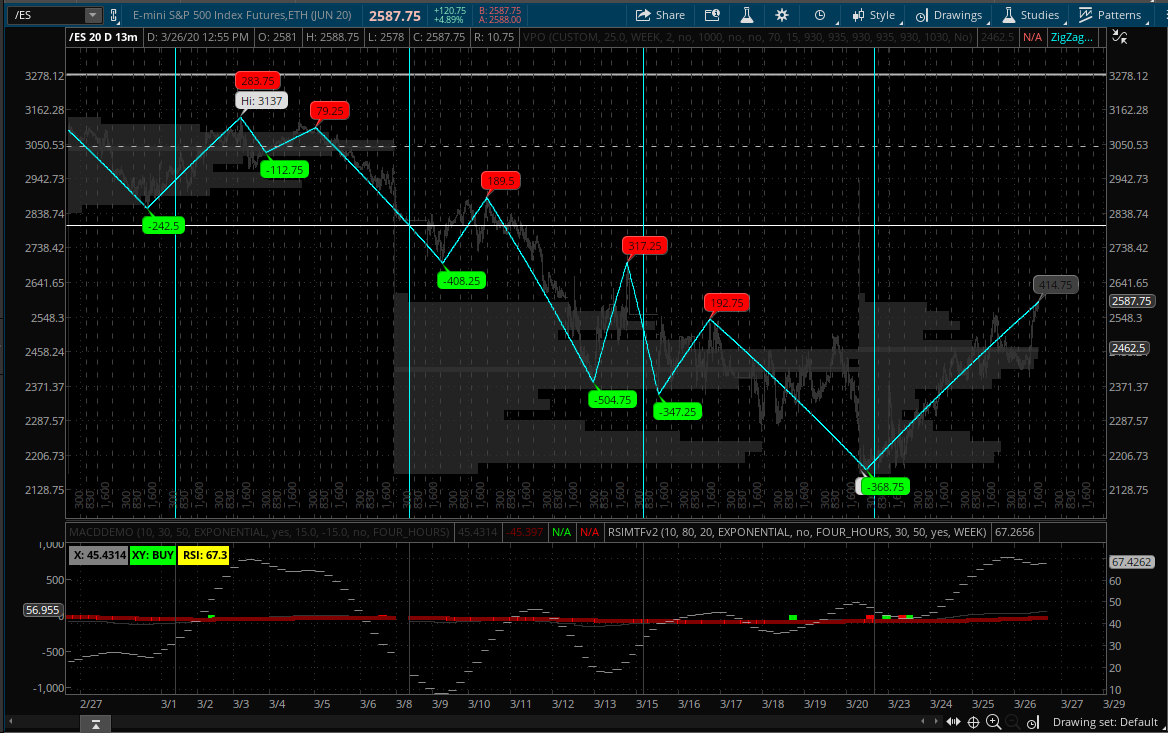
**BLOCKHIVE XYR DEMO**

ThinkorSwim(TOS) Javascript



**Custom Scripts**

**----------------------------------------------------------------**

#RSI v2

declare lower;

input length = 10;

input over\_Bought = 80;

input over\_Sold = 20;

input averageType = AverageType.EXPONENTIAL;

input showBreakoutSignals = no;

Input AGG = AggregationPeriod.four\_hours;

Input ema2 = 30;

Input malength = 50;

input showperiod = yes;

def NetChgAvg = MovingAverage(averageType, close(period = Agg) - close(period = Agg)[1], length);

def TotChgAvg = MovingAverage(averageType, AbsValue(close(period = Agg) - close(period = Agg)[1]), length);

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

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

plot OverSold = over\_Sold;

plot OverBought = over\_Bought;

Plot RSI = HullMovingAvg (RSI\_Raw,ema2); ## Added our smoothing

plot ma = expaverage(rsi,malength);

AddLabel(yes, " RSI: " + round(RSI,1) ,Color.YELLOW);

RSI.DefineColor("OverBought", GetColor(5));

RSI.DefineColor("Normal", GetColor(7));

RSI.DefineColor("OverSold", GetColor(1));

RSI.AssignValueColor(if RSI > over\_Bought then RSI.Color("OverBought") else if RSI < over\_Sold then RSI.Color("OverSold") else RSI.Color("Normal"));

OverSold.SetDefaultColor(GetColor(8));

OverBought.SetDefaultColor(GetColor(8));

plot UpSignal = if RSI crosses above OverSold then OverSold else Double.NaN;

plot DownSignal = if RSI crosses below OverBought then OverBought else Double.NaN;

UpSignal.SetHiding(!showBreakoutSignals);

DownSignal.SetHiding(!showBreakoutSignals);

RSI.DefineColor("OverBought", GetColor(5));

RSI.DefineColor("Normal", GetColor(7));

RSI.DefineColor("OverSold", GetColor(1));

RSI.AssignValueColor(if RSI > over\_Bought then RSI.Color("OverBought") else if RSI < over\_Sold then RSI.Color("OverSold") else RSI.Color("Normal"));

UpSignal.SetDefaultColor(Color.UPTICK);

UpSignal.SetPaintingStrategy(PaintingStrategy.ARROW\_UP);

DownSignal.SetDefaultColor(Color.DOWNTICK);

DownSignal.SetPaintingStrategy(PaintingStrategy.ARROW\_DOWN);

#plot onesigma = 80;

#plot negonesigma = 30;

#plot twosigma = 95;

#plot negtwosigma = 5;

#plot meanLine = 50;

#meanLine.SetDefaultColor(Color.DARK\_GRAY);

#onesigma.SetDefaultColor(Color.DARK\_GRAY);

#negonesigma.SetDefaultColor(Color.DARK\_GRAY);

#twosigma.SetDefaultColor(Color.DARK\_GRAY);

#negtwosigma.SetDefaultColor(Color.DARK\_GRAY);

#meanLine.SetLineWeight(1);

#onesigma.SetLineWeight(1);

#onesigma.SetPaintingStrategy(PaintingStrategy.DASHES);

#negonesigma.SetLineWeight(1);

#negonesigma.SetPaintingStrategy(PaintingStrategy.DASHES);

#twosigma.SetLineWeight(1);

#negtwosigma.SetLineWeight(1);

def SELL = RSI crosses above 70 ;

def BUY = RSI crosses below 30 ;

Alert(SELL, "Overbought", Alert.BAR, Sound.Chimes);

Alert(BUY, "Oversold", Alert.BAR, Sound.Chimes);

input period = {WEEK, year, default MONTH};

AddVerticalLine((period == period.WEEK and GetWeek() <> GetWeek()[1]) or (period == period.MONTH and GetMonth() <> GetMonth()[1]), "", Color.dark\_gray, Curve.FIRM);

#MACDV2

declare lower;

input fastLength = 10;

input slowLength = 30;

input MACDLength = 50;

input averageType = AverageType.EXPONENTIAL;

input showBreakoutSignals = yes;

input over\_Bought = 1;

input over\_Sold = -1;

input show\_vertical\_line = no;

input AGG = AggregationPeriod.FOUR\_HOURS;

##AddVerticalLine(upsignal && show\_vertical\_line, "", Color.UPTICK);

##AddVerticalLine(downsignal && show\_vertical\_line, "", Color.LIGHT\_RED);

plot Value =MovingAverage(averageType, close(period=agg), fastLength) - MovingAverage(averageType, close(period=agg), slowLength);

plot Avg = MovingAverage(averageType, Value, MACDLength);

plot Diff = Value - Avg;

plot ZeroLine = 0;

plot overbought = 5;

plot oversold = -5;

plot upsignal = if (Value > Avg and Value[1] <= Avg[1]) then Value[1] else Double.NaN;

upsignal.SetPaintingStrategy(PaintingStrategy.ARROW\_UP);

upsignal.SetDefaultColor(Color.CYAN);

plot downsignal = if (Value < Avg and Value[1] >= Avg[1]) then Value[1] else Double.NaN;

plot Upsignal2 = if Value crosses above over\_Sold then over\_Sold else Double.NaN;

plot Downsignal2 = if Value crosses below over\_Bought then over\_Bought else Double.NaN;

upsignal.SetHiding(!showBreakoutSignals);

downsignal.SetHiding(!showBreakoutSignals);

upsignal.SetDefaultColor(Color.WHITE);

upsignal.SetPaintingStrategy(PaintingStrategy.ARROW\_UP);

downsignal.SetDefaultColor(Color.WHITE);

downsignal.SetPaintingStrategy(PaintingStrategy.ARROW\_DOWN);

upsignal.SetHiding(!showBreakoutSignals);

downsignal.SetHiding(!showBreakoutSignals);

Value.SetDefaultColor(GetColor(9));

diff.SetDefaultColor(GetColor(9));

avg.SetDefaultColor(GetColor(5));

avg.SetPaintingStrategy(PaintingStrategy.HISTOGRAM);

avg.SetLineWeight(3);

avg.DefineColor("Positive and Up", Color.Gray);

avg.DefineColor("Positive and Down", Color.Gray);

avg.DefineColor("Negative and Down", Color.Gray);

avg.DefineColor("Negative and Up", Color.Gray);

avg.AssignValueColor(if avg >= 0 then if avg > avg[1] then avg.Color("Positive and Up") else avg.Color("Positive and Down") else if avg < avg[1] then avg.Color("Negative and Down") else avg.Color("Negative and Up"));

ZeroLine.SetDefaultColor(GetColor(0));

overbought.SetDefaultColor(GetColor(0));

oversold.SetDefaultColor(GetColor(0));

upsignal.SetDefaultColor(Color.GREEN);

upsignal.SetPaintingStrategy(PaintingStrategy.SQUARES);

downsignal.SetDefaultColor(Color.RED);

downsignal.SetPaintingStrategy(PaintingStrategy.SQUARES);

#avg.SetPaintingStrategy(PaintingStrategy.LINE);

#avg.SetLineWeight(2);

#avg.AssignValueColor(if avg > avg[1] and avg[1] > avg[2]

# then Color.dark\_gray else Color.Gray);

#value.AssignValueColor(if value > value[1] and value[1] > value[2]

# then Color.green else Color.red);

AddVerticalLine(upsignal && show\_vertical\_line, "", Color.UPTICK);

AddVerticalLine(downsignal && show\_vertical\_line, "", Color.LIGHT\_RED);

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

AddLabel(yes, " X: " + Value , Color.GRAY);

#(yes," Y: " + avg + ")

AddLabel(yes, Concat("XY: ", Concat("",

if Avg < Value then "BUY" else "SELL")),

if Avg > Value then Color.RED else Color.GREEN);

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

#WEEKLY LINES

input period = {WEEK, default MONTH};

AddVerticalLine((period == period.WEEK and GetWeek() <> GetWeek()[1]) or (period == period.MONTH and GetMonth() <> GetMonth()[1]), "", Color.cyan, curve.firm);

#ZIGZAGv2

## START CODE

## ZigZagSign TOMO modification, v0.2 written by Linus @Thinkscripter Lounge adapted from Thinkorswim ZigZagSign Script

input price = close;

input priceH = high; # swing high

input priceL = low; # swing low

input ATRreversalfactor = 10;

def ATR = reference ATR(length = 5);

def reversalAmount = ATRreversalfactor \* ATR;

input showlines = yes;

input displace = 1;

input showBubbleschange = yes;

def barNumber = BarNumber();

def barCount = HighestAll(If(IsNaN(price), 0, barNumber));

rec state = {default init, undefined, uptrend, downtrend};

rec minMaxPrice;

if (GetValue(state, 1) == GetValue(state.init, 0)) {

minMaxPrice = price;

state = state.undefined;

} else if (GetValue(state, 1) == GetValue(state.undefined, 0)) {

if (price <= GetValue(minMaxPrice, 1) - reversalAmount) {

state = state.downtrend;

minMaxPrice = priceL;

} else if (price >= GetValue(minMaxPrice, 1) + reversalAmount) {

state = state.uptrend;

minMaxPrice = priceH;

} else {

state = state.undefined;

minMaxPrice = GetValue(minMaxPrice, 1);

}

} else if (GetValue(state, 1) == GetValue(state.uptrend, 0)) {

if (price <= GetValue(minMaxPrice, 1) - reversalAmount) {

state = state.downtrend;

minMaxPrice = priceL;

} else {

state = state.uptrend;

minMaxPrice = Max(priceH, GetValue(minMaxPrice, 1));

}

} else {

if (price >= GetValue(minMaxPrice, 1) + reversalAmount) {

state = state.uptrend;

minMaxPrice = priceH;

} else {

state = state.downtrend;

minMaxPrice = Min(priceL, GetValue(minMaxPrice, 1));

}

}

def isCalculated = GetValue(state, 0) != GetValue(state, 1) and barNumber >= 1;

def futureDepth = barCount - barNumber;

def tmpLastPeriodBar;

if (isCalculated) {

if (futureDepth >= 1 and GetValue(state, 0) == GetValue(state, -1)) {

tmpLastPeriodBar = fold lastPeriodBarI = 2 to futureDepth + 1 with lastPeriodBarAcc = 1

while lastPeriodBarAcc > 0

do if (GetValue(state, 0) != GetValue(state, -lastPeriodBarI))

then -lastPeriodBarAcc

else lastPeriodBarAcc + 1;

} else {

tmpLastPeriodBar = 0;

}

} else {

tmpLastPeriodBar = Double.NaN;

}

def lastPeriodBar = if (!IsNaN(tmpLastPeriodBar)) then -AbsValue(tmpLastPeriodBar) else -futureDepth;

rec currentPriceLevel;

rec currentPoints;

if (state == state.uptrend and isCalculated) {

currentPriceLevel =

fold barWithMaxOnPeriodI = lastPeriodBar to 1 with barWithMaxOnPeriodAcc = minMaxPrice

do Max(barWithMaxOnPeriodAcc, GetValue(minMaxPrice, barWithMaxOnPeriodI));

currentPoints =

fold maxPointOnPeriodI = lastPeriodBar to 1 with maxPointOnPeriodAcc = Double.NaN

while IsNaN(maxPointOnPeriodAcc)

do if (GetValue(priceH, maxPointOnPeriodI) == currentPriceLevel)

then maxPointOnPeriodI

else maxPointOnPeriodAcc;

} else if (state == state.downtrend and isCalculated) {

currentPriceLevel =

fold barWithMinOnPeriodI = lastPeriodBar to 1 with barWithMinOnPeriodAcc = minMaxPrice

do Min(barWithMinOnPeriodAcc, GetValue(minMaxPrice, barWithMinOnPeriodI));

currentPoints =

fold minPointOnPeriodI = lastPeriodBar to 1 with minPointOnPeriodAcc = Double.NaN

while IsNaN(minPointOnPeriodAcc)

do if (GetValue(priceL, minPointOnPeriodI) == currentPriceLevel)

then minPointOnPeriodI

else minPointOnPeriodAcc;

} else if (!isCalculated and (state == state.uptrend or state == state.downtrend)) {

currentPriceLevel = GetValue(currentPriceLevel, 1);

currentPoints = GetValue(currentPoints, 1) + 1;

} else {

currentPoints = 1;

currentPriceLevel = GetValue(price, currentPoints);

}

plot "ZZ$" = if (barNumber == barCount or barNumber == 1) then if state == state.uptrend then priceH else priceL else if (currentPoints == 0) then currentPriceLevel else Double.NaN;

rec zzSave = if !IsNaN("ZZ$" ) then if (barNumber == barCount or barNumber == 1) then if IsNaN(barNumber[-1]) and state == state.uptrend then priceH else priceL else currentPriceLevel else GetValue(zzSave, 1);

def chg = (if barNumber == barCount and currentPoints < 0 then priceH else if barNumber == barCount and currentPoints > 0 then priceL else currentPriceLevel) - GetValue(zzSave, 1);

def isUp = chg >= 0;

#Higher/Lower/Equal High, Higher/Lower/Equal Low

def xxhigh = if zzSave == priceH then Round(high, 2) else Round(xxhigh[1], 2);

def chghigh = Round(Round(high, 2) - Round(xxhigh[1], 2), 2);

def xxlow = if zzSave == priceL then Round(low, 2) else Round(xxlow[1], 2);

def chglow = Round(Round(low, 2) - Round(xxlow[1], 2), 2);

rec isConf = AbsValue(chg) >= reversalAmount or (IsNaN(GetValue("ZZ$", 1)) and GetValue(isConf, 1));

"ZZ$".EnableApproximation();

"ZZ$".DefineColor("Up Trend", Color.GRAY);

"ZZ$".DefineColor("Down Trend", Color.GRAY);

"ZZ$".DefineColor("Undefined", Color.GRAY);

"ZZ$".AssignValueColor(if !isConf then "ZZ$".Color("Undefined" ) else if isUp then "ZZ$".Color("Up Trend" ) else "ZZ$".Color("Down Trend" ));

DefineGlobalColor("Unconfirmed", Color.dark\_gray);

DefineGlobalColor("Up", Color.RED);

DefineGlobalColor("Down", Color.GREEN);

AddChartBubble(showBubbleschange and !IsNaN("ZZ$" ) and barNumber != 1, if isUp then high else low , Round(chg, 2) , if barCount == barNumber or !isConf then GlobalColor("Unconfirmed" ) else if isUp then GlobalColor("up" ) else GlobalColor("down" ), isUp);

#MACDV2 STRAT

input fastLength = 30;

input slowLength = 50;

input macdLength = 90;

input averageType = AverageType.EXPONENTIAL;

def diff = reference MACD(fastLength, slowLength, macdLength, averageType).Diff;

def value = MovingAverage(averageType, close, fastLength) - MovingAverage(averageType, close, slowLength);

def Avg = MovingAverage(averageType, Value, MACDLength);

AddOrder(OrderType.BUY\_AUTO, value crosses above avg and value[1] <= avg[1] , tickColor = GetColor(0), arrowColor = GetColor(0), name = "LE");

AddOrder(OrderType.SELL\_AUTO, value crosses below avg and value[1] >= avg[1], tickColor = GetColor(1), arrowColor = GetColor(1), name = "SE");

#CLOCK

declare once\_per\_bar;

input time = 0830;

AddVerticalLine(secondsFromTime(time)[1] < 0 && secondsFromTime(time) >= 0, time,color.dark\_gray);

declare once\_per\_bar;

input time2 = 300;

AddVerticalLine(secondsFromTime(time2)[1] < 0 && secondsFromTime(time2) >= 0, time2,color.dark\_gray);

declare once\_per\_bar;

input time3 = 1600;

AddVerticalLine(secondsFromTime(time3)[1] < 0 && secondsFromTime(time3) >= 0, time3,color.dark\_gray);

input period = {WEEK, default MONTH};

AddVerticalLine((period == period.WEEK and GetWeek() <> GetWeek()[1]) or (period == period.MONTH and GetMonth() <> GetMonth()[1]), "", Color.gray, curve.firm);

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_