

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

// © LuxAlgo

//@version=6

indicator("MFB - Value Reading Session", overlay = true, maxBarsBack = 5000)

// --- Settings ---

// Anchored to the Previous Day's RTH Session (9:30am - 4:00pm)

sessionTime = input.session("0930-1600", "Structural Session", tooltip = "The time range
of the PREVIOUS day used for context (Institutional RTH).")

timezoneInput = input.string("America/New_York", "Timezone", tooltip = "Timezone for the
session calculation.")

valueAreaPct = input.float(70.0, "Value Area Percentage", minval = 0, maxval = 100) / 100.0

rowCount = input.int(50, "Row Resolution", minval = 10, maxval = 100)

histWidth = input.int(40, "Histogram Width (Bars)", minval = 5, maxval = 200)

labelSize = input.string(size.normal, "Label Size", options = [size.small, size.normal,
size.large, size.huge])

// --- Visibility Settings ---

vaOpacity = input.int(30, "VA Opacity %", minval = 0, maxval = 100)

outOpacity = input.int(70, "Outside VA Opacity %", minval = 0, maxval = 100)

histThickness = input.int(3, "Histogram Thickness", minval = 1, maxval = 5)

// --- Colors ---

COLOR_UP = #26a69a // Teal

COLOR_DOWN = #ef5350 // Pink/Red

```
COLOR_POC    = #9c27b0 // Purple
COLOR_VA     = #6a1b9a // Dark Purple
COLOR_BG     = color.new(#9c27b0, 95)
```

```
// --- Logic: Session Detection ---
```

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

```
inSession = not na(t)
```

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

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

```
// --- Data Collection ---
```

```
[ltfO, ltfC, ltfV] = request.security_lower_tf(syminfo.tickerid, "1", [open, close, volume])
```

```
type BarInfo
```

```
    float price
```

```
    float volume
```

```
    bool isUp
```

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

```
var int  currentSessionStart = na
```

```
var float currentSessionHigh  = 0.0
```

```
var float currentSessionLow   = 1e10
```

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

```
var int  lastSessionStartBar = na
```

```
var int  lastSessionEndBar   = na
```

```
var float lastSessionHigh  = 0.0
```

```
var float lastSessionLow   = 0.0
```

```
if isNewSession
```

```
    currentSessionData.clear()
```

```
    currentSessionStart := bar_index
```

```
    currentSessionHigh  := high
```

```
    currentSessionLow   := low
```

```
if inSession
```

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

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

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

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

```
            p = array.get(ltfC, i)
```

```
            v = array.get(ltfV, i)
```

```
            o = array.get(ltfO, i)
```

```
            if not na(p) and not na(v)
```

```
                currentSessionData.push(BarInfo.new(p, v, p >= nz(o, p)))
```

```
if isSessionEnd
```

```
    lastCompletedData := currentSessionData.copy()
```

```
    lastSessionStartBar := currentSessionStart
```

```
    lastSessionEndBar   := bar_index
```

```
    lastSessionHigh     := currentSessionHigh
```

```
    lastSessionLow      := currentSessionLow
```

```

// --- Rendering (ONLY ON THE LAST BAR) ---

var line[] profileLines = array.new_line()

var label[] profileLabels = array.new_label()


if barstate.islast and array.size(lastCompletedData) > 0

    for l in profileLines
        line.delete(l)

    array.clear(profileLines)

    for lb in profileLabels
        label.delete(lb)

    array.clear(profileLabels)


float priceRange = lastSessionHigh - lastSessionLow
if priceRange > 0

    float priceStep = priceRange / rowCount

    float[] rowUpVol  = array.new_float(rowCount, 0.0)
    float[] rowDownVol = array.new_float(rowCount, 0.0)
    float[] rowTotal  = array.new_float(rowCount, 0.0)
    float totalVol = 0.0


    for bar in lastCompletedData

        int rowIdx = math.min(rowCount - 1, math.floor((bar.price - lastSessionLow) /
priceStep))

        if rowIdx >= 0

            if bar.isUp

```

```

        rowUpVol.set(rowIdx, rowUpVol.get(rowIdx) + bar.volume)
    else
        rowDownVol.set(rowIdx, rowDownVol.get(rowIdx) + bar.volume)
    rowTotal.set(rowIdx, rowTotal.get(rowIdx) + bar.volume)
    totalVol += bar.volume

float maxVol = array.max(rowTotal)
if maxVol > 0
    int pocIdx = array.indexOf(rowTotal, maxVol)
    float pocPrice = lastSessionLow + (pocIdx * priceStep) + (priceStep / 2)

    float vaTarget = totalVol * valueAreaPct
    float currentVaVol = maxVol
    int upIdx = pocIdx
    int downIdx = pocIdx
    while currentVaVol < vaTarget and (upIdx < rowCount - 1 or downIdx > 0)
        float uV = upIdx < rowCount - 1 ? rowTotal.get(upIdx + 1) : 0
        float dV = downIdx > 0 ? rowTotal.get(downIdx - 1) : 0
        if uV >= dV
            upIdx += 1
            currentVaVol += uV
        else
            downIdx -= 1
            currentVaVol += dV

    float vahPrice = lastSessionLow + (upIdx * priceStep) + priceStep

```

```

float valPrice = lastSessionLow + (downIdx * priceStep)

// Draw Histogram
for i = 0 to rowCount - 1

    float rowY = lastSessionLow + (i * priceStep) + (priceStep / 2)

    float rT = rowTotal.get(i)

    if rT > 0

        int totalLen = math.round((rT / maxVol) * histWidth)

        int upLen = math.round((rowUpVol.get(i) / rT) * totalLen)

        int trans = (i >= downIdx and i <= upIdx) ? vaOpacity : outOpacity

        profileLines.push(line.new(lastSessionStartBar, rowY, lastSessionStartBar +
upLen, rowY, color = color.new(COLOR_UP, trans), width = histThickness))

        profileLines.push(line.new(lastSessionStartBar + upLen, rowY,
lastSessionStartBar + totalLen, rowY, color = color.new(COLOR_DOWN, trans), width =
histThickness))

// Draw Extension Lines from end of session into the current day

profileLines.push(line.new(lastSessionEndBar, pocPrice, bar_index + 50, pocPrice,
color = COLOR_POC, width = 2, extend = extend.right))

profileLines.push(line.new(lastSessionEndBar, vahPrice, bar_index + 50, vahPrice,
color = COLOR_VA, style = line.style_dashed, width = 2, extend = extend.right))

profileLines.push(line.new(lastSessionEndBar, valPrice, bar_index + 50, valPrice,
color = COLOR_VA, style = line.style_dashed, width = 2, extend = extend.right))

profileLabels.push(label.new(bar_index + 5, pocPrice, "POC", color = #00000000,
textcolor = COLOR_POC, style = label.style_label_left, size = labelSize))

```

```
profileLabels.push(label.new(bar_index + 5, vahPrice, "VAH", color = #00000000,  
textcolor = COLOR_VA, style = label.style_label_left, size = labelSize))
```

```
profileLabels.push(label.new(bar_index + 5, valPrice, "VAL", color = #00000000,  
textcolor = COLOR_VA, style = label.style_label_left, size = labelSize))
```

```
// Background highlighting for the session
```

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