

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

//@version=6

indicator('MFB - Premarket 5 min Sweep/FVG-123', overlay = true, max\_labels\_count = 500,  
max\_lines\_count = 500)

// --- Group Name: FVG Settings ---

fvg\_max\_age = input.int(100, 'Max FVG Age (Bars)', minval = 1, group = 'FVG Settings',  
tooltip = 'Only consider FVGs created within this many bars.')

fvg\_min\_width = input.float(0, 'Min FVG Width (Ticks)', minval = 0, group = 'FVG Settings',  
tooltip = 'Minimum size of the FVG in ticks to be considered valid.')

priming\_lookback = input.int(30, 'Level Touch Lookback', minval = 1, group = 'Logic  
Settings', tooltip = 'How many bars the "Level Touch" remains active.')

// --- Group Name: IFVG Sweep Settings ---

show\_ifvg = input.bool(true, 'Enable IFVG Sweep Setup', group = 'IFVG Sweep Settings',  
tooltip = 'Toggle the IFVG trade setup (Sweep -> FVG beyond level -> Inversion).')

ifvg\_rr = input.float(2.0, 'IFVG Take Profit RR', minval = 0.1, step = 0.1, group = 'IFVG  
Sweep Settings', tooltip = 'Risk-to-Reward ratio for the IFVG setup.')

// --- Group Name: 9 EMA Retrace Settings ---

emaFastLen = input.int(9, "Fast EMA", group = "9 EMA Retrace Settings")

emaSlowLen = input.int(20, "Slow EMA", group = "9 EMA Retrace Settings")

emaRrRatio = input.float(2.0, "Risk/Reward Ratio", step = 0.5, group = "9 EMA Retrace  
Settings")

lineLength = input.int(25, "Trade Line Length (Bars)", minval = 1, tooltip = "How many bars  
into the future the entry/stop/target lines should project", group = "9 EMA Retrace Settings")

```
requireCross = input.bool(true, "Require EMA Cross Before Retrace", tooltip = "If disabled, signals can trigger on any pullback to the 9 EMA as long as the 9 EMA is above/below the 20 EMA (Continuation trades).", group = "9 EMA Retrace Settings")
```

```
// --- Group Name: Display Settings ---
```

```
show_labels = input.bool(true, 'Show Entry Labels', group = 'Display Settings', tooltip = 'Toggles the visibility of signals.')
```

```
show_history = input.bool(false, 'Show Trade History', group = 'Display Settings', tooltip = 'If disabled, only the most recent trade will be shown.')
```

```
max_history = input.int(5, 'Max Trades to Show', minval = 1, maxval = 50, group = 'Display Settings', tooltip = 'How many historical trades to keep on the chart.')
```

```
// =====
```

```
// 🕒 Levels Logic (INTERNAL ONLY)
```

```
// =====
```

```
is_in_sess(s) => not na(time('D', s, 'America/New_York'))
```

```
is_new_b(s) =>
```

```
    t = time('D', s, 'America/New_York')
```

```
    na(t[1]) and not na(t) or t[1] < t
```

```
get_bounds(s) =>
```

```
    var float h = na
```

```
    var float l = na
```

```
    if ta.change(time('D', 'America/New_York')) != 0
```

```
        h := na
```

```
        l := na
```

```
    if is_in_sess(s)
```

```

new_bar = is_new_b(s)

l := new_bar ? low : na(l) ? low : math.min(l, low)

h := new_bar ? high : na(h) ? high : math.max(h, high)

[h, l]

[ash, asl] = get_bounds('1800-0000') // Asian
[loh, lol] = get_bounds('0000-0600') // London
[o15h, o15l] = get_bounds('0930-0945') // 9:30 Open

pdh = request.security(syminfo.tickerid, 'D', high[1], lookahead = barmerge.lookahead_on)
pdl = request.security(syminfo.tickerid, 'D', low[1], lookahead = barmerge.lookahead_on)
pwh = request.security(syminfo.tickerid, 'W', high[1], lookahead = barmerge.lookahead_on)
pwl = request.security(syminfo.tickerid, 'W', low[1], lookahead = barmerge.lookahead_on)

// =====

// ⚡ Intention & Level Tracking

// =====

float highest_extreme = math.max(nz(pdh), nz(ash), nz(loh), nz(o15h), nz(pwh))

float lowest_extreme = math.min(nz(pdl, 999999), nz(asl, 999999), nz(lol, 999999),
nz(o15l, 999999), nz(pwl, 999999))

bullCrossPdh = ta.crossover(close, pdh)
bullCrossAsh = ta.crossover(close, ash)
bullCrossLoh = ta.crossover(close, loh)
bullCrossO15h = ta.crossover(close, o15h)
bullCrossPwh = ta.crossover(close, pwh)

```

breachBullLogic = bullCrossPdh or bullCrossAsh or bullCrossLoh or bullCrossO15h or  
bullCrossPwh

bearCrossPdl = ta.crossunder(close, pdl)

bearCrossAsl = ta.crossunder(close, asl)

bearCrossLol = ta.crossunder(close, lol)

bearCrossO15l = ta.crossunder(close, o15l)

bearCrossPwl = ta.crossunder(close, pwl)

breachBearLogic = bearCrossPdl or bearCrossAsl or bearCrossLol or bearCrossO15l or  
bearCrossPwl

var bool touchedHigh = false

var bool touchedLow = false

if high >= highest\_extreme

    touchedHigh := true

if low <= lowest\_extreme

    touchedLow := true

last\_high\_touch = ta.barssince(high >= highest\_extreme)

last\_low\_touch = ta.barssince(low <= lowest\_extreme)

if not na(last\_high\_touch) and last\_high\_touch > priming\_lookback

    touchedHigh := false

if not na(last\_low\_touch) and last\_low\_touch > priming\_lookback

    touchedLow := false

```

// =====
// 📈 9 EMA Retrace Logic
// =====

emaFast = ta.ema(close, emaFastLen)
emaSlow = ta.ema(close, emaSlowLen)

var int longState = 0
var int shortState = 0

emaCrossUp = ta.crossover(emaFast, emaSlow)
emaCrossDn = ta.crossunder(emaFast, emaSlow)

if emaCrossUp
    longState := 1
    shortState := 0
else if emaCrossDn
    shortState := 1
    longState := 0
else if not requireCross
    if emaFast > emaSlow and longState == 0
        longState := 1
    if emaFast < emaSlow and shortState == 0
        shortState := 1

if longState == 1 and low <= emaFast
    longState := 2

```

```
if shortState == 1 and high >= emaFast
```

```
    shortState := 2
```

```
longSignal = longState == 2 and close > emaFast
```

```
shortSignal = shortState == 2 and close < emaFast
```

```
if longSignal
```

```
    longState := 0
```

```
if shortSignal
```

```
    shortState := 0
```

```
plot(emaFast, "Fast EMA", color = color.blue)
```

```
plot(emaSlow, "Slow EMA", color = color.red)
```

```
plotshape(longSignal and show_labels, "Long Entry", shape.triangleup, location.belowbar,  
color.new(color.green, 0), size = size.small)
```

```
plotshape(shortSignal and show_labels, "Short Entry", shape.triangledown,  
location.abovebar, color.new(color.red, 0), size = size.small)
```

```
// =====
```

```
// ✨ FVG Logic
```

```
// =====
```

```
type FVG
```

```
    float top
```

```
    float bottom
```

```
    bool isBull
```

bool active

int createdBar

bool touched

bool done

int lastTouchBar

bool beyondHigh

bool beyondLow

var fvgArray = array.new<FVG>()

if bar\_index >= 2

bool isBull = low > high[2]

bool isBear = high < low[2]

if isBull or isBear

float fvgTop = isBull ? low : low[2]

float fvgBot = isBull ? high[2] : high

bool isBeyondHigh = touchedHigh and fvgBot >= highest\_extreme

bool isBeyondLow = touchedLow and fvgTop <= lowest\_extreme

array.push(fvgArray, FVG.new(fvgTop, fvgBot, isBull, true, bar\_index, false, false, na,  
isBeyondHigh, isBeyondLow))

if array.size(fvgArray) > 100

array.shift(fvgArray)

```

atr = ta.atr(14)

var string lastIntention = ""

if breachBullLogic
    lastIntention := "Bullish"
else if breachBearLogic
    lastIntention := "Bearish"

var line[] all_lines = array.new_line()
var label[] all_labels = array.new_label()

manage_history() =>
    limit = show_history ? max_history : 1
    while array.size(all_labels) > limit
        label.delete(array.shift(all_labels))
    if array.size(all_lines) >= 3
        line.delete(array.shift(all_lines))
        line.delete(array.shift(all_lines))
        line.delete(array.shift(all_lines))

bool alertTrigger = false
var int lastSignalBar = 0

// Track EMA Retrace Lines
if longSignal
    alertTrigger := true
    entryPrice = close

```

```

stopLoss = emaSlow

risk = entryPrice - stopLoss

takeProfit = entryPrice + (risk * emaRrRatio)

if risk > 0

    if show_labels

        label.new(bar_index, low - atr, "EMA Retrace ▲", color = #00000000, textcolor =
color.green, style = label.style_label_up, size = size.normal)

        line.new(bar_index, entryPrice, bar_index + lineLength, entryPrice, color = color.gray,
style = line.style_dotted, width = 2)

        line.new(bar_index, stopLoss, bar_index + lineLength, stopLoss, color = color.red,
style = line.style_dotted, width = 2)

        line.new(bar_index, takeProfit, bar_index + lineLength, takeProfit, color = color.green,
style = line.style_dotted, width = 2)

if shortSignal

    alertTrigger := true

    entryPrice = close

    stopLoss = emaSlow

    risk = stopLoss - entryPrice

    takeProfit = entryPrice - (risk * emaRrRatio)

if risk > 0

    if show_labels

        label.new(bar_index, high + atr, "EMA Retrace ▼", color = #00000000, textcolor =
color.red, style = label.style_label_down, size = size.normal)

        line.new(bar_index, entryPrice, bar_index + lineLength, entryPrice, color = color.gray,
style = line.style_dotted, width = 2)

```

```
    line.new(bar_index, stopLoss, bar_index + lineLength, stopLoss, color = color.red,
style = line.style_dotted, width = 2)
```

```
    line.new(bar_index, takeProfit, bar_index + lineLength, takeProfit, color = color.green,
style = line.style_dotted, width = 2)
```

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

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

```
        fvgObj = array.get(fvgArray, i)
```

```
        if fvgObj.active
```

```
            // --- IFVG Logic ---
```

```
            if show_ifvg and bar_index > lastSignalBar
```

```
                bool isBullIFVG = not fvgObj.isBull and close > fvgObj.top and fvgObj.beyondLow
and touchedLow
```

```
                bool isBearIFVG = fvgObj.isBull and close < fvgObj.bottom and fvgObj.beyondHigh
and touchedHigh
```

```
            if (isBullIFVG or isBearIFVG) and barstate.isconfirmed
```

```
                lastSignalBar := bar_index
```

```
                alertTrigger := true
```

```
                float ep = close
```

```
                float sl = isBullIFVG ? low - (2 * syminfo.mintick) : high + (2 * syminfo.mintick)
```

```
                float r = math.abs(ep - sl)
```

```
                float tp = isBullIFVG ? ep + (r * ifvg_rr) : ep - (r * ifvg_rr)
```

```
            if show_labels
```

```
                array.push(all_labels, label.new(bar_index, isBullIFVG ? low - atr * 1.5 : high +
atr * 1.5, "IFVG Sweep " + (isBullIFVG ? "▲" : "▼"), color = #00000000, textcolor = #5b9cf6,
style = isBullIFVG ? label.style_label_up : label.style_label_down, size = size.normal))
```

```

        array.push(all_lines, line.new(bar_index, ep, bar_index + 10, ep, color =
#5b9cf6, style = line.style_dotted, width = 2))

        array.push(all_lines, line.new(bar_index, sl, bar_index + 10, sl, color = #f23645,
style = line.style_dotted, width = 2))

        array.push(all_lines, line.new(bar_index, tp, bar_index + 10, tp, color = #089981,
style = line.style_dotted, width = 2))

        manage_history()

```

```

alert("IFVG Sweep Entry Identified", alert.freq_once_per_bar_close)

```

```

fvgObj.done := true

```

```

fvgObj.active := false

```

```

touchedHigh := false

```

```

touchedLow := false

```

```

continue

```

```

// --- Standard FVG-123 Logic ---

```

```

if (fvgObj.isBull ? low <= fvgObj.bottom : high >= fvgObj.top)

```

```

    fvgObj.active := false

```

```

    fvgObj.done := true

```

```

if not fvgObj.done

```

```

    afterP = bar_index > fvgObj.createdBar

```

```

    if afterP and (high >= fvgObj.bottom) and (low <= fvgObj.top) and (fvgObj.isBull ?
close >= fvgObj.bottom : close <= fvgObj.top)

```

```

        fvgObj.touched := true

```

```

        fvgObj.lastTouchBar := bar_index

```

```
if afterP and ((not fvgObj.isBull and close > fvgObj.top) or (fvgObj.isBull and close < fvgObj.bottom))
```

```
fvgObj.done := true
```

```
haveT = fvgObj.touched and not na(fvgObj.lastTouchBar) and bar_index > fvgObj.lastTouchBar and afterP
```

```
if (fvgObj.isBull and haveT and close > fvgObj.top) or (not fvgObj.isBull and haveT and close < fvgObj.bottom)
```

```
bool is_sweep_setup = fvgObj.isBull ? (lastIntention == "Bearish") : (lastIntention == "Bullish")
```

```
bool levelEngaged = fvgObj.isBull ?
```

```
((is_sweep_setup and touchedLow) or low <= lowest_extreme) :
```

```
((is_sweep_setup and touchedHigh) or high >= highest_extreme)
```

```
if is_sweep_setup and levelEngaged and (bar_index - fvgObj.createdBar <= fvg_max_age) and barstate.isconfirmed and bar_index > lastSignalBar
```

```
lastSignalBar := bar_index
```

```
alertTrigger := true
```

```
if show_labels
```

```
float ep = close
```

```
float sl = fvgObj.isBull ? fvgObj.bottom - syminfo.mintick : fvgObj.top + syminfo.mintick
```

```
float r = math.abs(ep - sl)
```

```
float tp = fvgObj.isBull ? ep + (r * 2) : ep - (r * 2)
```

```
array.push(all_labels, label.new(bar_index, fvgObj.isBull ? low - atr * 2.5 : high + atr * 2.5, "Sweep/FVG-123 " + (fvgObj.isBull ? "▲" : "▼"), color = #00000000, textcolor = #5b9cf6, style = fvgObj.isBull ? label.style_label_up : label.style_label_down, size = size.normal))
```

```
array.push(all_lines, line.new(bar_index, ep, bar_index + 10, ep, color =  
#5b9cf6, style = line.style_dotted, width = 2))
```

```
array.push(all_lines, line.new(bar_index, sl, bar_index + 10, sl, color =  
#f23645, style = line.style_dotted, width = 2))
```

```
array.push(all_lines, line.new(bar_index, tp, bar_index + 10, tp, color =  
#089981, style = line.style_dotted, width = 2))
```

```
manage_history()
```

```
alert("Sweep/FVG-123 Entry Identified", alert.freq_once_per_bar_close)
```

```
touchedHigh := false
```

```
touchedLow := false
```

```
fvgObj.done := true
```

```
alertcondition(alertTrigger, title = "Trade Signal", message = "Trade Setup Identified (IFVG,  
FVG-123, or 9 EMA Retrace)")
```