

// 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. Helps avoid "historical" FVG
signals.')

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 before resetting.
Prevents signals from very old level sweeps.')

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

show_labels = input.bool(true, 'Show Entry Labels', group = 'Display Settings', tooltip =
'Toggles the visibility of the "Sweep/FVG-123" setup labels and trade lines.')

// =====

// 🕒 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
```

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

```

is5mChart = timeframe.isintraday and timeframe.multiplier == 5 and timeframe.isminutes
if not is5mChart and barstate.islast

```

```
runtime.error("Designed for 5m charts only.")
```

```
breachBullLogic = ta.crossover(close, pdh) or ta.crossover(close, ash) or  
ta.crossover(close, loh) or ta.crossover(close, o15h) or ta.crossover(close, pwh)
```

```
breachBearLogic = ta.crossunder(close, pdl) or ta.crossunder(close, asl) or  
ta.crossunder(close, lol) or ta.crossunder(close, o15l) or ta.crossunder(close, pwl)
```

```
var bool touchedHigh = false
```

```
var bool touchedLow = false
```

```
if high >= pdh or high >= ash or high >= loh or high >= o15h or high >= pwh
```

```
    touchedHigh := true
```

```
if low <= pdl or low <= asl or low <= lol or low <= o15l or low <= pwl
```

```
    touchedLow := true
```

```
last_high_touch = ta.barssince(high >= pdh or high >= ash or high >= loh or high >= o15h or  
high >= pwh)
```

```
last_low_touch = ta.barssince(low <= pdl or low <= asl or low <= lol or low <= o15l or low <= pwl)
```

```
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
```

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

```
// 💎 FVG-123 Logic
```

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

type FVG

float top

float bottom

bool isBull

bool active

int createdBar

bool touched

bool done

int lastTouchBar

var fvgArray = array.new<FVG>()

if bar_index >= 2

if low > high[2]

array.push(fvgArray, FVG.new(low, high[2], true, true, bar_index, false, false, na))

if high < low[2]

array.push(fvgArray, FVG.new(low[2], high, false, true, bar_index, false, false, na))

if array.size(fvgArray) > 100

array.shift(fvgArray)

atr = ta.atr(14)

var string lastIntention = ""

if breachBullLogic

lastIntention := "Bullish"

else if breachBearLogic

```
lastIntention := "Bearish"
```

```
// This variable will trigger our named alert condition
```

```
bool alertTrigger = false
```

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

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

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

```
    if fvgObj.active
```

```
      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 <= pdl or low <= asl or low <= lol) :

((is_sweep_setup and touchedHigh) or high >= pdh or high >= ash or high >=
loh)
```

```
bool fvg_valid = (bar_index - fvgObj.createdBar <= fvg_max_age) and

(math.abs(fvgObj.top - fvgObj.bottom) / syminfo.mintick >= fvg_min_width)
```

```
if is_sweep_setup and levelEngaged and fvg_valid and barstate.isconfirmed

alertTrigger := true

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)
```

```
color text_col = #5b9cf6
```

```
string promptBase = "Sweep/FVG-123"
```

```
float textY = fvgObj.isBull ? low - atr * 2.8 : high + atr * 2.8
```

```
if show_labels
```

```
label.new(bar_index, textY, promptBase + (fvgObj.isBull ? " ▲ " : " ▼ "), color =
#00000000, textcolor = text_col, style = fvgObj.isBull ? label.style_label_up :
label.style_label_down, size = size.normal)
```

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

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

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

```
alert("5 min sweep only", alert.freq_once_per_bar_close)
```

```
touchedHigh := false
```

```
touchedLow := false
```

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

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

```
// 🛎 Named Alert Condition
```

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

```
alertcondition(alertTrigger, title = "5 min sweep only", message = "5 min sweep only")
```