```
//@version=5
VERSION                  = 'v1_0_6'// 2024.06.04
strategy(
  'CCI + T3 + RedK + VWAP',
  shorttitle             = 'CCI + T3 + RedK + VWAP' + VERSION,
  overlay                = false,
  explicit_plot_zorder   = true,
  initial_capital        = 10000,
  default_qty_type       = strategy.percent_of_equity,
  default_qty_value      = 1)

// Project #860
// Original code:"CCI" framework by Bullit-in script library
// and  "T3 Volatility Quality Index (VQI) w/ DSL & Pips Filtering" by loxx
// and "Directional Volume EStimate from Price Action" by RedKTrader aka D_VESPA
// and "VWAP" framework by Bullit-in script library
// Actual version: @gu5tavo71 for Tradingfy (@Tradingfy by Twitter)
// This script reuses open source code from another authors:
// @PineCoders, Built-in Library, and Community Scripts
// Disclaimer: I am not a financial advisor.
//             For purpose educate only. Use at your own risk.

G_SCRIPT01   = '■ ' + 'CCI'
i_script01   = input.bool       (true,      G_SCRIPT01 + ' On/Off'          , group = G_SCRIPT01)
//#region ———— <↓↓↓ G_SCRIPT01 ↓↓↓> {
//@version=5
// indicator(title="Commodity Channel Index", shorttitle="CCI", format=format.price, precision=2,
timeframe="", timeframe_gaps=true)
cciLen = input.int(20, minval=1)
src = input(hlc3, title="Source")
ma = ta.sma(src, cciLen)
cci = (src - ma) / (0.015 * ta.dev(src, cciLen))
// plot(cci, "CCI", color=#2962FF)
// band1 = hline(100, "Upper Band", color=#787B86, linestyle=hline.style_dashed)
// hline(0, "Middle Band", color=color.new(#787B86, 50))
// band0 = hline(-100, "Lower Band", color=#787B86, linestyle=hline.style_dashed)
// fill(band1, band0, color=color.rgb(33, 150, 243, 90), title="Background")
```

```
ma(source, cciLen, type) =>
    switch type
        "SMA" => ta.sma(source, cciLen)
        "EMA" => ta.ema(source, cciLen)
        "SMMA (RMA)" => ta.rma(source, cciLen)
        "WMA" => ta.wma(source, cciLen)
        "VWMA" => ta.vwma(source, cciLen)

typeMA = input.string(title = "Method", defval = "SMA", options=["SMA", "EMA", "SMMA (RMA)",
"WMA", "VWMA"], group="Smoothing")
smoothingLength = input.int(title = "Length", defval = 5, minval = 1, maxval = 100,
group="Smoothing")

smoothingLine = ma(cci, smoothingLength, typeMA)
// plot(smoothingLine, title="Smoothing Line", color=#f37f20, display=display.none)

// plot (
  // cci,
  // title      = 'cci',
  // linewidth  = 2,
  // color      = cciCol,
  // style      = plot.style_line)
//#endregion }
// ————————————————— <↑↑↑ G_SCRIPT01 ↑↑↑>


G_SCRIPT02   = '■ ' + 'T3'
i_script02   = input.bool       (true,      G_SCRIPT02 + ' On/Off'          , group = G_SCRIPT02)
//#region ———————— <↓↓↓ G_SCRIPT02 ↓↓↓> {
// This source code is subject to the terms of the Mozilla Public License 2.0 at
https://mozilla.org/MPL/2.0/
// © loxx

// //@version=5
// indicator("T3 Volatility Quality Index (VQI) w/ DSL & Pips Filtering [Loxx]",
     // shorttitle="T3VQIDSLPF [Loxx]",
     // overlay = false,
     // timeframe="",
     // timeframe_gaps = true)

import loxx/loxxmas/1

greencolor = #2DD204
```

```
redcolor = #D2042D

darkGreenColor =  #1B7E02
darkRedColor = #93021F

_iT3(src, per, hot, clean)=>
    a = hot
    _c1 = -a * a * a
    _c2 = 3 * a * a + 3 * a * a * a
    _c3 = -6 * a * a - 3 * a - 3 * a * a * a
    _c4 = 1 + 3 * a + a * a * a + 3 * a * a

    alpha = 0.

    if (clean == "T3 New")
        alpha := 2.0 / (2.0 + (per - 1.0) / 2.0)
    else
        alpha := 2.0 / (1.0 + per)

    _t30 = src, _t31 = src
    _t32 = src, _t33 = src
    _t34 = src, _t35 = src

    _t30 := nz(_t30[1]) + alpha * (src -  nz(_t30[1]))
    _t31 := nz(_t31[1]) + alpha * (_t30 - nz(_t31[1]))
    _t32 := nz(_t32[1]) + alpha * (_t31 - nz(_t32[1]))
    _t33 := nz(_t33[1]) + alpha * (_t32 - nz(_t33[1]))
    _t34 := nz(_t34[1]) + alpha * (_t33 - nz(_t34[1]))
    _t35 := nz(_t35[1]) + alpha * (_t34 - nz(_t35[1]))
    out =
        _c1 * _t35 + _c2 * _t34 +
        _c3 * _t33 +  _c4 * _t32
    out

_declen()=>
    mtckstr = str.tostring(syminfo.mintick)
    da = str.split(mtckstr, ".")
    temp = array.size(da)
    dlen = 0.
    if syminfo.mintick < 1
        dstr = array.get(da, 1)
        dlen := str.length(dstr)
    dlen
```

```
variant(type, src, len) =>
    sig = 0.0
    trig = 0.0
    special = false
    if type == "Exponential Moving Average - EMA"
        [t, s, b] = loxxmas.ema(src, len)
        sig := s
        trig := t
        special := b
    else if type == "Fast Exponential Moving Average - FEMA"
        [t, s, b] = loxxmas.fema(src, len)
        sig := s
        trig := t
        special := b
    trig

PriceSmoothing = input.int(5, "Source Smoothing Period", group= "Basic Settings")
t3hot = input.float(.5, "T3 Hot", group= "Basic Settings")
t3swt = input.string("T3 New", "T3 Type", options = ["T3 New", "T3 Original"], group = "Basic
Settings")

FilterInPips = input.float(1.9, "Filter in Pips", group= "Basic Settings")

sigmatype = input.string("Exponential Moving Average - EMA", "Signal/DSL Smoothing", options
= ["Exponential Moving Average - EMA", "Fast Exponential Moving Average - FEMA"], group =
"Signal/DSL Settings")
Ma1Period = input.int(9, "DSL Period", group= "Signal/DSL Settings")

colorbars = input.bool(true, "Color bars?", group = "UI Options")
showSigs = input.bool(true, "Show signals?", group = "UI Options")

pipMultiplier = math.pow(10, _declen() % 2)

cHigh = _iT3(high, PriceSmoothing, t3hot, t3swt)
cLow = _iT3(low, PriceSmoothing, t3hot, t3swt)
cOpen = _iT3(open, PriceSmoothing, t3hot, t3swt)
cClose = _iT3(close, PriceSmoothing, t3hot, t3swt)
pClose =  _iT3(nz(close[1]), PriceSmoothing, t3hot, t3swt)

val = 0., valc = 0.
truerng = math.max(cHigh, pClose) - math.min(cLow, pClose)
rng = cHigh - cLow
```

```
vqi = (rng != 0 and truerng != 0) ? ((cClose - pClose) / truerng + (cClose - cOpen) / rng) * 0.5 :
val[1]

val := nz(val[1]) + math.abs(vqi) * (cClose - pClose + cClose - cOpen) * 0.5
if (FilterInPips > 0)
    if (math.abs(val - val[1]) < FilterInPips * pipMultiplier * syminfo.mintick)
        val := nz(val[1])

sig = nz(val[1])

temp = variant(sigmatype, val, Ma1Period)
levelu = 0., leveld = 0., mid = 0.
levelu := (val > sig) ? temp : nz(levelu[1])
leveld := (val < sig) ? temp : nz(leveld[1])

// colorout = val > levelu ? greencolor : val < leveld ? redcolor : color.gray

// plot(val, "VQI", color = colorout, linewidth = 3)

// plot(levelu, "Level Up", color = darkGreenColor)
// plot(leveld, "Level Down", color = darkRedColor)

goLong = ta.crossover(val, levelu)
goShort = ta.crossunder(val, leveld)

// plotshape(showSigs and goLong, title = "Long", color = color.yellow, textcolor = color.yellow,
text = "L", style = shape.triangleup, location = location.bottom, size = size.auto)
// plotshape(showSigs and goShort, title = "Short", color = color.fuchsia, textcolor =
color.fuchsia, text = "S", style = shape.triangledown, location = location.top, size = size.auto)

// alertcondition(goLong, title = "High Volatility", message = "T3 Volatility Quality Index (VQI) w/
DSL & Pips Filtering [Loxx]: Uptrend\nSymbol: {{ticker}}\nPrice: {{close}}")
// alertcondition(goShort, title = "Low Volatility", message = "T3 Volatility Quality Index (VQI) w/
DSL & Pips Filtering [Loxx]: Downtrend\nSymbol: {{ticker}}\nPrice: {{close}}")

// barcolor(colorbars ? colorout : na)
//#endregion }
// ———————————— <↑↑↑ G_SCRIPT02 ↑↑↑>


G_SCRIPT03   = '■ ' + 'RedK D_VESPA'
i_script03   = input.bool       (true,      G_SCRIPT03 + ' On/Off'          , group = G_SCRIPT03)
//#region ———— <↓↓↓ G_SCRIPT03 ↓↓↓> {
```

```
// This source code is subject to the terms of the Mozilla Public License 2.0 at
https://mozilla.org/MPL/2.0/
// © RedKTrader - May 2023

// //@version=5
// indicator(title='Directional Volume Estimate from Pirce Action v1.0', shorttitle='RedK D_VESPA
v1.0',
  // format=format.volume, timeframe='', timeframe_gaps=false, explicit_plot_zorder = true)

// Originally this was based on V.Viewer - it estimates and plots average supply & demand
volume based on price action
// it uses the shape of a price bar to estimate the supply vs demand split of the traded volume
// & plots a moving average of both, and an estimated Average "Net Volume"
// This provides an insightful way to look at the traded volume compared to the classic volume
histogram view
// --------------------------------
//
// - Supply/Demand calc now includes 2-bar gaps (improved algo)
// - Add option for MA type for calculation
// - Add option to show Net Volume as 3-color bars
// - Visual simplification and improvements to be less distracting & more actionable
// - options to display/hide main visuals while maintaining the status line consistency (Avg
Supply, Avg Demand, Avg Net)
// - add alerts for NetVol moving into Buy (crosses 0 up) or Sell (crosses 0 down) modes
// - implement a "sentiment" timeframe - options 1W, 1D, 1H
// using request.scurity() calls to bring in the data from the HTF and pass to the Calc_VESPA
function

//
============================================================================
=======================
GetAverage(_data, _len, MAOption) =>
    value = switch MAOption
        'SMA' => ta.sma(_data, _len)
        'EMA' => ta.ema(_data, _len)
        'RMA' => ta.rma(_data, _len)
        =>
            ta.wma(_data, _len)

//
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
Calc_VESPA(bool VolumeOn, bool GapsOn) =>
```

```
    //
*****************************************************************************************************************
********
    // Extrapolate avg estimated Buy & Sell volume per bar
    // How the updated buy/sell average estimated volume algo works:
    // buy volume is assoicated with price up-moves, sell volume is associated with price
down-moves
    // so each of the bulls and bears will get the equivalent of the top & bottom wicks,
    // for up bars, bulls get the value of the "body", else the bears get the "body"
    // open gaps are allocated to bulls or bears depending on the gap direction (if the option is
selected)
    // if there's no volume, then this will just reflect the price action split between buyers/sellers
    //
*****************************************************************************************************************
********
    o = open
    c = close
    h = high
    l = low
    v = na(volume) or not VolumeOn ? 1 : volume

    gap        = o - c[1]

    bull_gap   = math.max(gap, 0)
    bear_gap   = math.abs(math.min(gap, 0))

    body       = math.abs(c - o)
    BarRange   = h - l
    wick       = BarRange - body

    up_bar     = c > o

    bull       = wick + (up_bar ? body : 0) + (GapsOn ? bull_gap : 0)
    bear       = wick + (up_bar ? 0 : body) + (GapsOn ? bear_gap : 0)

    ViRange    = bull + bear

    // Rare cases with very low TF's (mainly FOREX) where no price movement occurs, ViRange
(including gaps) = 0
    BScore     = ViRange > 0 ? bull / ViRange : 0.5
    BuyVol     = BScore * v
    SellVol    = v - BuyVol
```

```
  // Return Estimated Buy & Sell Volume values
    [BuyVol, SellVol]

//
**************************************************************************************************************
********
// inputs
//
**************************************************************************************************************
********
length      = input.int(title='Volume Length',    defval=16,  minval=1)
AvgType     = input.string("WMA", "Average type", options = ['SMA', 'EMA', 'RMA', 'WMA'])

smooth      = input.int(title='Smoothing',        defval=8,   minval=1)

VolumeWeighted  = input.bool(true, "Volume Weighted (Keep on)")
GapImpact       = input.bool(true, "2-bar Gap Impact (Keep on)")

ShowNetVolBars  = input.bool(true, "Show NetVol Bars")
ShowNetVolHisto = input.bool(true, "Show NetVol Histogram")

//
**************************************************************************************************************
********
// variables
//
**************************************************************************************************************
********

// Calculate estimated Buy & Sell colume
[B_BuyVol, B_SellVol] = Calc_VESPA(VolumeWeighted, GapImpact)

// Calc average Buy & Sell vol and NetVol from estimate
demand      = ta.wma(GetAverage(B_BuyVol, length, AvgType), smooth)
supply      = ta.wma(GetAverage(B_SellVol, length, AvgType), smooth)
NetVol      = demand - supply

//
**************************************************************************************************************
********
// Plots  -- classic volume bars have been removed
//
**************************************************************************************************************
```

****

```
col_red         = color.new(#ff0000, 00)
col_green       = color.new(#00ff00, 00)
col_hist_red    = color.new(#ef5350, 25)
col_hist_green  = color.new(#089981, 25)

col_gold        = color.new(#ffeb3b, 20)

//plot(v,         title='Volume',         style=plot.style_columns,   color=up_bar ? col_green : col_red,
display=display.none)


//
================================================================================
================================================
// NetVol Bars Plot
//
================================================================================
================================================
nvo = fixnan(supply)                        // fixes NaN values - observed mainly on Renko
nvc = fixnan(demand)
nvh = math.max(nvo, nvc)
nvl = math.min(nvo, nvc)

rising      = ta.change(NetVol) > 0


c_barup     = color.new(#11ff20, 60)
c_bardn     = color.new(#ff1111, 60)
c_bardj     = color.new(#ffffff, 50)

c_barupb    = color.new(#1b5e20, 50)
c_bardnb    = color.new(#981919, 50)
c_bardjb    = color.new(#9598a1, 50)

// barcolor    = nvc > nvo and rising ? c_barup : nvc < nvo and not rising ? c_bardn : c_bardj
borcolor    = nvc > nvo and rising ? c_barupb : nvc < nvo and not rising ? c_bardnb : c_bardjb

// plotcandle(nvo, nvh, nvl, nvc, 'NetVol Bars', barcolor, barcolor, bordercolor = borcolor,
  // display = ShowNetVolBars ? display.pane : display.none)

// hline(0,        title='zero line', linestyle=hline.style_dotted,    color=col_gold,
```

```
  // display = ShowNetVolHisto ? display.all : display.none)  //hides with histogram

// plot(supply,    title='Supply',         color=col_red,       linewidth=2)
// plot(demand,    title='Demand',         color=col_green,     linewidth=2)

//
================================================================================
================================================
// Net Volume Histogram Plot
//
================================================================================
================================================
// c_NetVol = NetVol >= 0 ? col_hist_green : col_hist_red
// plot(NetVol,    title='NetVol Histogram',  style=plot.style_columns,      color = c_NetVol,
linewidth = 4,
  // display = ShowNetVolHisto ? display.all : display.status_line + display.data_window)


//
================================================================================
================================================
// Secondary TF Average Net Volume Plot
//
================================================================================
================================================

S_gp = "Secondary TF"

ShowS_NetVol  = input.bool(false, 'Show Secondary', group = S_gp, inline = "Senti_TF")    //
STF plot hidden by default

i_STF         = input.string("Chart", 'TF', options = ['Chart', '1Wk', '1Day' , '1Hr'], group = S_gp,
inline = "Senti_TF")
S_length      = input.int(12, "Length", minval = 1, group = S_gp, inline = "Senti_L")
S_smooth      = input.int(4, "Smooth", minval = 1, group = S_gp, inline = "Senti_L")

STF = switch i_STF
   '1Wk'   => 'W'
   '1Day'  => 'D'
   '1Hr'   => '60'
   =>
      timeframe.period
```

```
// Error trap here if selected secondary TF is lower than chart
float chartTF_Mins = timeframe.in_seconds() / 60
float i_STF_Mins  = timeframe.in_seconds(STF) / 60
if chartTF_Mins > i_STF_Mins and ShowS_NetVol
    runtime.error("Secondary timeframe must be equal to, or higher than, the chart's timeframe.")


[S_BuyVol, S_SellVol] = request.security(syminfo.tickerid, STF, Calc_VESPA(VolumeWeighted,
GapImpact) )

S_demand      = request.security(syminfo.tickerid, STF, ta.wma(GetAverage(S_BuyVol,
S_length, AvgType), S_smooth))
S_supply      = request.security(syminfo.tickerid, STF, ta.wma(GetAverage(S_SellVol, S_length,
AvgType), S_smooth))

S_NetVol      = S_demand - S_supply

c_S_NetVol = S_NetVol >= 0 ? color.aqua : color.orange
// plot(S_NetVol,    title='Secondary TF NetVol',  color = c_S_NetVol, linewidth = 2,
  // display = ShowS_NetVol ? display.all : display.data_window)

// plot(S_supply, color = color.red)
// plot(S_demand, color = color.green)

//
================================================================================
================================================
// Alerts
//
================================================================================
================================================

// Primary TF Alerts
Al_NetVol_up    = ta.crossover(NetVol, 0)
Al_NetVol_dn    = ta.crossunder(NetVol, 0)
Al_NetVol_swing = Al_NetVol_up or Al_NetVol_dn

// alertcondition(Al_NetVol_up,    "A1.  Avg NetVol Crossing 0 Up",      "Avg NetVol Positive -
Bullish Mode Detected!")
// alertcondition(Al_NetVol_dn,    "A2.  Avg NetVol Crossing 0 Down",    "Avg NetVol Negative -
Bearish Mode Detected!")
// alertcondition(Al_NetVol_swing, "A3.  Avg NetVol Crossing 0",         "Avg NetVol Swing -
Possible Mode Reversal Detected!")
```

```
// Secondary TF Alerts
Al_SNetVol_up    = ta.crossover(S_NetVol, 0)
Al_SNetVol_dn    = ta.crossunder(S_NetVol, 0)
Al_SNetVol_swing = Al_SNetVol_up or Al_SNetVol_dn

// alertcondition(Al_SNetVol_up,    "B1.  STF Avg NetVol Crossing 0 Up",      "Secondary TF Avg NetVol Positive - Bullish Mode Detected!")
// alertcondition(Al_SNetVol_dn,    "B2.  STF Avg NetVol Crossing 0 Down",    "Secondary TF Avg NetVol Negative - Bearish Mode Detected!")
// alertcondition(Al_SNetVol_swing, "B3.  STF Avg NetVol Crossing 0",         "Secondary TF Avg NetVol Swing - Possible Mode Reversal Detected!")



//#endregion }
// ———————————————— <↑↑↑ G_SCRIPT03 ↑↑↑>


G_SCRIPT04   = '■ ' + 'VWAP'
i_script04   = input.bool       (true,      G_SCRIPT04 + ' On/Off'          , group = G_SCRIPT04)
//#region ————— <↓↓↓ G_SCRIPT04 ↓↓↓> {
// //@version=5
// indicator(title="Volume Weighted Average Price", shorttitle="VWAP", overlay=true, timeframe="", timeframe_gaps=true)

hideonDWM = input(false, title="Hide VWAP on 1D or Above", group="VWAP Settings", display = display.data_window)
var anchor = input.string(defval = "Session", title="Anchor Period",
 options=["Session", "Week", "Month", "Quarter", "Year", "Decade", "Century", "Earnings", "Dividends", "Splits"], group="VWAP Settings")
vwapSrc = input(title = "Source", defval = hlc3, group="VWAP Settings", display = display.data_window)
offset = input.int(0, title="Offset", group="VWAP Settings", minval=0, display = display.data_window)

BANDS_GROUP = "Bands Settings"
CALC_MODE_TOOLTIP = "Determines the units used to calculate the distance of the bands. When 'Percentage' is selected, a multiplier of 1 means 1%."
calcModeInput = input.string("Standard Deviation", "Bands Calculation Mode", options = ["Standard Deviation", "Percentage"], group = BANDS_GROUP, tooltip = CALC_MODE_TOOLTIP, display = display.data_window)
showBand_1 = input(true, title = "", group = BANDS_GROUP, inline = "band_1", display =
```

```
display.data_window)
bandMult_1 = input.float(1.0, title = "Bands Multiplier #1", group = BANDS_GROUP, inline =
"band_1", step = 0.5, minval=0, display = display.data_window)
showBand_2 = input(false, title = "", group = BANDS_GROUP, inline = "band_2", display =
display.data_window)
bandMult_2 = input.float(2.0, title = "Bands Multiplier #2", group = BANDS_GROUP, inline =
"band_2", step = 0.5, minval=0, display = display.data_window)
showBand_3 = input(false, title = "", group = BANDS_GROUP, inline = "band_3", display =
display.data_window)
bandMult_3 = input.float(3.0, title = "Bands Multiplier #3", group = BANDS_GROUP, inline =
"band_3", step = 0.5, minval=0, display = display.data_window)

if ta.cum(volume) == 0
    runtime.error("No volume is provided by the data vendor.")

new_earnings = request.earnings(syminfo.tickerid, earnings.actual, barmerge.gaps_on,
barmerge.lookahead_on, ignore_invalid_symbol=true)
new_dividends = request.dividends(syminfo.tickerid, dividends.gross, barmerge.gaps_on,
barmerge.lookahead_on, ignore_invalid_symbol=true)
new_split = request.splits(syminfo.tickerid, splits.denominator, barmerge.gaps_on,
barmerge.lookahead_on, ignore_invalid_symbol=true)

isNewPeriod = switch anchor
        "Earnings"  => not na(new_earnings)
        "Dividends" => not na(new_dividends)
        "Splits"    => not na(new_split)
        "Session"   => timeframe.change("D")
        "Week"      => timeframe.change("W")
        "Month"     => timeframe.change("M")
        "Quarter"   => timeframe.change("3M")
        "Year"      => timeframe.change("12M")
        "Decade"    => timeframe.change("12M") and year % 10 == 0
        "Century"   => timeframe.change("12M") and year % 100 == 0
        => false

isEsdAnchor = anchor == "Earnings" or anchor == "Dividends" or anchor == "Splits"
if na(vwapSrc[1]) and not isEsdAnchor
        isNewPeriod := true

float vwapValue = na
float upperBandValue1 = na
float lowerBandValue1 = na
float upperBandValue2 = na
```

```
float lowerBandValue2 = na
float upperBandValue3 = na
float lowerBandValue3 = na

if not (hideonDWM and timeframe.isdwm)
    [_vwap, _stdevUpper, _] = ta.vwap(vwapSrc, isNewPeriod, 1)
        vwapValue := _vwap
    stdevAbs = _stdevUpper - _vwap
        bandBasis = calcModeInput == "Standard Deviation" ? stdevAbs : _vwap * 0.01
        upperBandValue1 := _vwap + bandBasis * bandMult_1
        lowerBandValue1 := _vwap - bandBasis * bandMult_1
        upperBandValue2 := _vwap + bandBasis * bandMult_2
        lowerBandValue2 := _vwap - bandBasis * bandMult_2
        upperBandValue3 := _vwap + bandBasis * bandMult_3
        lowerBandValue3 := _vwap - bandBasis * bandMult_3

// plot(vwapValue, title="VWAP", color=#2962FF, offset=offset)

// upperBand_1 = plot(upperBandValue1, title="Upper Band #1", color=color.green, offset=offset,
display = showBand_1 ? display.all : display.none)
// lowerBand_1 = plot(lowerBandValue1, title="Lower Band #1", color=color.green, offset=offset,
display = showBand_1 ? display.all : display.none)
// fill(upperBand_1, lowerBand_1, title="Bands Fill #1", color= color.new(color.green, 95)    ,
display = showBand_1 ? display.all : display.none)

// upperBand_2 = plot(upperBandValue2, title="Upper Band #2", color=color.olive, offset=offset,
display = showBand_2 ? display.all : display.none)
// lowerBand_2 = plot(lowerBandValue2, title="Lower Band #2", color=color.olive, offset=offset,
display = showBand_2 ? display.all : display.none)
// fill(upperBand_2, lowerBand_2, title="Bands Fill #2", color= color.new(color.olive, 95)    ,
display = showBand_2 ? display.all : display.none)

// upperBand_3 = plot(upperBandValue3, title="Upper Band #3", color=color.teal, offset=offset,
display = showBand_3 ? display.all : display.none)
// lowerBand_3 = plot(lowerBandValue3, title="Lower Band #3", color=color.teal, offset=offset,
display = showBand_3 ? display.all : display.none)
// fill(upperBand_3, lowerBand_3, title="Bands Fill #3", color= color.new(color.teal, 95)    ,
display = showBand_3 ? display.all : display.none)

//#endregion }
// ———————————————— <↑↑↑ G_SCRIPT04 ↑↑↑>

G_SCRIPT     = '■ ' + 'CCI + T3 + RedK + VWAP'
```

```
// ———————————————— <function_declarations>
// @function               Label for HeatMap
// @param   _txt           (string) You txt
// @param   _y             (int) Level for label
// @returns                (label) Label w/you text
f_labelPrint(_txt, _y)=>
    var label lh = label.new(
      x = bar_index,
      y = na,
      text = '',
      textalign = text.align_right,
      textcolor = color.gray,
      color = color.new(color.black, 100),
      style = label.style_label_left,
      size = size.normal)
    label.set_text(lh, _txt)
    label.set_xy(lh, bar_index + 1, _y)
    label.delete(na)

// ———————————————— <calculations>
ccDir        = cci > 0 ? 1 : -1
t3Dir        = val > levelu ? 1 : val < leveld ? -1 : 0
dvespaDir    = NetVol >= 0 ? 1 : -1
vwapDir      = vwapValue < close ? 1 : -1

//<set initial values>
var marketPosition = 0.0

//<triggers>
leTrigger1      = i_script01 ? ccDir     ==  1 : true
leTrigger2      = i_script02 ? t3Dir     ==  1 : true
leTrigger3      = i_script03 ? dvespaDir ==  1 : true
leTrigger4      = i_script04 ? vwapDir   ==  1 : true
leTrigger       = leTrigger1 and leTrigger2 and leTrigger3 and leTrigger4
seTrigger1      = i_script01 ? ccDir     == -1 : true
seTrigger2      = i_script02 ? t3Dir     == -1 : true
seTrigger3      = i_script03 ? dvespaDir == -1 : true
seTrigger4      = i_script04 ? vwapDir   == -1 : true
seTrigger       = seTrigger1 and seTrigger2 and seTrigger3 and seTrigger4
lxTrigger       = false
sxTrigger       = false

//<marketPosition>
```

```
// marketPosition is a numerical variable:1 for Long, -1 for Short and 0 for eXit
switch
    lxTrigger and marketPosition[1] ==  1  => marketPosition :=  0
    sxTrigger and marketPosition[1] == -1  => marketPosition :=  0
    leTrigger and marketPosition[1] <=  0  => marketPosition :=  1
    seTrigger and marketPosition[1] >=  0  => marketPosition := -1
longX       = lxTrigger and marketPosition[1] ==  1 and marketPosition ==  0
shortX      = sxTrigger and marketPosition[1] == -1 and marketPosition ==  0
longE       = leTrigger and marketPosition[1] <=  0 and marketPosition ==  1
shortE      = seTrigger and marketPosition[1] >=  0 and marketPosition == -1

//<color>
cciCol      = i_script01 ?
              ccDir ==  1 ? color.new(color.green, 0) :
              ccDir == -1 ? color.new(color.red,   0) : na
            : color.new(color.gray, 50)
t3Col       = i_script02 ?
              t3Dir ==  1 ? color.new(#2DD204, 0) :
              t3Dir == -1 ? color.new(#D2042D, 0) : na
            : color.new(color.gray, 50)
dvespaCol   = i_script03 ?
              dvespaDir ==  1 ? color.new(#089981, 25) :
              dvespaDir == -1 ? color.new(#ef5350, 25) : na
            : color.new(color.gray, 50)
vwapCol     = i_script04 ?
              vwapDir ==  1 ? color.new(color.green, 0) :
              vwapDir == -1 ? color.new(color.red,   0) : na
            : color.new(color.gray, 50)

// ———————————————— <strategy_calls>
//<long orders>
if longE
    strategy.entry(
      'Long',
      strategy.long,
      comment       = 'LE')
if strategy.position_size > 0
    strategy.exit(
      id            = 'Long Exit',
      from_entry    = 'Long',
      stop          = na,
      limit         = na)
if longX
```

```
    strategy.close(
      'Long',
      comment        = 'LX')
//<short orders>
if shortE
    strategy.entry(
      'Short',
      strategy.short,
      comment        = 'SE')
if strategy.position_size < 0
    strategy.exit(
      id             = 'Short Exit',
      from_entry     = 'Short',
      stop           = na,
      limit          = na)
if shortX
    strategy.close(
      'Short',
      comment        = 'SX')

// ———————————————— <visuals>
plotchar(
  longE      ? 5 : na,
  title      = 'Long',
  color      = color.new(color.green, 0),
  char       = '▲',
  size       = size.small,
  location   = location.absolute)
plotchar(
  shortE      ? 5 : na,
  title      = 'Short',
  color      = color.new(color.red, 0),
  char       = '▼',
  size       = size.small,
  location   = location.absolute)

f_labelPrint('CCI',  4)
plotchar(
  4,
  title      = 'CCI',
  color      = cciCol,
  char       = '■',
  size       = size.small,
```

```
  location   = location.absolute)

f_labelPrint('T3',   3)
plotchar(
  3,
  title      = 'T3',
  color      = t3Col,
  char       = '■',
  size       = size.small,
  location   = location.absolute)

f_labelPrint('RedK', 2)
plotchar(
  2,
  title      = 'RedK',
  color      = dvespaCol,
  char       = '■',
  size       = size.small,
  location   = location.absolute)

f_labelPrint('VWAP', 1)
plotchar(
  1,
  title      = 'VWAP',
  color      = vwapCol,
  char       = '■',
  size       = size.small,
  location   = location.absolute)


//<debug>
plot(
  na,
  title      = "——— <debug> ———",
  editable   = false,
  display    = display.data_window)
plotchar(ccDir,      'ccDir',     editable = false, display = display.data_window)
plotchar(t3Dir,      't3Dir',     editable = false, display = display.data_window)
plotchar(dvespaDir , 'dvespaDir', editable = false, display = display.data_window)
plotchar(vwapDir,    'vwapDir',   editable = false, display = display.data_window)
```