

```

// 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('Universal MFB - Flow/Sweep/FVG-123 Setup', 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')
fvg_min_width = input.float(0, 'Min FVG Width (Ticks)', minval = 0, group = 'FVG Settings')
priming_lookback = input.int(30, 'Level Touch Lookback', minval = 1, group = 'Logic
Settings')

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

show_dots_hrf = input.bool(true, 'Show HTF Intention Dots', group = 'Display Settings')
show_levels = input.bool(true, 'Show Key Levels (PDH/PDL/etc.)', group = 'Display Settings')
show_labels = input.bool(true, 'Show Entry Labels (Chd/FVG-123)', group = 'Display
Settings')

// --- Group Name: Style Settings ---

pdh_color = input.color(color.new(#ab47bc, 30), 'PDH/PDL Color', group = 'Style Settings',
inline = 'd')

pwh_color = input.color(color.new(#5b9cf6, 30), 'PWH/PWL Color', group = 'Style Settings',
inline = 'w')

sess_color = input.color(color.new(color.gray, 50), 'Session Levels Color', group = 'Style
Settings', inline = 's')

// =====

```

```
// 🚫 Levels Logic
```

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

```
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
```

```
    t_change = ta.change(time('D', 'America/New_York'))
```

```
    if t_change != 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')
```

```
[loh, lol] = get_bounds('0000-0600')
```

```
[o15h, o15l] = get_bounds('0930-0945')
```

```
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)
```

```
is_sb_time = not na(time('D', '0300-0400', 'America/New_York')) or not na(time('D', '1000-1100', 'America/New_York')) or not na(time('D', '1400-1500', 'America/New_York'))
```

```
// Plotting
```

```
t_day_change = ta.change(time('D'))
```

```
plot(show_levels ? (t_day_change != 0 ? na : pdh) : na, 'PDH', color = pdh_color, style = plot.style_linebr, linewidth = 2)
```

```
plot(show_levels ? (t_day_change != 0 ? na : pdl) : na, 'PDL', color = pdh_color, style = plot.style_linebr, linewidth = 2)
```

```
plot(show_levels ? ash : na, 'Asian High', color = sess_color, style = plot.style_linebr)
```

```
plot(show_levels ? asl : na, 'Asian Low', color = sess_color, style = plot.style_linebr)
```

```
// Right-side Labels
```

```
var label lbl_pdh = na, var label lbl_pdl = na
```

```
if barstate.islast and show_levels
```

```
    label.delete(lbl_pdh), label.delete(lbl_pdl)
```

```
    lbl_pdh := label.new(bar_index + 2, pdh, "PDH", color = #00000000, textcolor = pdh_color, style = label.style_label_left, size = size.small)
```

```
    lbl_pdl := label.new(bar_index + 2, pdl, "PDL", color = #00000000, textcolor = pdh_color, style = label.style_label_left, size = size.small)
```

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

```
// ⚡ Intention & HTF Support Logic
```

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

```
is1mChart = timeframe.isintraday and timeframe.multiplier == 1
```

isHTF = timeframe.isintraday and timeframe.multiplier >= 5

breachBull = ta.crossover(close, pdh) or ta.crossover(close, ash) or ta.crossover(close, pwh)

breachBear = ta.crossunder(close, pdl) or ta.crossunder(close, asl) or ta.crossunder(close, pwl)

touchHigh = high >= pdh or high >= ash or high >= pwh

touchLow = low <= pdl or low <= asl or low <= pwl

// Get 5m data for 1m chart support

[bBull5m, bBear5m, tHigh5m, tLow5m] = request.security(syminfo.tickerid, "5",  
[breachBull, breachBear, touchHigh, touchLow])

// Effective logic: use 5m data if on 1m chart

effBreachBull = is1mChart ? bBull5m : breachBull

effBreachBear = is1mChart ? bBear5m : breachBear

effTouchHigh = is1mChart ? tHigh5m : touchHigh

effTouchLow = is1mChart ? tLow5m : touchLow

var string lastIntention = ""

if effBreachBull

lastIntention := "Bullish"

else if effBreachBear

lastIntention := "Bearish"

var bool primedHigh = false

var bool primedLow = false

```
if effTouchHigh
```

```
    primedHigh := true
```

```
if effTouchLow
```

```
    primedLow := true
```

```
bars_since_touch_high = ta.barssince(effTouchHigh)
```

```
bars_since_touch_low = ta.barssince(effTouchLow)
```

```
if not na(bars_since_touch_high) and bars_since_touch_high > priming_lookback
```

```
    primedHigh := false
```

```
if not na(bars_since_touch_low) and bars_since_touch_low > priming_lookback
```

```
    primedLow := false
```

```
// Dots (HTF Intention)
```

```
t_5m_change = ta.change(time("5"))
```

```
is_htf_change = is1mChart ? t_5m_change != 0 : true
```

```
plotshape(show_dots_htf and is_htf_change and effBreachBull, "Bull Intention",  
shape.circle, location.belowbar, color.purple, size=size.tiny)
```

```
plotshape(show_dots_htf and is_htf_change and effBreachBear, "Bear Intention",  
shape.circle, location.abovebar, color.purple, size=size.tiny)
```

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

```
// 💎 CHoCH & CISD (Ch'd) Logic
```

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

```
var float lastSwingHigh = na
```

```
var float lastSwingLow = na
```

```
ph = ta.pivohigh(high, 3, 3)
```

```
pl = ta.pivotlow(low, 3, 3)
```

```
if not na(ph)
```

```
    lastSwingHigh := ph
```

```
if not na(pl)
```

```
    lastSwingLow := pl
```

```
var float lastBearFVGTop = na
```

```
var float lastBullFVGBot = na
```

```
if high < low[2]
```

```
    lastBearFVGTop := low[2]
```

```
if low > high[2]
```

```
    lastBullFVGBot := high[2]
```

```
x_swing_high = ta.crossover(close, lastSwingHigh)
```

```
x_fvg_top  = ta.crossover(close, lastBearFVGTop)
```

```
u_swing_low = ta.crossunder(close, lastSwingLow)
```

```
u_fvg_bot  = ta.crossunder(close, lastBullFVGBot)
```

```
bool bullChd = is1mChart and primedLow and (x_swing_high or x_fvg_top)
```

```
bool bearChd = is1mChart and primedHigh and (u_swing_low or u_fvg_bot)
```

```
plotshape(show_labels and bullChd, "Bull Ch'd", shape.diamond, location.belowbar,  
#089981, text="Ch'd", textcolor=#089981, size=size.small)
```

```
plotshape(show_labels and bearChd, "Bear Ch'd", shape.diamond, location.abovebar,  
#f23645, text="Ch'd", textcolor=#f23645, size=size.small)
```

```

var bool chd_active_bull = false
var bool chd_active_bear = false
if bullChd
    chd_active_bull := true, primedLow := false
if bearChd
    chd_active_bear := true, primedHigh := 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]
        fvgArray.push(FVG.new(low, high[2], true, true, bar_index, false, false, na))
    if high < low[2]
        fvgArray.push(FVG.new(low[2], high, false, true, bar_index, false, false, na))

```

```

if fvgArray.size() > 100
    fvgArray.shift()

atr = ta.atr(14)

if fvgArray.size() > 0
    for i = fvgArray.size() - 1 to 0
        fvgObj = fvgArray.get(i)
        if fvgObj.active and (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)
                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

            if fvgObj.touched and bar_index > fvgObj.lastTouchBar and ((fvgObj.isBull and close >
fvgObj.top) or (not fvgObj.isBull and close < fvgObj.bottom))
                bool is_flow = fvgObj.isBull ? (lastIntention == "Bullish") : (lastIntention ==
"Bearish")

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

```



```
bool valid = is1mChart and (fvgObj.isBull ? (is_flow and primedHigh) or (is_sweep
and primedLow) or chd_active_bull : (is_flow and primedLow) or (is_sweep and
primedHigh) or chd_active_bear)
```

```
if valid and (bar_index - fvgObj.createdBar <= fvg_max_age) and
barstate.isconfirmed
```

```
float ep = close
```

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

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

```
string p = is_flow ? "Flow-FVG-123" : (is_sb_time ? "SB-Sweep/FVG-123" :
"Sweep/FVG-123")
```

```
if show_labels
```

```
label.new(bar_index, fvgObj.isBull ? low - atr*2 : high + atr*2, p + (fvgObj.isBull
? " ▲" : " ▼"), color=#00000000, textcolor=#5b9cf6, style=fvgObj.isBull ?
label.style_label_up : label.style_label_down)
```

```
// Gradient Box Representation
```

```
box.new(bar_index, tp, bar_index + 15, sl, border_color=color.new(#5b9cf6, 80),
bgcolor=color.new(fvgObj.isBull ? #089981 : #f23645, 90))
```

```
// Clean dotted lines
```

```
line.new(bar_index, ep, bar_index + 15, ep, color=#5b9cf6,
style=line.style_dashed, width=2)
```

```
line.new(bar_index, sl, bar_index + 15, sl, color=#f23645,
style=line.style_dotted, width=4)
```

```
line.new(bar_index, tp, bar_index + 15, tp, color=#089981,
style=line.style_dotted, width=4)
```

```
alert(p + " Entry Confirmed", alert.freq_once_per_bar_close)
```

```
        primedHigh := false, primedLow := false, chd_active_bull := false,  
chd_active_bear := false
```

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