

// This Pine Script® code is subject to the terms of the Mozilla Public License 2.0 at <https://mozilla.org/MPL/2.0/> MPL-2.0

//@version=6

indicator("MFB - VRS - 5 days", overlay = true, maxBarsBack = 5000, maxBoxesCount = 500, maxLinesCount = 100, maxLabelsCount = 100)

// --- Constants ---

const int MAX_PROFILES = 5

const string DEFAULT_SESSION = "0930-1600"

const string DEFAULT_TIMEZONE = "America/New_York"

COLOR_UP = #26a69a

COLOR_DOWN = #ef5350

COLOR_POC = #9c27b0

COLOR_VA = #6a1b9a

COLOR_MID = #fdd835

COLOR_BG = color.new(#9c27b0, 95)

// --- Inputs ---

sessionTime = input.session(DEFAULT_SESSION, "Structural Session", tooltip = "NY session used to build each completed volume profile.")

timezoneInput = input.string(DEFAULT_TIMEZONE, "Timezone", tooltip = "Timezone used for session detection.")

```
valueAreaPct = input.float(70.0, "Value Area Percentage", minval = 0.0, maxval  
= 100.0, tooltip = "Percentage of total volume included in each value area.") /  
100.0
```

```
rowCount = input.int(50, "Row Resolution", minval = 10, maxval = 50, tooltip =  
"Price rows used by each profile. The 50-row maximum keeps five profiles  
within the drawing-object limit.")
```

```
histWidth = input.int(40, "Histogram Width (Bars)", minval = 5, maxval = 200,  
tooltip = "Maximum horizontal width of each profile histogram, clipped to its  
own NY session.")
```

```
showHistogramsInput = input.bool(true, "Show Histograms", tooltip = "Display  
the five historical volume distributions.")
```

```
showLevelsInput = input.bool(true, "Show VAL / POC / VAH", tooltip = "Display  
fixed value-area levels for each completed session.")
```

```
showVaMidInput = input.bool(true, "Show VA Mid", tooltip = "Display the  
midpoint between VAL and VAH for each completed session.")
```

```
showLabelsInput = input.bool(true, "Show Session Labels", tooltip = "Show D-  
1 through D-5 labels on each completed profile.")
```

```
showBackgroundInput = input.bool(true, "Highlight NY Sessions", tooltip =  
"Highlight the active NY structural session.")
```

```
showDirectionalInput = input.bool(true, "Show 5-Day Direction Box", tooltip =  
"Summarize five completed profiles as bullish, bearish, or mixed using POC  
migration, VA Mid migration, and aggregate up/down volume.")
```

```
directionBoxPositionInput = input.string(position.bottom_right, "Direction Box  
Position", options = [position.top_left, position.top_center, position.top_right,  
position.middle_left, position.middle_center, position.middle_right,  
position.bottom_left, position.bottom_center, position.bottom_right], tooltip  
= "Position of the five-day directional summary box. Bottom-right is the  
default.")
```

```
// --- Style Inputs ---
```

```
vaOpacity = input.int(30, "VA Opacity %", minval = 0, maxval = 100, group =  
"Style", tooltip = "Transparency of histogram rows inside the value area.")
```

```
outOpacity = input.int(75, "Outside VA Opacity %", minval = 0, maxval = 100,  
group = "Style", tooltip = "Transparency of histogram rows outside the value  
area.")
```

```
histThickness = input.int(1, "Histogram Border Width", minval = 0, maxval = 3,  
group = "Style", tooltip = "Border width used for histogram boxes.")
```

```
levelWidth = input.int(2, "Level Width", minval = 1, maxval = 4, group = "Style",  
tooltip = "Width of the VAL, POC, VAH, and VA Mid lines.")
```

```
// --- Session Data Types ---
```

```
type BarInfo
```

```
    float price
```

```
    float volume
```

```
    bool isUp
```

```
type SessionProfile
```

```
    int startBar
```

```
    int endBar
```

```
    float sessionLow
```

```
    float sessionHigh
```

```
    float priceStep
```

```
    float[] upVolumes
```

float[] downVolumes

float[] totalVolumes

float maxVolume

int lowerIndex

int upperIndex

float pocPrice

float vahPrice

float valPrice

float vaMidPrice

bool isValid

// --- Functions ---

buildProfile(BarInfo[] sourceData, int profileStartBar, int profileEndBar, float profileHigh, float profileLow, int rows, float percentage) =>

SessionProfile profile = SessionProfile.new()

profile.startBar := profileStartBar

profile.endBar := profileEndBar

profile.sessionLow := profileLow

profile.sessionHigh := profileHigh

profile.isValid := false

float priceRange = profileHigh - profileLow

if priceRange > 0.0 and array.size(sourceData) > 0

```

float step = priceRange / rows

float[] upVolumes = array.new_float(rows, 0.0)
float[] downVolumes = array.new_float(rows, 0.0)
float[] totalVolumes = array.new_float(rows, 0.0)
float totalVolume = 0.0

for profileBar in sourceData
    int rowIndex = int(math.min(rows - 1, math.floor((profileBar.price -
profileLow) / step)))
    if rowIndex >= 0 and rowIndex < rows
        float rowVolume = nz(profileBar.volume, 0.0)
        if profileBar.isUp
            array.set(upVolumes, rowIndex, array.get(upVolumes, rowIndex) +
rowVolume)
        else
            array.set(downVolumes, rowIndex, array.get(downVolumes,
rowIndex) + rowVolume)
            array.set(totalVolumes, rowIndex, array.get(totalVolumes, rowIndex) +
rowVolume)
        totalVolume += rowVolume

float maximumVolume = array.max(totalVolumes)
if maximumVolume > 0.0 and totalVolume > 0.0
    int pocIndex = array.indexof(totalVolumes, maximumVolume)

```

```
float targetVolume = totalVolume * percentage
```

```
float valueAreaVolume = maximumVolume
```

```
int upperIndex = pocIndex
```

```
int lowerIndex = pocIndex
```

```
int expansionSteps = 0
```

```
while valueAreaVolume < targetVolume and (upperIndex < rows - 1 or  
lowerIndex > 0) and expansionSteps < rows
```

```
    expansionSteps += 1
```

```
    bool canExpandUp = upperIndex < rows - 1
```

```
    bool canExpandDown = lowerIndex > 0
```

```
    float upperVolume = canExpandUp ? array.get(totalVolumes,  
upperIndex + 1) : 0.0
```

```
    float lowerVolume = canExpandDown ? array.get(totalVolumes,  
lowerIndex - 1) : 0.0
```

```
    if canExpandUp and (not canExpandDown or upperVolume >=  
lowerVolume)
```

```
        upperIndex += 1
```

```
        valueAreaVolume += upperVolume
```

```
    else if canExpandDown
```

```
        lowerIndex -= 1
```

```
        valueAreaVolume += lowerVolume
```

```
float poc = profileLow + pocIndex * step + step / 2.0
```

```
float vah = profileLow + (upperIndex + 1) * step
```

```
float val = profileLow + lowerIndex * step
```

```
profile.priceStep := step
```

```
profile.upVolumes := upVolumes
```

```
profile.downVolumes := downVolumes
```

```
profile.totalVolumes := totalVolumes
```

```
profile.maxVolume := maximumVolume
```

```
profile.lowerIndex := lowerIndex
```

```
profile.upperIndex := upperIndex
```

```
profile.pocPrice := poc
```

```
profile.vahPrice := vah
```

```
profile.valPrice := val
```

```
profile.vaMidPrice := val + (vah - val) / 2.0
```

```
profile.isValid := true
```

```
profile
```

```
clearDrawings(box[] boxes, line[] lines, label[] labels) =>
```

```
for profileBox in boxes
```

```
    box.delete(profileBox)
```

```
array.clear(boxes)
```

```
for profileLine in lines
    line.delete(profileLine)
array.clear(lines)
```

```
for profileLabel in labels
    label.delete(profileLabel)
array.clear(labels)
```

```
// --- Session Detection ---
```

```
sessionTimeValue = time(timeframe.period, sessionTime + ":23456",
    timezoneInput)
```

```
inSession = not na(sessionTimeValue)
```

```
isNewSession = inSession and not inSession[1]
```

```
isSessionEnd = not inSession and inSession[1]
```

```
// --- One-Minute Data Collection ---
```

```
[ltfOpen, ltfClose, ltfVolume] = request.security_lower_tf(syminfo.tickerid, "1",
    [open, close, volume])
```

```
var BarInfo[] currentSessionData = array.new<BarInfo>()
```

```
var int currentSessionStartBar = na
```

```
var float currentSessionHigh = na
```

```
var float currentSessionLow = na
```

```
var SessionProfile[] completedProfiles = array.new<SessionProfile>()
```

```
if isNewSession
```

```
    array.clear(currentSessionData)
```

```
    currentSessionStartBar := bar_index
```

```
    currentSessionHigh := high
```

```
    currentSessionLow := low
```

```
if inSession
```

```
    currentSessionHigh := math.max(nz(currentSessionHigh, high), high)
```

```
    currentSessionLow := math.min(nz(currentSessionLow, low), low)
```

```
if array.size(ltfClose) > 0
```

```
    for i = 0 to array.size(ltfClose) - 1
```

```
        float lowerClose = array.get(ltfClose, i)
```

```
        float lowerVolume = array.get(ltfVolume, i)
```

```
        float lowerOpen = array.get(ltfOpen, i)
```

```
        if not na(lowerClose) and not na(lowerVolume)
```

```
            array.push(currentSessionData, BarInfo.new(lowerClose,  
lowerVolume, lowerClose >= nz(lowerOpen, lowerClose)))
```

```
if isSessionEnd and not na(currentSessionStartBar) and not  
na(currentSessionHigh) and not na(currentSessionLow)
```

```
SessionProfile completedProfile = buildProfile(currentSessionData,  
currentSessionStartBar, bar_index - 1, currentSessionHigh,  
currentSessionLow, rowCount, valueAreaPct)
```

```
if completedProfile.isValid
```

```
    array.unshift(completedProfiles, completedProfile)
```

```
    if array.size(completedProfiles) > MAX_PROFILES
```

```
        array.pop(completedProfiles)
```

```
// --- Drawing Storage ---
```

```
var box[] profileBoxes = array.new_box()
```

```
var line[] profileLines = array.new_line()
```

```
var label[] profileLabels = array.new_label()
```

```
var table directionTable = na
```

```
// --- Rendering ---
```

```
if barstate.islast
```

```
    clearDrawings(profileBoxes, profileLines, profileLabels)
```

```
if array.size(completedProfiles) > 0
```

```
    for profileIndex = 0 to array.size(completedProfiles) - 1
```

```
        SessionProfile profile = array.get(completedProfiles, profileIndex)
```

```
        int sessionBarWidth = math.max(1, profile.endBar - profile.startBar)
```

```
        int profileWidth = math.min(histWidth, sessionBarWidth)
```

```
        int profileTransparency = math.min(90, 15 + profileIndex * 15)
```

```

int vaTransparency = math.min(95, vaOpacity + profileIndex * 5)
int outsideTransparency = math.min(98, outOpacity + profileIndex * 4)
color upColor = color.new(COLOR_UP, profileTransparency)
color downColor = color.new(COLOR_DOWN, profileTransparency)

if showHistogramsInput
    for rowIndex = 0 to rowCount - 1
        float rowVolume = array.get(profile.totalVolumes, rowIndex)
        if rowVolume > 0.0
            float rowTop = profile.sessionLow + (rowIndex + 1) *
profile.priceStep
            float rowBottom = profile.sessionLow + rowIndex *
profile.priceStep
            int totalLength = int(math.round(rowVolume / profile.maxVolume *
profileWidth))
            int upLength = int(math.round(array.get(profile.upVolumes,
rowIndex) / rowVolume * totalLength))
            int totalRight = math.min(profile.endBar, profile.startBar +
totalLength)
            int upRight = math.min(profile.endBar, profile.startBar + upLength)
            int transparency = rowIndex >= profile.lowerIndex and rowIndex <=
profile.upperIndex ? vaTransparency : outsideTransparency

            if upLength > 0

```

```
        array.push(profileBoxes, box.new(profile.startBar, rowTop,
upRight, rowBottom, border_color = histThickness > 0 ?
color.new(COLOR_UP, transparency) : na, border_width = histThickness,
bgcolor = color.new(upColor, transparency)))
```

```
    if totalRight > upRight
```

```
        array.push(profileBoxes, box.new(upRight, rowTop, totalRight,
rowBottom, border_color = histThickness > 0 ? color.new(COLOR_DOWN,
transparency) : na, border_width = histThickness, bgcolor =
color.new(downColor, transparency)))
```

```
    if showLevelsInput
```

```
        color levelFade = color.new(chart.fg_color, math.min(80, profileIndex
* 12))
```

```
        array.push(profileLines, line.new(profile.startBar, profile.pocPrice,
profile.endBar, profile.pocPrice, xloc = xloc.bar_index, extend = extend.none,
color = color.new(COLOR_POC, profileIndex * 8), width = levelWidth))
```

```
        array.push(profileLines, line.new(profile.startBar, profile.vahPrice,
profile.endBar, profile.vahPrice, xloc = xloc.bar_index, extend = extend.none,
color = color.new(COLOR_VA, profileIndex * 8), style = line.style_dashed,
width = levelWidth))
```

```
        array.push(profileLines, line.new(profile.startBar, profile.valPrice,
profile.endBar, profile.valPrice, xloc = xloc.bar_index, extend = extend.none,
color = color.new(COLOR_VA, profileIndex * 8), style = line.style_dashed,
width = levelWidth))
```

```
    if showVaMidInput
```

```
        array.push(profileLines, line.new(profile.startBar, profile.vaMidPrice,
profile.endBar, profile.vaMidPrice, xloc = xloc.bar_index, extend =
```

```
extend.none, color = color.new(COLOR_MID, profileIndex * 8), style =  
line.style_dotted, width = levelWidth))
```

```
    if showLabelsInput
```

```
        string dayLabel = "D-" + str.toString(profileIndex + 1)
```

```
        array.push(profileLabels, label.new(profile.startBar, profile.pocPrice,  
dayLabel + " POC", xloc = xloc.bar_index, color = #00000000, textcolor =  
color.new(COLOR_POC, profileIndex * 8), style = label.style_label_right, size =  
size.tiny))
```

```
        array.push(profileLabels, label.new(profile.startBar, profile.vahPrice,  
dayLabel + " VAH", xloc = xloc.bar_index, color = #00000000, textcolor =  
color.new(COLOR_VA, profileIndex * 8), style = label.style_label_right, size =  
size.tiny))
```

```
        array.push(profileLabels, label.new(profile.startBar, profile.valPrice,  
dayLabel + " VAL", xloc = xloc.bar_index, color = #00000000, textcolor =  
color.new(COLOR_VA, profileIndex * 8), style = label.style_label_right, size =  
size.tiny))
```

```
    if showVaMidInput
```

```
        array.push(profileLabels, label.new(profile.startBar,  
profile.vaMidPrice, dayLabel + " VA MID", xloc = xloc.bar_index, color =  
#00000000, textcolor = color.new(COLOR_MID, profileIndex * 8), style =  
label.style_label_right, size = size.tiny))
```

```
// --- Five-Day Directional Summary ---
```

```
if barstate.islast
```

```
    if showDirectionalInput
```

```
        if na(directionTable)
```

```
directionTable := table.new(directionBoxPositionInput, 1, 7, bgcolor =  
color.new(chart.bg_color, 5), frame_color = COLOR_MID, frame_width = 2,  
border_color = color.new(chart.fg_color, 70), border_width = 1)
```

```
int profileCount = array.size(completedProfiles)
```

```
string statusText = "BUILDING DATA"
```

```
string scoreText = "Completed profiles: " + str.toString(profileCount) + "/" +  
str.toString(MAX_PROFILES)
```

```
string pocText = "POC migration: waiting"
```

```
string vaText = "VA Mid migration: waiting"
```

```
string volumeText = "Histogram volume: waiting"
```

```
color statusColor = COLOR_MID
```

```
if profileCount >= MAX_PROFILES
```

```
    int bullishScore = 0
```

```
    int bearishScore = 0
```

```
    SessionProfile newestProfile = array.get(completedProfiles, 0)
```

```
    SessionProfile oldestProfile = array.get(completedProfiles,  
MAX_PROFILES - 1)
```

```
    if newestProfile.pocPrice > oldestProfile.pocPrice
```

```
        bullishScore += 2
```

```
    else if newestProfile.pocPrice < oldestProfile.pocPrice
```

bearishScore += 2

if newestProfile.vaMidPrice > oldestProfile.vaMidPrice

 bullishScore += 2

else if newestProfile.vaMidPrice < oldestProfile.vaMidPrice

 bearishScore += 2

for transitionIndex = 0 to MAX_PROFILES - 2

 SessionProfile newerProfile = array.get(completedProfiles,
transitionIndex)

 SessionProfile olderProfile = array.get(completedProfiles,
transitionIndex + 1)

if newerProfile.pocPrice > olderProfile.pocPrice

 bullishScore += 1

else if newerProfile.pocPrice < olderProfile.pocPrice

 bearishScore += 1

if newerProfile.vaMidPrice > olderProfile.vaMidPrice

 bullishScore += 1

else if newerProfile.vaMidPrice < olderProfile.vaMidPrice

 bearishScore += 1

float aggregateUpVolume = 0.0

```

float aggregateDownVolume = 0.0

for profileIndex = 0 to MAX_PROFILES - 1

    SessionProfile profileForScore = array.get(completedProfiles,
profileIndex)

    for rowIndex = 0 to rowCount - 1

        aggregateUpVolume += array.get(profileForScore.upVolumes,
rowIndex)

        aggregateDownVolume += array.get(profileForScore.downVolumes,
rowIndex)

    if aggregateUpVolume > aggregateDownVolume

        bullishScore += 2

    else if aggregateDownVolume > aggregateUpVolume

        bearishScore += 2

int netScore = bullishScore - bearishScore

statusText := netScore >= 3 ? "BULLISH PROFILE TREND" : netScore <= -
3 ? "BEARISH PROFILE TREND" : "MIXED / BALANCED PROFILE"

statusColor := netScore >= 3 ? COLOR_UP : netScore <= -3 ?
COLOR_DOWN : COLOR_MID

scoreText := "Score: " + str.toString(bullishScore) + " bullish / " +
str.toString(bearishScore) + " bearish"

pocText := newestProfile.pocPrice > oldestProfile.pocPrice ? "POC
migration: rising" : newestProfile.pocPrice < oldestProfile.pocPrice ? "POC
migration: falling" : "POC migration: flat"

```

```
vaText := newestProfile.vaMidPrice > oldestProfile.vaMidPrice ? "VA Mid  
migration: rising" : newestProfile.vaMidPrice < oldestProfile.vaMidPrice ? "VA  
Mid migration: falling" : "VA Mid migration: flat"
```

```
volumeText := aggregateUpVolume > aggregateDownVolume ?  
"Histogram volume: bullish" : aggregateDownVolume > aggregateUpVolume ?  
"Histogram volume: bearish" : "Histogram volume: balanced"
```

```
color cellBackground = color.new(chart.bg_color, 0)
```

```
table.cell(directionTable, 0, 0, "MFB - VRS - 5 DAYS", text_color =  
COLOR_MID, text_size = size.normal, text_halign = text.align_left, bgcolor =  
cellBackground)
```

```
table.cell(directionTable, 0, 1, statusText, text_color = statusColor,  
text_size = size.large, text_halign = text.align_left, bgcolor = cellBackground)
```

```
table.cell(directionTable, 0, 2, scoreText, text_color = chart.fg_color,  
text_size = size.normal, text_halign = text.align_left, bgcolor = cellBackground)
```

```
table.cell(directionTable, 0, 3, pocText, text_color = COLOR_POC, text_size  
= size.normal, text_halign = text.align_left, bgcolor = cellBackground)
```

```
table.cell(directionTable, 0, 4, vaText, text_color = COLOR_VA, text_size =  
size.normal, text_halign = text.align_left, bgcolor = cellBackground)
```

```
table.cell(directionTable, 0, 5, volumeText, text_color = statusColor,  
text_size = size.normal, text_halign = text.align_left, bgcolor = cellBackground)
```

```
table.cell(directionTable, 0, 6, "Fixed after each NY close", text_color =  
color.new(chart.fg_color, 20), text_size = size.small, text_halign =  
text.align_left, bgcolor = cellBackground)
```

```
else if not na(directionTable)
```

```
table.delete(directionTable)
```

```
directionTable := na
```

```
// --- Background Highlighting ---
```

```
bgcolor(showBackgroundInput and inSession ? COLOR_BG : na)
```