

```
// 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 Acceptance", shorttitle="MFB - VA", overlay=false)
```

```
// --- Groups ---
```

```
string G_SESS = "Session Settings"
```

```
string G_VWAP = "VWAP Settings"
```

```
string G_VA = "Value Area (Bands)"
```

```
string G_EMA = "Momentum Filter (EMAs)"
```

```
string G_RSI = "RSI Oscillator"
```

```
string G_VA_ACC = "Value Acceptance Settings"
```

```
string G_DASH = "Dashboard"
```

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

```
string sessionInput = input.session("0930-1600", "Session Time",  
group=G_SESS, tooltip="Define the active trading session.")
```

```
string timeZoneInput = input.string("America/New_York", "Timezone",  
group=G_SESS, tooltip="Timezone for the session calculation.")
```

```
bool showHighlight = input.bool(true, "Highlight Session", group=G_SESS,  
tooltip="Highlights the background during the active session.")
```

```
bool showPrev = input.bool(false, "Show Previous Session Levels",  
group=G_SESS, tooltip="Keep levels visible after the session ends.")
```

```
sourceInput      = input.source(ohlc4, "VWAP Source", group=G_VWAP,  
tooltip="Source price for VWAP calculations.")
```

```
bool showVwapLine = input.bool(true, "Show Running VWAP",  
group=G_VWAP)
```

```
bool showStartVwap = input.bool(false, "Show Session Start VWAP",  
group=G_VWAP)
```

```
bool showHighVwap = input.bool(false, "Show Session High VWAP",  
group=G_VWAP)
```

```
bool showLowVwap  = input.bool(false, "Show Session Low VWAP",  
group=G_VWAP)
```

```
// Value Area / StdDev Bands
```

```
bool showBands    = input.bool(false, "Show Value Area (1 StdDev)",  
group=G_VA)
```

```
float stdDevMult   = input.float(1.0, "StdDev Multiplier", minval=0.1,  
group=G_VA, tooltip="Multiplier for the Value Area bands.")
```

```
// EMA Inputs
```

```
bool showEma9      = input.bool(true, "Show 9 EMA", group=G_EMA)
```

```
bool showEma20     = input.bool(true, "Show 20 EMA", group=G_EMA)
```

```
int lenEma9        = input.int(9, "9 EMA Length", minval=1, group=G_EMA)
```

```
int lenEma20       = input.int(20, "20 EMA Length", minval=1, group=G_EMA)
```

```
// RSI Inputs
```

```
bool showRsi    = input.bool(true, "Show RSI Oscillator", group=G_RSI)
int  lenRsi     = input.int(14, "RSI Length", minval=1, group=G_RSI)
sourceRsi      = input.source(close, "RSI Source", group=G_RSI)
```

```
// Value Acceptance Inputs
```

```
int  acceptanceBars = input.int(3, "Acceptance Consecutive Bars", minval=1,
group=G_VA_ACC, tooltip="Number of consecutive bars the alignment must
hold to be considered a confirmed 'Value Acceptance'.")
```

```
bool showSignals  = input.bool(true, "Show Acceptance Signals",
group=G_VA_ACC, tooltip="Displays chart shapes when Value Acceptance
occurs.")
```

```
// Dashboard Input
```

```
bool showDash    = input.bool(true, "Show Bias Dashboard",
group=G_DASH)
```

```
// --- Calculations ---
```

```
t = time(timeframe.period, sessionInput, timezoneInput)
```

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

```
bool isNewSession = inSession and na(t[1])
```

```
// EMA Calculations
```

```
float ema9 = ta.ema(close, lenEma9)
```

```
float ema20 = ta.ema(close, lenEma20)
```

```
// Cumulative variables for VWAP
```

```
var float sumPV = 0.0
```

```
var float sumV = 0.0
```

```
var float sumPV2 = 0.0
```

```
// Levels to track
```

```
var float startVwap = na
```

```
var float highVwap = na
```

```
var float lowVwap = na
```

```
var float sessHigh = -1.0e10
```

```
var float sessLow = 1.0e10
```

```
if isNewSession
```

```
    sumPV := sourceInput * volume
```

```
    sumV := volume
```

```
    sumPV2 := volume * math.pow(sourceInput, 2)
```

```
    sessHigh := high
```

```
    sessLow := low
```

```
    startVwap := sumPV / sumV
```

```
    highVwap := startVwap
```

```
    lowVwap := startVwap
```

```
else if inSession
```

```
    sumPV += sourceInput * volume
```

```
sumV += volume
```

```
sumPV2 += volume * math.pow(sourceInput, 2)
```

```
if high > sessHigh
```

```
    sessHigh := high
```

```
    highVwap := sumPV / sumV
```

```
if low < sessLow
```

```
    sessLow := low
```

```
    lowVwap := sumPV / sumV
```

```
// Calculate Running VWAP & Bands
```

```
float vwapValue = sumV > 0 ? sumPV / sumV : na
```

```
float variance = sumV > 0 ? (sumPV2 / sumV) - math.pow(vwapValue, 2) : 0
```

```
float stdev = math.sqrt(math.max(0, variance))
```

```
float upperBand = vwapValue + stdev * stdDevMult
```

```
float lowerBand = vwapValue - stdev * stdDevMult
```

```
// Handle Persistence
```

```
bool display = inSession or showPrev
```

```
// RSI Calculation
```

```
float rsiValue = ta.rsi(sourceRsi, lenRsi)
```

```
// --- Alignment Logic (The Dashboard Rules) ---
```

```
bool isBullish = close > vwapValue and rsiValue > 50 and ema9 > ema20
```

```
bool isBearish = close < vwapValue and rsiValue < 50 and ema9 < ema20
```

```
// Value Acceptance Calculation
```

```
bool bullishAccepted = math.sum(isBullish ? 1 : 0, acceptanceBars) ==  
acceptanceBars
```

```
bool bearishAccepted = math.sum(isBearish ? 1 : 0, acceptanceBars) ==  
acceptanceBars
```

```
bool bullAcceptanceSignal = bullishAccepted and not bullishAccepted[1]
```

```
bool bearAcceptanceSignal = bearishAccepted and not bearishAccepted[1]
```

```
string currentBias = bullishAccepted ? "BULL CONFIRMED" : bearishAccepted  
? "BEAR CONFIRMED" : isBullish ? "BULLISH" : isBearish ? "BEARISH" :  
"NEUTRAL"
```

```
color biasColor = bullishAccepted ? #089981 : bearishAccepted ? #f23645 :  
isBullish ? color.new(#089981, 40) : isBearish ? color.new(#f23645, 40) :  
#5b9cf6
```

```
color vwapColor = bullishAccepted ? #056656 : bearishAccepted ? #f23645 :  
#5b9cf6
```

```
// --- Visualization ---
```

```
// Dashboard Table (Anchored to top right)
```

```
var table dash = table.new(position.top_right, 2, 2,  
frame_color=chart.fg_color, frame_width=1, border_width=1,  
border_color=chart.fg_color)
```

```
if showDash and barstate.islast
```

```
    table.cell(dash, 0, 0, "BIAS", bgcolor=color.new(chart.bg_color, 10),  
text_color=chart.fg_color, text_size=size.small)
```

```
    table.cell(dash, 1, 0, currentBias, bgcolor=biasColor, text_color=color.white,  
text_size=size.small)
```

```
    table.cell(dash, 0, 1, "VALUE", bgcolor=color.new(chart.bg_color, 10),  
text_color=chart.fg_color, text_size=size.small)
```

```
    table.cell(dash, 1, 1, close > vwapValue ? "ABOVE VWAP" : "BELOW VWAP",  
bgcolor=color.new(chart.bg_color, 20), text_color=chart.fg_color,  
text_size=size.small)
```

```
// Background Highlight on Main Chart
```

```
bgcolor(showHighlight and inSession ? color.new(#5b9cf6, 92) : na,  
title="Session Highlight", force_overlay=true)
```

```
// Running VWAP & Bands on Main Chart
```

```
plot(showVwapLine and display and not isNewSession ? vwapValue : na,  
"Running VWAP", color=vwapColor, linewidth=2, style=plot.style_linebr,  
force_overlay=true)
```

```
p_upper = plot(showBands and display and not isNewSession ? upperBand :  
na, "Upper VA", color=color.new(#089981, 50), style=plot.style_linebr,  
force_overlay=true)
```

```
p_lower = plot(showBands and display and not isNewSession ? lowerBand :  
na, "Lower VA", color=color.new(#f23645, 50), style=plot.style_linebr,  
force_overlay=true)
```

```
fill(p_upper, p_lower, showBands and display ? color.new(#5b9cf6, 95) : na,  
title="Value Area Fill")
```

```
// Acceptance Signals on Main Chart
```

```
// (Removed from main chart as requested)
```

```
// Horizontal captured levels on Main Chart
```

```
plot(showStartVwap and display and not isNewSession ? startVwap : na,  
"Session Start VWAP", color=color.fuchsia, linewidth=1,  
style=plot.style_linebr, force_overlay=true)
```

```
plot(showHighVwap and display and not isNewSession ? highVwap : na,  
"Session High VWAP", color=color.purple, linewidth=1, style=plot.style_linebr,  
force_overlay=true)
```

```
plot(showLowVwap and display and not isNewSession ? lowVwap : na,  
"Session Low VWAP", color=color.navy, linewidth=1, style=plot.style_linebr,  
force_overlay=true)
```

```
// Momentum Filter EMAs on Main Chart
```

```
plot(showEma9 ? ema9 : na, "9 EMA", color=#089981, linewidth=1,  
force_overlay=true)
```

```
plot(showEma20 ? ema20 : na, "20 EMA", color=#f23645, linewidth=1,  
force_overlay=true)
```

```

// RSI Oscillator in separate pane

hline(70, "Overbought", color=color.new(#f23645, 50),
linestyle=hline.style_dashed)

hline(50, "Center", color=color.new(chart.fg_color, 80),
linestyle=hline.style_dotted)

hline(30, "Oversold", color=color.new(#089981, 50),
linestyle=hline.style_dashed)


color rsiLineColor = rsiValue > 50 ? #089981 : #f23645
plotRsi = plot(showRsi ? rsiValue : na, "RSI", color=rsiLineColor, linewidth=2)
plotMid = plot(50, "RSI Midline", display=display.none)


fill(plotRsi, plotMid,
    top_value  = math.max(50, rsiValue),
    bottom_value = math.min(50, rsiValue),
    top_color   = rsiValue > 50 ? color.new(#089981, 50) :
color.new(chart.bg_color, 100),
    bottom_color = rsiValue > 50 ? color.new(chart.bg_color, 100) :
color.new(#f23645, 50),
    title      = "RSI Gradient")


// Momentum Change label on lower box (RSI Pane)

// (Removed as requested)


// --- RSI Divergence ---

```

```
int pivotRight = input.int(5, "Pivot Right", group=G_RSI)
```

```
int pivotLeft = input.int(5, "Pivot Left", group=G_RSI)
```

```
float ph = ta.pivohigh(rsiValue, pivotLeft, pivotRight)
```

```
float pl = ta.pivotlow(rsiValue, pivotLeft, pivotRight)
```

```
bool isRsiPH = not na(ph)
```

```
bool isRsiPL = not na(pl)
```

```
var float lastRsiPH = na
```

```
var float lastPricePH = na
```

```
var int lastRsiPHBar = na
```

```
var float lastRsiPL = na
```

```
var float lastPricePL = na
```

```
var int lastRsiPLBar = na
```

```
if isRsiPH
```

```
    lastRsiPH := ph
```

```
    lastPricePH := high[pivotRight]
```

```
    lastRsiPHBar := bar_index[pivotRight]
```

```
if isRsiPL
```

```
lastRsiPL := pl
```

```
lastPricePL := low[pivotRight]
```

```
lastRsiPLBar := bar_index[pivotRight]
```

```
// Bearish Divergence (Price higher high, RSI lower high)
```

```
bool bearDiv = isRsiPH and (high[pivotRight] > lastPricePH[1]) and (ph < lastRsiPH[1])
```

```
// Bullish Divergence (Price lower low, RSI higher low)
```

```
bool bullDiv = isRsiPL and (low[pivotRight] < lastPricePL[1]) and (pl > lastRsiPL[1])
```

```
if bearDiv
```

```
    line.new(lastRsiPHBar[1], lastRsiPH[1], bar_index[pivotRight], ph, color=color.new(#f23645, 0), width=2, force_overlay=false)
```

```
if bullDiv
```

```
    line.new(lastRsiPLBar[1], lastRsiPL[1], bar_index[pivotRight], pl, color=color.new(#089981, 0), width=2, force_overlay=false)
```

```
// --- Alerts ---
```

```
alertcondition(isBullish and not isBullish[1], "Full Bullish Alignment", "MFB alignment: Price > VWAP, RSI > 50, 9EMA > 20EMA.")
```

```
alertcondition(isBearish and not isBearish[1], "Full Bearish Alignment", "MFB alignment: Price < VWAP, RSI < 50, 9EMA < 20EMA.")
```

```
alertcondition(bullAcceptanceSignal, "Value Acceptance (Bullish)", "Momentum Accepted: Bullish conditions met for chosen consecutive bars.")
```

```
alertcondition(bearAcceptanceSignal, "Value Acceptance (Bearish)",  
"Momentum Accepted: Bearish conditions met for chosen consecutive bars.")
```

```
alertcondition(bullAcceptanceSignal or bearAcceptanceSignal, "Momentum  
Change", "Momentum Accepted: Trend changed for chosen consecutive  
bars.")
```

```
alertcondition(bullDiv or bearDiv, "RSI Divergence", "RSI divergence  
detected.")
```