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level14, if con14 then dn14 else up14, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level15, if con15 then dn15 else up15, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level16, if con16 then dn16 else up16, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level17, if con17 then dn17 else up17, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level18, if con18 then dn18 else up18, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level19, if con19 then dn19 else up19, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level20, if con20 then dn20 else up20, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level21, if con21 then dn21 else up21, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level22, if con22 then dn22 else up22, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level23, if con23 then dn23 else up23, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level24, if con24 then dn24 else up24, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level25, if con25 then dn25 else up25, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level26, if con26 then dn26 else up26, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level27, if con27 then dn27 else up27, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level28, if con28 then dn28 else up28, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level29, if con29 then dn29 else up29, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level30, if con30 then dn30 else up30, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level31, if con31 then dn31 else up31, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level32, if con32 then dn32 else up32, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level33, if con33 then dn33 else up33, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level34, if con34 then dn34 else up34, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level35, if con35 then dn35 else up35, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level36, if con36 then dn36 else up36, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level37, if con37 then dn37 else up37, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level38, if con38 then dn38 else up38, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level39, if con39 then dn39 else up39, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level40, if con40 then dn40 else up40, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level41, if con41 then dn41 else up41, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level42, if con42 then dn42 else up42, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level43, if con43 then dn43 else up43, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level44, if con44 then dn44 else up44, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level45, if con45 then dn45 else up45, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level46, if con46 then dn46 else up46, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level47, if con47 then dn47 else up47, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level48, if con48 then dn48 else up48, Color.DARK\_GREEN, Color.DARK\_RED, yes);

AddCloud(level49, if con49 then dn49 else up49, Color.DARK\_GREEN, Color.DARK\_RED, yes);

#AddChartBubble(x, low, x, Color.RED, no);

def ph2 = if high == highs then bar else ph2[1];

def pl2 = if low == lows then bar else pl2[1];

plot hhPoint = if bar == highestAll(ph2) then high else na;

plot llPoint = if bar == highestAll(pl2) then low else na;

hhPoint.SetLineWeight(2);

llPoint.SetLineWeight(2);

hhPoint.SetPaintingStrategy(PaintingStrategy.POINTS);

llPoint.SetPaintingStrategy(PaintingStrategy.POINTS);

hhPoint.SetDefaultColor(Color.WHITE);

llPoint.SetDefaultColor(Color.WHITE);

def totVol = fold j = 0 to x with q do

q + GetValue(volume, j);

def m1 = Max(bin00,Max(bin01,Max(bin02,Max(bin03,Max(bin04,Max(bin05,Max(bin06,Max(bin07,Max(bin08, Max(bin09, bin10))))))))));

def m2 = Max(bin11,Max(bin12,Max(bin13,Max(bin14,Max(bin15,Max(bin16,Max(bin17,Max(bin18,Max(bin19, bin20)))))))));

def m3 = Max(bin21,Max(bin22,Max(bin23,Max(bin24,Max(bin25,Max(bin26,Max(bin27,Max(bin28,Max(bin29, bin30)))))))));

def m4 = Max(bin31,Max(bin32,Max(bin33,Max(bin34,Max(bin35,Max(bin36,Max(bin37,Max(bin38,Max(bin39, bin40)))))))));

def m5 = Max(bin41,Max(bin42,Max(bin43,Max(bin44,Max(bin45,Max(bin46,Max(bin47,Max(bin48, bin49))))))));

def maxBin = Max(m1, Max(m2, Max(m3, Max(m4, m5))));

def start = if !isNaN(hh) then start[1] + 1 else 0;

def center = (bar == highestAll(bar- floor(highestAll(start)/2)));

def loc;

Switch (labelOptions) {

Case "Left" :

loc = !isNaN(hh) and isNaN(hh[1]);

Case "Right" :

loc = isNaN(hh) and !isNaN(hh[1]);

Case "Don't Show" :

loc = no;

Default :

loc = center;

}

AddChartBubble(loc, (up00 + dn00) / 2, AsPercent(1 - bin00 / maxBin), Color.WHITE);

AddChartBubble(loc, (up01 + dn01) / 2, AsPercent(1 - bin01 / maxBin), Color.WHITE);

AddChartBubble(loc, (up02 + dn02) / 2, AsPercent(1 - bin02 / maxBin), Color.WHITE);

AddChartBubble(loc, (up03 + dn03) / 2, AsPercent(1 - bin03 / maxBin), Color.WHITE);

AddChartBubble(loc, (up04 + dn04) / 2, AsPercent(1 - bin04 / maxBin), Color.WHITE);

AddChartBubble(loc, (up05 + dn05) / 2, AsPercent(1 - bin05 / maxBin), Color.WHITE);

AddChartBubble(loc, (up06 + dn06) / 2, AsPercent(1 - bin06 / maxBin), Color.WHITE);

AddChartBubble(loc, (up07 + dn07) / 2, AsPercent(1 - bin07 / maxBin), Color.WHITE);

AddChartBubble(loc, (up08 + dn08) / 2, AsPercent(1 - bin08 / maxBin), Color.WHITE);

AddChartBubble(loc, (up09 + dn09) / 2, AsPercent(1 - bin09 / maxBin), Color.WHITE);

AddChartBubble(loc, (up10 + dn10) / 2, AsPercent(1 - bin10 / maxBin), Color.WHITE);

AddChartBubble(loc, (up11 + dn11) / 2, AsPercent(1 - bin11 / maxBin), Color.WHITE);

AddChartBubble(loc, (up12 + dn12) / 2, AsPercent(1 - bin12 / maxBin), Color.WHITE);

AddChartBubble(loc, (up13 + dn13) / 2, AsPercent(1 - bin13 / maxBin), Color.WHITE);

AddChartBubble(loc, (up14 + dn14) / 2, AsPercent(1 - bin14 / maxBin), Color.WHITE);

AddChartBubble(loc, (up15 + dn15) / 2, AsPercent(1 - bin15 / maxBin), Color.WHITE);

AddChartBubble(loc, (up16 + dn16) / 2, AsPercent(1 - bin16 / maxBin), Color.WHITE);

AddChartBubble(loc, (up17 + dn17) / 2, AsPercent(1 - bin17 / maxBin), Color.WHITE);

AddChartBubble(loc, (up18 + dn18) / 2, AsPercent(1 - bin18 / maxBin), Color.WHITE);

AddChartBubble(loc, (up19 + dn19) / 2, AsPercent(1 - bin19 / maxBin), Color.WHITE);

AddChartBubble(loc, (up20 + dn20) / 2, AsPercent(1 - bin20 / maxBin), Color.WHITE);

AddChartBubble(loc, (up21 + dn21) / 2, AsPercent(1 - bin21 / maxBin), Color.WHITE);

AddChartBubble(loc, (up22 + dn22) / 2, AsPercent(1 - bin22 / maxBin), Color.WHITE);

AddChartBubble(loc, (up23 + dn23) / 2, AsPercent(1 - bin23 / maxBin), Color.WHITE);

AddChartBubble(loc, (up24 + dn24) / 2, AsPercent(1 - bin24 / maxBin), Color.WHITE);

AddChartBubble(loc, (up25 + dn25) / 2, AsPercent(1 - bin25 / maxBin), Color.WHITE);

AddChartBubble(loc, (up26 + dn26) / 2, AsPercent(1 - bin26 / maxBin), Color.WHITE);

AddChartBubble(loc, (up27 + dn27) / 2, AsPercent(1 - bin27 / maxBin), Color.WHITE);

AddChartBubble(loc, (up28 + dn28) / 2, AsPercent(1 - bin28 / maxBin), Color.WHITE);

AddChartBubble(loc, (up29 + dn29) / 2, AsPercent(1 - bin29 / maxBin), Color.WHITE);

AddChartBubble(loc, (up30 + dn30) / 2, AsPercent(1 - bin30 / maxBin), Color.WHITE);

AddChartBubble(loc, (up31 + dn31) / 2, AsPercent(1 - bin31 / maxBin), Color.WHITE);

AddChartBubble(loc, (up32 + dn32) / 2, AsPercent(1 - bin32 / maxBin), Color.WHITE);

AddChartBubble(loc, (up33 + dn33) / 2, AsPercent(1 - bin33 / maxBin), Color.WHITE);

AddChartBubble(loc, (up34 + dn34) / 2, AsPercent(1 - bin34 / maxBin), Color.WHITE);

AddChartBubble(loc, (up35 + dn35) / 2, AsPercent(1 - bin35 / maxBin), Color.WHITE);

AddChartBubble(loc, (up36 + dn36) / 2, AsPercent(1 - bin36 / maxBin), Color.WHITE);

AddChartBubble(loc, (up37 + dn37) / 2, AsPercent(1 - bin37 / maxBin), Color.WHITE);

AddChartBubble(loc, (up38 + dn38) / 2, AsPercent(1 - bin38 / maxBin), Color.WHITE);

AddChartBubble(loc, (up39 + dn39) / 2, AsPercent(1 - bin39 / maxBin), Color.WHITE);

AddChartBubble(loc, (up40 + dn40) / 2, AsPercent(1 - bin40 / maxBin), Color.WHITE);

AddChartBubble(loc, (up41 + dn41) / 2, AsPercent(1 - bin41 / maxBin), Color.WHITE);

AddChartBubble(loc, (up42 + dn42) / 2, AsPercent(1 - bin42 / maxBin), Color.WHITE);

AddChartBubble(loc, (up43 + dn43) / 2, AsPercent(1 - bin43 / maxBin), Color.WHITE);

AddChartBubble(loc, (up44 + dn44) / 2, AsPercent(1 - bin44 / maxBin), Color.WHITE);

AddChartBubble(loc, (up45 + dn45) / 2, AsPercent(1 - bin45 / maxBin), Color.WHITE);

AddChartBubble(loc, (up46 + dn46) / 2, AsPercent(1 - bin46 / maxBin), Color.WHITE);

AddChartBubble(loc, (up47 + dn47) / 2, AsPercent(1 - bin47 / maxBin), Color.WHITE);

AddChartBubble(loc, (up48 + dn48) / 2, AsPercent(1 - bin48 / maxBin), Color.WHITE);

AddChartBubble(loc, (up49 + dn49) / 2, AsPercent(1 - bin49 / maxBin), Color.WHITE);