

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

//@version=6

indicator('MFB - Universal I.B.', overlay = true, max_labels_count = 500, max_lines_count = 500)

// --- Constants ---

const string TIMEZONE = 'America/New_York'

// --- Inputs ---

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

fvgMinWidthInput = input.float(0, 'Min FVG Width (Ticks)', minval = 0, group = 'FVG Settings', tooltip = 'Minimum FVG size in ticks.')

onlyFvgAfterKeyTouchInput = input.bool(true, 'Only FVG-123 After Key-Level Continuation', group = 'Logic Settings', tooltip = 'When enabled, FVG-123 trade plans require a recent requested-key-level touch followed by continuation through that level. Sweep-plus-FVG reversals are excluded.')

onlyIbResponsesInput = input.bool(true, 'Restrict Trade Logic To Requested Key Levels', group = 'Logic Settings', tooltip = 'When enabled, CISD and FVG-123 trade plans use PDH, PDL, PWH, PWL, the three I.B. ranges, and the completed Asian, London, and New York full-session ranges. Disable to also include the 15m and 30m opening ranges.')

levelTouchLookbackInput = input.int(30, 'Level Touch Lookback', minval = 1, group = 'Logic Settings', tooltip = 'Number of bars for which a key-level touch remains active.')

```
londonSessionInput = input.session('0000-0100', 'London I.B. (New York Time)', group = 'Initial Balance Sessions', tooltip = 'Default is 00:00–01:00 New York time.')
```

```
asianSessionInput = input.session('1800-1900', 'Asian I.B. (New York Time)', group = 'Initial Balance Sessions', tooltip = 'Default is 18:00–19:00 New York time.')
```

```
nySessionInput = input.session('0930-1030', 'New York I.B. (New York Time)', group = 'Initial Balance Sessions', tooltip = 'Default is 09:30–10:30 New York time.')
```

```
londonFullSessionInput = input.session('0000-0600', 'London Full Session', group = 'Initial Balance Sessions', tooltip = 'The full London window controls how long the London I.B. red lines remain visible. Default is 00:00–06:00 New York time.')
```

```
asianFullSessionInput = input.session('1800-0000', 'Asian Full Session', group = 'Initial Balance Sessions', tooltip = 'The full Asian window controls how long the Asian I.B. red lines remain visible. Default is 18:00–00:00 New York time.')
```

```
nyFullSessionInput = input.session('0930-1600', 'New York Full Session', group = 'Initial Balance Sessions', tooltip = 'The full New York window controls how long the New York I.B. red lines remain visible. Default is 09:30–16:00 New York time.')
```

```
showDots1hInput = input.bool(false, 'Show Intention Dots on 1h Chart', group = 'Display Settings', tooltip = 'Displays purple dots when price breaches key levels on the 1-hour chart. Hidden by default.')
```

```
showDots30mInput = input.bool(false, 'Show Intention Dots on 30m Chart', group = 'Display Settings', tooltip = 'Displays purple dots when price breaches key levels on the 30-minute chart. Hidden by default.')
```

showDots15mInput = input.bool(false, 'Show Intention Dots on 15m Chart',
group = 'Display Settings', tooltip = 'Displays purple dots when price breaches
key levels on the 15-minute chart. Hidden by default.')

showDots5mInput = input.bool(false, 'Show Intention Dots on 5m Chart',
group = 'Display Settings', tooltip = 'Displays purple dots when price breaches
key levels on the 5-minute chart. Hidden by default.')

showDots1mInput = input.bool(false, 'Show Intention Dots on 1m/3m Chart',
group = 'Display Settings', tooltip = 'Displays purple dots on 1m and 3m charts
when a 5-minute breach occurred.')

showLevelsInput = input.bool(true, 'Show Level Plots', group = 'Display
Settings', tooltip = 'Master switch for the level plots.')

showOnlyIbLevelsInput = input.bool(true, 'Show Only Initial Balance Levels',
group = 'Display Settings', tooltip = 'When enabled, only the three red Initial
Balance high/low pairs are displayed. Other levels remain available to the
trade logic.')

showFullSessionLevelsInput = input.bool(false, 'Show Full Session
Highs/Lows', group = 'Display Settings', tooltip = 'Displays the full-session
high/low reference levels. These remain hidden by default; they are still used
internally to end the I.B. line display.')

showLabelsInput = input.bool(true, 'Show Setup Labels', group = 'Display
Settings', tooltip = 'Shows Chd, CISD, Flow, Sweep, Acceptance, and FVG-123
labels.')

showTradeLinesInput = input.bool(true, 'Show Qualified Trade Entry Lines',
group = 'Display Settings', tooltip = 'Shows entry, stop, and target lines only for
qualified session-I.B. CISD or key-level FVG-123 responses.')

tradeProjectionInput = input.int(10, 'Trade Line Projection (Bars)', minval = 1,
maxval = 100, group = 'Display Settings', tooltip = 'Number of bars for which
trade-plan lines are projected.')

```
entryLineWidthInput = input.int(2, 'Trade Entry Line Width', minval = 1, maxval
= 10, group = 'Display Settings', tooltip = 'Thickness of entry, stop, and target
lines.')
```

```
pdhColorInput = input.color(color.new(#ab47bc, 30), 'PDH/PDL Color', group
= 'Style', inline = 'daily')
```

```
pwhColorInput = input.color(color.new(#5b9cf6, 30), 'PWH/PWL Color', group
= 'Style', inline = 'weekly')
```

```
sessionColorInput = input.color(color.new(color.gray, 50), 'Opening-Range
Color', group = 'Style', inline = 'session')
```

```
ibColorInput = input.color(color.new(#f23645, 0), 'I.B. Color', group = 'Style',
inline = 'ib')
```

```
// --- Session and level functions ---
```

```
isInSession(string sessionInput) =>
```

```
    not na(time(timeframe.period, sessionInput, TIMEZONE))
```

```
getBounds(string sessionInput) =>
```

```
    var float sessionHigh = na
```

```
    var float sessionLow = na
```

```
    bool inSession = isInSession(sessionInput)
```

```
    bool newSession = inSession and not inSession[1]
```

```
    if newSession
```

```
        sessionHigh := high
```

```
        sessionLow := low
```

else if inSession

sessionHigh := na(sessionHigh) ? high : math.max(sessionHigh, high)

sessionLow := na(sessionLow) ? low : math.min(sessionLow, low)

[sessionHigh, sessionLow]

isAtOrAbove(float level) =>

not na(level) and high >= level

isAtOrBelow(float level) =>

not na(level) and low <= level

// --- Session levels ---

[asianlbHigh, asianlbLow] = getBounds(asianSessionInput)

[londonlbHigh, londonlbLow] = getBounds(londonSessionInput)

[nylbHigh, nylbLow] = getBounds(nySessionInput)

[asianFullHigh, asianFullLow] = getBounds(asianFullSessionInput)

[londonFullHigh, londonFullLow] = getBounds(londonFullSessionInput)

[nyFullHigh, nyFullLow] = getBounds(nyFullSessionInput)

[o15High, o15Low] = getBounds('0930-0945')

[o30High, o30Low] = getBounds('0930-1000')

inAsianBuild = isInSession(asianSessionInput)

inLondonBuild = isInSession(londonSessionInput)

```
inNyBuild = isInSession(nySessionInput)

inAsianFull = isInSession(asianFullSessionInput)

inLondonFull = isInSession(londonFullSessionInput)

inNyFull = isInSession(nyFullSessionInput)

asianIbReady = not inAsianBuild and not na(asianIbHigh) and not
na(asianIbLow)

londonIbReady = not inLondonBuild and not na(londonIbHigh) and not
na(londonIbLow)

nyIbReady = not inNyBuild and not na(nyIbHigh) and not na(nyIbLow)

asianFullReady = not inAsianFull and not na(asianFullHigh) and not
na(asianFullLow)

londonFullReady = not inLondonFull and not na(londonFullHigh) and not
na(londonFullLow)

nyFullReady = not inNyFull and not na(nyFullHigh) and not na(nyFullLow)

asianIbActive = asianIbReady and inAsianFull

londonIbActive = londonIbReady and inLondonFull

nyIbActive = nyIbReady and inNyFull
```

```
// --- Higher-timeframe levels ---
```

```
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)
```

```
// --- Key-level breach events ---
```

```
nyHighCrossRaw = ta.crossover(close, nylbHigh)
```

```
nyLowCrossRaw = ta.crossunder(close, nylbLow)
```

```
londonHighCrossRaw = ta.crossover(close, londonlbHigh)
```

```
londonLowCrossRaw = ta.crossunder(close, londonlbLow)
```

```
asianHighCrossRaw = ta.crossover(close, asianlbHigh)
```

```
asianLowCrossRaw = ta.crossunder(close, asianlbLow)
```

```
asianFullHighCrossRaw = ta.crossover(close, asianFullHigh)
```

```
asianFullLowCrossRaw = ta.crossunder(close, asianFullLow)
```

```
londonFullHighCrossRaw = ta.crossover(close, londonFullHigh)
```

```
londonFullLowCrossRaw = ta.crossunder(close, londonFullLow)
```

```
nyFullHighCrossRaw = ta.crossover(close, nyFullHigh)
```

```
nyFullLowCrossRaw = ta.crossunder(close, nyFullLow)
```

```
o15HighCross = ta.crossover(close, o15High)
```

```
o15LowCross = ta.crossunder(close, o15Low)
```

```
o30HighCross = ta.crossover(close, o30High)
```

```
o30LowCross = ta.crossunder(close, o30Low)
```

```
pdhCross = ta.crossover(close, pdh)
```

```
pdlCross = ta.crossunder(close, pdl)
```

```
pwhCross = ta.crossover(close, pwh)
```

pwlCross = ta.crossunder(close, pwl)

nyHighCross = nyIbActive and nyHighCrossRaw

nyLowCross = nyIbActive and nyLowCrossRaw

londonHighCross = londonIbActive and londonHighCrossRaw

londonLowCross = londonIbActive and londonLowCrossRaw

asianHighCross = asianIbActive and asianHighCrossRaw

asianLowCross = asianIbActive and asianLowCrossRaw

asianFullHighCross = asianFullReady and asianFullHighCrossRaw

asianFullLowCross = asianFullReady and asianFullLowCrossRaw

londonFullHighCross = londonFullReady and londonFullHighCrossRaw

londonFullLowCross = londonFullReady and londonFullLowCrossRaw

nyFullHighCross = nyFullReady and nyFullHighCrossRaw

nyFullLowCross = nyFullReady and nyFullLowCrossRaw

requestedBullBreach = pdhCross or pwhCross or nyHighCross or
londonHighCross or asianHighCross or asianFullHighCross or
londonFullHighCross or nyFullHighCross

requestedBearBreach = pdlCross or pwlCross or nyLowCross or
londonLowCross or asianLowCross or asianFullLowCross or
londonFullLowCross or nyFullLowCross

bullBreachLogic = onlyIbResponsesInput ? requestedBullBreach :
requestedBullBreach or o15HighCross or o30HighCross

bearBreachLogic = onlyIbResponsesInput ? requestedBearBreach :
requestedBearBreach or o15LowCross or o30LowCross

```

var int lastBullBreachBar = na
var int lastBearBreachBar = na
var float lastBullBreachLevel = na
var float lastBearBreachLevel = na
if bullBreachLogic
    lastBullBreachBar := bar_index
    lastBullBreachLevel := na
    if pdhCross
        lastBullBreachLevel := pdh
    if pwhCross
        lastBullBreachLevel := na(lastBullBreachLevel) ? pwh :
math.max(lastBullBreachLevel, pwh)
    if nyHighCross
        lastBullBreachLevel := na(lastBullBreachLevel) ? nyIbHigh :
math.max(lastBullBreachLevel, nyIbHigh)
    if londonHighCross
        lastBullBreachLevel := na(lastBullBreachLevel) ? londonIbHigh :
math.max(lastBullBreachLevel, londonIbHigh)
    if asianHighCross
        lastBullBreachLevel := na(lastBullBreachLevel) ? asianIbHigh :
math.max(lastBullBreachLevel, asianIbHigh)
    if asianFullHighCross
        lastBullBreachLevel := na(lastBullBreachLevel) ? asianFullHigh :
math.max(lastBullBreachLevel, asianFullHigh)

```

```

if londonFullHighCross

    lastBullBreachLevel := na(lastBullBreachLevel) ? londonFullHigh :
math.max(lastBullBreachLevel, londonFullHigh)

if nyFullHighCross

    lastBullBreachLevel := na(lastBullBreachLevel) ? nyFullHigh :
math.max(lastBullBreachLevel, nyFullHigh)

if o15HighCross

    lastBullBreachLevel := na(lastBullBreachLevel) ? o15High :
math.max(lastBullBreachLevel, o15High)

if o30HighCross

    lastBullBreachLevel := na(lastBullBreachLevel) ? o30High :
math.max(lastBullBreachLevel, o30High)

if bearBreachLogic

    lastBearBreachBar := bar_index

    lastBearBreachLevel := na

    if pdlCross

        lastBearBreachLevel := pdl

    if pwlCross

        lastBearBreachLevel := na(lastBearBreachLevel) ? pwl :
math.min(lastBearBreachLevel, pwl)

    if nyLowCross

        lastBearBreachLevel := na(lastBearBreachLevel) ? nyLow :
math.min(lastBearBreachLevel, nyLow)

    if londonLowCross

```

```
lastBearBreachLevel := na(lastBearBreachLevel) ? londonlbLow :  
math.min(lastBearBreachLevel, londonlbLow)
```

```
if asianLowCross
```

```
lastBearBreachLevel := na(lastBearBreachLevel) ? asianlbLow :  
math.min(lastBearBreachLevel, asianlbLow)
```

```
if asianFullLowCross
```

```
lastBearBreachLevel := na(lastBearBreachLevel) ? asianFullLow :  
math.min(lastBearBreachLevel, asianFullLow)
```

```
if londonFullLowCross
```

```
lastBearBreachLevel := na(lastBearBreachLevel) ? londonFullLow :  
math.min(lastBearBreachLevel, londonFullLow)
```

```
if nyFullLowCross
```

```
lastBearBreachLevel := na(lastBearBreachLevel) ? nyFullLow :  
math.min(lastBearBreachLevel, nyFullLow)
```

```
if o15LowCross
```

```
lastBearBreachLevel := na(lastBearBreachLevel) ? o15Low :  
math.min(lastBearBreachLevel, o15Low)
```

```
if o30LowCross
```

```
lastBearBreachLevel := na(lastBearBreachLevel) ? o30Low :  
math.min(lastBearBreachLevel, o30Low)
```

```
// --- Key-level touch tracking ---
```

```
highKeyTouch = isAtOrAbove(pdh) or isAtOrAbove(pwh) or  
isAtOrAbove(o15High) or isAtOrAbove(o30High) or (nylbActive and  
isAtOrAbove(nylbHigh)) or (londonlbActive and isAtOrAbove(londonlbHigh)) or  
(asianlbActive and isAtOrAbove(asianlbHigh)) or (nyFullReady and
```

isAtOrAbove(nyFullHigh)) or (londonFullReady and
isAtOrAbove(londonFullHigh)) or (asianFullReady and
isAtOrAbove(asianFullHigh))

lowKeyTouch = isAtOrBelow(pdl) or isAtOrBelow(pwl) or isAtOrBelow(o15Low)
or isAtOrBelow(o30Low) or (nylbActive and isAtOrBelow(nylbLow)) or
(londonlbActive and isAtOrBelow(londonlbLow)) or (asianlbActive and
isAtOrBelow(asianlbLow)) or (nyFullReady and isAtOrBelow(nyFullLow)) or
(londonFullReady and isAtOrBelow(londonFullLow)) or (asianFullReady and
isAtOrBelow(asianFullLow))

ibHighTouch = (nylbActive and isAtOrAbove(nylbHigh)) or (londonlbActive and
isAtOrAbove(londonlbHigh)) or (asianlbActive and isAtOrAbove(asianlbHigh))

ibLowTouch = (nylbActive and isAtOrBelow(nylbLow)) or (londonlbActive and
isAtOrBelow(londonlbLow)) or (asianlbActive and isAtOrBelow(asianlbLow))

fullHighTouch = (nyFullReady and isAtOrAbove(nyFullHigh)) or
(londonFullReady and isAtOrAbove(londonFullHigh)) or (asianFullReady and
isAtOrAbove(asianFullHigh))

fullLowTouch = (nyFullReady and isAtOrBelow(nyFullLow)) or
(londonFullReady and isAtOrBelow(londonFullLow)) or (asianFullReady and
isAtOrBelow(asianFullLow))

requestedHighTouch = isAtOrAbove(pdh) or isAtOrAbove(pwh) or ibHighTouch
or fullHighTouch

requestedLowTouch = isAtOrBelow(pdl) or isAtOrBelow(pwl) or ibLowTouch or
fullLowTouch

lastHighTouch = ta.barssince(highKeyTouch)

lastLowTouch = ta.barssince(lowKeyTouch)

lastlbHighTouch = ta.barssince(ibHighTouch)

lastlbLowTouch = ta.barssince(ibLowTouch)

lastFullHighTouch = ta.barssince(fullHighTouch)

lastFullLowTouch = ta.barssince(fullLowTouch)

lastRequestedHighTouch = ta.barssince(requestedHighTouch)

lastRequestedLowTouch = ta.barssince(requestedLowTouch)

recentHighTouch = not na(lastHighTouch) and lastHighTouch <= levelTouchLookbackInput

recentLowTouch = not na(lastLowTouch) and lastLowTouch <= levelTouchLookbackInput

recentlbHighTouch = not na(lastlbHighTouch) and lastlbHighTouch <= levelTouchLookbackInput

recentlbLowTouch = not na(lastlbLowTouch) and lastlbLowTouch <= levelTouchLookbackInput

recentFullHighTouch = not na(lastFullHighTouch) and lastFullHighTouch <= levelTouchLookbackInput

recentFullLowTouch = not na(lastFullLowTouch) and lastFullLowTouch <= levelTouchLookbackInput

recentRequestedHighTouch = not na(lastRequestedHighTouch) and lastRequestedHighTouch <= levelTouchLookbackInput

recentRequestedLowTouch = not na(lastRequestedLowTouch) and lastRequestedLowTouch <= levelTouchLookbackInput

recentlbTouch = recentlbHighTouch or recentlbLowTouch

recentFullTouch = recentFullHighTouch or recentFullLowTouch

recentSessionHighTouch = recentRequestedHighTouch

recentSessionLowTouch = recentRequestedLowTouch

recentSessionTouch = recentSessionHighTouch or recentSessionLowTouch

recentKeyTouch = recentHighTouch or recentLowTouch

```
// --- Plot levels ---
```

```
showOtherLevels = showLevelsInput and not showOnlyIbLevelsInput
```

```
plot(showOtherLevels ? pdh : na, 'PDH', color = pdhColorInput, style =  
plot.style_linebr, linewidth = 2)
```

```
plot(showOtherLevels ? pdl : na, 'PDL', color = pdhColorInput, style =  
plot.style_linebr, linewidth = 2)
```

```
plot(showOtherLevels ? pwh : na, 'PWH', color = pwhColorInput, style =  
plot.style_linebr, linewidth = 2)
```

```
plot(showOtherLevels ? pwl : na, 'PWL', color = pwhColorInput, style =  
plot.style_linebr, linewidth = 2)
```

```
// I.B. red lines are visible only after the measuring window and only through  
the full session close.
```

```
plot(showLevelsInput and asianIbActive ? asianIbHigh : na, 'Asian I.B. High  
(18:00-19:00)', color = ibColorInput, style = plot.style_linebr, linewidth = 2)
```

```
plot(showLevelsInput and asianIbActive ? asianIbLow : na, 'Asian I.B. Low  
(18:00-19:00)', color = ibColorInput, style = plot.style_linebr, linewidth = 2)
```

```
plot(showLevelsInput and londonIbActive ? londonIbHigh : na, 'London I.B.  
High (00:00-01:00)', color = ibColorInput, style = plot.style_linebr, linewidth =  
2)
```

```
plot(showLevelsInput and londonIbActive ? londonIbLow : na, 'London I.B.  
Low (00:00-01:00)', color = ibColorInput, style = plot.style_linebr, linewidth =  
2)
```

```
plot(showLevelsInput and nylbActive ? nylbHigh : na, 'New York I.B. High  
(09:30-10:30)', color = ibColorInput, style = plot.style_linebr, linewidth = 2)
```

```
plot(showLevelsInput and nyIbActive ? nyIbLow : na, 'New York I.B. Low  
(09:30-10:30)', color = ibColorInput, style = plot.style_linebr, linewidth = 2)
```

```
// Full-session levels are hidden by default but can be restored independently.
```

```
plot(showOtherLevels and showFullSessionLevelsInput and inAsianFull ?  
asianFullHigh : na, 'Asian Full-Session High (18:00-00:00)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels and showFullSessionLevelsInput and inAsianFull ?  
asianFullLow : na, 'Asian Full-Session Low (18:00-00:00)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels and showFullSessionLevelsInput and inLondonFull ?  
londonFullHigh : na, 'London Full-Session High (00:00-06:00)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels and showFullSessionLevelsInput and inLondonFull ?  
londonFullLow : na, 'London Full-Session Low (00:00-06:00)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels and showFullSessionLevelsInput and inNyFull ?  
nyFullHigh : na, 'New York Full-Session High (09:30-16:00)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels and showFullSessionLevelsInput and inNyFull ?  
nyFullLow : na, 'New York Full-Session Low (09:30-16:00)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels ? o15High : na, '9:30 Open High (15m)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels ? o15Low : na, '9:30 Open Low (15m)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels ? o30High : na, '9:30 Open High (30m)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
plot(showOtherLevels ? o30Low : na, '9:30 Open Low (30m)', color =  
sessionColorInput, style = plot.style_linebr)
```

```
// --- Right-side level labels ---
```

```
var label pdhLabel = na
```

```
var label pdlLabel = na
```

```
var label pwhLabel = na
```

```
var label pwlLabel = na
```

```
var label asianHighLabel = na
```

```
var label asianLowLabel = na
```

```
var label londonHighLabel = na
```

```
var label londonLowLabel = na
```

```
var label nyHighLabel = na
```

```
var label nyLowLabel = na
```

```
if barstate.islast
```

```
  label.delete(pdhLabel)
```

```
  label.delete(pdlLabel)
```

```
  label.delete(pwhLabel)
```

```
  label.delete(pwlLabel)
```

```
  label.delete(asianHighLabel)
```

```
  label.delete(asianLowLabel)
```

```

label.delete(londonHighLabel)

label.delete(londonLowLabel)

label.delete(nyHighLabel)

label.delete(nyLowLabel)

if showOtherLevels

    if not na(pdh)

        pdhLabel := label.new(bar_index + 2, pdh, 'PDH', color = #00000000,
textcolor = pdhColorInput, style = label.style_label_left, size = size.small)

    if not na(pdl)

        pdlLabel := label.new(bar_index + 2, pdl, 'PDL', color = #00000000,
textcolor = pdhColorInput, style = label.style_label_left, size = size.small)

    if not na(pwh)

        pwhLabel := label.new(bar_index + 2, pwh, 'PWH', color = #00000000,
textcolor = pwhColorInput, style = label.style_label_left, size = size.small)

    if not na(pwl)

        pwlLabel := label.new(bar_index + 2, pwl, 'PWL', color = #00000000,
textcolor = pwhColorInput, style = label.style_label_left, size = size.small)

if showLevelsInput

    if asianIbActive

        asianHighLabel := label.new(bar_index + 2, asianIbHigh, 'Asian I.B. High
18:00-19:00', color = #00000000, textcolor = ibColorInput, style =
label.style_label_left, size = size.tiny)

        asianLowLabel := label.new(bar_index + 2, asianIbLow, 'Asian I.B. Low
18:00-19:00', color = #00000000, textcolor = ibColorInput, style =
label.style_label_left, size = size.tiny)

    if londonIbActive

```

```
londonHighLabel := label.new(bar_index + 2, londonIbHigh, 'London I.B.  
High 00:00-01:00', color = #00000000, textcolor = ibColorInput, style =  
label.style_label_left, size = size.tiny)
```

```
londonLowLabel := label.new(bar_index + 2, londonIbLow, 'London I.B.  
Low 00:00-01:00', color = #00000000, textcolor = ibColorInput, style =  
label.style_label_left, size = size.tiny)
```

```
if nylbActive
```

```
nyHighLabel := label.new(bar_index + 2, nylbHigh, 'NY I.B. High 09:30-  
10:30', color = #00000000, textcolor = ibColorInput, style =  
label.style_label_left, size = size.tiny)
```

```
nyLowLabel := label.new(bar_index + 2, nylbLow, 'NY I.B. Low 09:30-  
10:30', color = #00000000, textcolor = ibColorInput, style =  
label.style_label_left, size = size.tiny)
```

```
// --- Intention dots ---
```

```
is1hChart = timeframe.isintraday and timeframe.multiplier == 60 and  
timeframe.isminutes
```

```
is30mChart = timeframe.isintraday and timeframe.multiplier == 30 and  
timeframe.isminutes
```

```
is15mChart = timeframe.isintraday and timeframe.multiplier == 15 and  
timeframe.isminutes
```

```
is5mChart = timeframe.isintraday and timeframe.multiplier == 5 and  
timeframe.isminutes
```

```
is3mChart = timeframe.isintraday and timeframe.multiplier == 3 and  
timeframe.isminutes
```

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

```
isSupportedChart = timeframe.isintraday and timeframe.isminutes and  
(timeframe.multiplier == 1 or timeframe.multiplier == 3 or  
timeframe.multiplier == 5 or timeframe.multiplier == 15 or  
timeframe.multiplier == 30 or timeframe.multiplier == 60)
```

```
[breachBull5m, breachBear5m] = request.security(syminfo.tickerid, '5',  
[bullBreachLogic, bearBreachLogic])
```

```
is5mEnd = ta.change(time('5')) != 0
```

```
plotBullDot = (is1hChart and showDots1hInput and bullBreachLogic) or  
(is30mChart and showDots30mInput and bullBreachLogic) or (is15mChart  
and showDots15mInput and bullBreachLogic) or (is5mChart and  
showDots5mInput and bullBreachLogic) or ((is1mChart or is3mChart) and  
showDots1mInput and is5mEnd and breachBull5m)
```

```
plotBearDot = (is1hChart and showDots1hInput and bearBreachLogic) or  
(is30mChart and showDots30mInput and bearBreachLogic) or (is15mChart  
and showDots15mInput and bearBreachLogic) or (is5mChart and  
showDots5mInput and bearBreachLogic) or ((is1mChart or is3mChart) and  
showDots1mInput and is5mEnd and breachBear5m)
```

```
plotshape(plotBullDot, title = 'Bullish Breakout Dot', style = shape.circle,  
location = location.belowbar, color = color.purple, size = size.tiny)
```

```
plotshape(plotBearDot, title = 'Bearish Breakout Dot', style = shape.circle,  
location = location.abovebar, color = color.purple, size = size.tiny)
```

```
// --- Chd / lower-timeframe CISD logic ---
```

```
var float lastSwingHigh = na
```

```
var float lastSwingLow = na
```

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

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

```

if not na(pivotHigh)
    lastSwingHigh := pivotHigh
if not na(pivotLow)
    lastSwingLow := pivotLow
var float lastBearFvgTop = na
var float lastBullFvgBottom = na
if high < low[2]
    lastBearFvgTop := low[2]
if low > high[2]
    lastBullFvgBottom := high[2]
chochBullEvent = ta.crossover(close, lastSwingHigh)
chochBearEvent = ta.crossunder(close, lastSwingLow)
cisdBullEvent = ta.crossover(close, lastBearFvgTop)
cisdBearEvent = ta.crossunder(close, lastBullFvgBottom)
lowerTimeframe = is1mChart or is3mChart
bullChd = lowerTimeframe and (onlyIbResponsesInput ?
recentSessionLowTouch : recentLowTouch) and (chochBullEvent or
cisdBullEvent) and close > lastSwingHigh and close > lastBearFvgTop
bearChd = lowerTimeframe and (onlyIbResponsesInput ?
recentSessionHighTouch : recentHighTouch) and (chochBearEvent or
cisdBearEvent) and close < lastSwingLow and close < lastBullFvgBottom
plotshape(showLabelsInput and bullChd, title = 'Bullish Chd', text = 'Chd',
style = shape.diamond, location = location.belowbar, color = #089981,
textcolor = #089981, size = size.small)

```

```
plotshape(showLabelsInput and bearChd, title = 'Bearish Chd', text = 'Chd',  
style = shape.diamond, location = location.abovebar, color = #f23645,  
textcolor = #f23645, size = size.small)
```

```
if bullChd
```

```
    alert('Universal MFB - Bullish Chd key-level response on ' + syminfo.ticker,  
    alert.freq_once_per_bar_close)
```

```
if bearChd
```

```
    alert('Universal MFB - Bearish Chd key-level response on ' + syminfo.ticker,  
    alert.freq_once_per_bar_close)
```

```
// --- 5m session-range CISD logic ---
```

```
var float displacedHighLevel = na
```

```
var float displacedHighLow = na
```

```
var bool highDisplaced = false
```

```
var float displacedLowLevel = na
```

```
var float displacedLowHigh = na
```

```
var bool lowDisplaced = false
```

```
var int displacedHighBar = na
```

```
var int displacedLowBar = na
```

```
if isSupportedChart and bullBreachLogic
```

```
    highDisplaced := true
```

```
    displacedHighBar := bar_index
```

```
    displacedHighLow := low
```

```

displacedHighLevel := na
if pdhCross
    displacedHighLevel := pdh
if pwhCross
    displacedHighLevel := na(displacedHighLevel) ? pwh :
math.max(displacedHighLevel, pwh)
if nyHighCross
    displacedHighLevel := na(displacedHighLevel) ? nyIbHigh :
math.max(displacedHighLevel, nyIbHigh)
if londonHighCross
    displacedHighLevel := na(displacedHighLevel) ? londonIbHigh :
math.max(displacedHighLevel, londonIbHigh)
if asianHighCross
    displacedHighLevel := na(displacedHighLevel) ? asianIbHigh :
math.max(displacedHighLevel, asianIbHigh)
if asianFullHighCross
    displacedHighLevel := na(displacedHighLevel) ? asianFullHigh :
math.max(displacedHighLevel, asianFullHigh)
if londonFullHighCross
    displacedHighLevel := na(displacedHighLevel) ? londonFullHigh :
math.max(displacedHighLevel, londonFullHigh)
if nyFullHighCross
    displacedHighLevel := na(displacedHighLevel) ? nyFullHigh :
math.max(displacedHighLevel, nyFullHigh)
if o15HighCross

```

```
displacedHighLevel := na(displacedHighLevel) ? o15High :  
math.max(displacedHighLevel, o15High)
```

```
if o30HighCross
```

```
displacedHighLevel := na(displacedHighLevel) ? o30High :  
math.max(displacedHighLevel, o30High)
```

```
if isSupportedChart and bearBreachLogic
```

```
lowDisplaced := true
```

```
displacedLowBar := bar_index
```

```
displacedLowHigh := high
```

```
displacedLowLevel := na
```

```
if pdlCross
```

```
displacedLowLevel := pdl
```

```
if pwlCross
```

```
displacedLowLevel := na(displacedLowLevel) ? pwl :  
math.min(displacedLowLevel, pwl)
```

```
if nyLowCross
```

```
displacedLowLevel := na(displacedLowLevel) ? nylbLow :  
math.min(displacedLowLevel, nylbLow)
```

```
if londonLowCross
```

```
displacedLowLevel := na(displacedLowLevel) ? londonlbLow :  
math.min(displacedLowLevel, londonlbLow)
```

```
if asianLowCross
```

```
displacedLowLevel := na(displacedLowLevel) ? asianlbLow :  
math.min(displacedLowLevel, asianlbLow)
```

if asianFullLowCross

displacedLowLevel := na(displacedLowLevel) ? asianFullLow :
math.min(displacedLowLevel, asianFullLow)

if londonFullLowCross

displacedLowLevel := na(displacedLowLevel) ? londonFullLow :
math.min(displacedLowLevel, londonFullLow)

if nyFullLowCross

displacedLowLevel := na(displacedLowLevel) ? nyFullLow :
math.min(displacedLowLevel, nyFullLow)

if o15LowCross

displacedLowLevel := na(displacedLowLevel) ? o15Low :
math.min(displacedLowLevel, o15Low)

if o30LowCross

displacedLowLevel := na(displacedLowLevel) ? o30Low :
math.min(displacedLowLevel, o30Low)

if isSupportedChart

if highDisplaced and not na(displacedHighLevel) and close >
displacedHighLevel and low > displacedHighLow and bar_index -
displacedHighBar > 10

highDisplaced := false

if lowDisplaced and not na(displacedLowLevel) and close <
displacedLowLevel and high < displacedLowHigh and bar_index -
displacedLowBar > 10

lowDisplaced := false

bear5mCisd = isSupportedChart and highDisplaced and not
na(displacedHighLevel) and close < displacedHighLevel and close <
displacedHighLow and barstate.isconfirmed

bull5mCisd = isSupportedChart and lowDisplaced and not
na(displacedLowLevel) and close > displacedLowLevel and close >
displacedLowHigh and barstate.isconfirmed

if bear5mCisd

 highDisplaced := false

 alert('MFB - Universal I.B. - Bearish Session CISD on ' + syminfo.ticker,
 alert.freq_once_per_bar_close)

 if showTradeLinesInput

 entryPrice = close

 stopPrice = high

 risk = math.abs(stopPrice - entryPrice)

 targetTwo = entryPrice - risk * 2

 line.new(bar_index, entryPrice, bar_index + tradeProjectionInput,
entryPrice, color = #5b9cf6, style = line.style_dotted, width =
entryLineWidthInput)

 line.new(bar_index, stopPrice, bar_index + tradeProjectionInput,
stopPrice, color = #f23645, style = line.style_dotted, width =
entryLineWidthInput)

 line.new(bar_index, targetTwo, bar_index + tradeProjectionInput,
targetTwo, color = #089981, style = line.style_dotted, width =
entryLineWidthInput)

```

if bull5mCisd

    lowDisplaced := false

    alert('MFB - Universal I.B. - Bullish Session CISD on ' + syminfo.ticker,
    alert.freq_once_per_bar_close)

    if showTradeLinesInput

        entryPrice = close

        stopPrice = low

        risk = math.abs(entryPrice - stopPrice)

        targetTwo = entryPrice + risk * 2

        line.new(bar_index, entryPrice, bar_index + tradeProjectionInput,
        entryPrice, color = #5b9cf6, style = line.style_dotted, width =
        entryLineWidthInput)

        line.new(bar_index, stopPrice, bar_index + tradeProjectionInput,
        stopPrice, color = #f23645, style = line.style_dotted, width =
        entryLineWidthInput)

        line.new(bar_index, targetTwo, bar_index + tradeProjectionInput,
        targetTwo, color = #089981, style = line.style_dotted, width =
        entryLineWidthInput)

plotshape(showLabelsInput and bear5mCisd, title = 'Bearish Session CISD',
text = 'CISD', style = shape.labeldown, location = location.abovebar, color =
#f23645, textcolor = color.white, size = size.small)

plotshape(showLabelsInput and bull5mCisd, title = 'Bullish Session CISD',
text = 'CISD', style = shape.labelup, location = location.belowbar, color =
#089981, textcolor = color.white, size = size.small)

```

```
// --- FVG-123 logic ---
```

```
type Fvg
```

```
    float top
```

```
    float bottom
```

```
    bool isBull
```

```
    bool active
```

```
    int createdBar
```

```
    int keyTouchBar
```

```
    bool touched
```

```
    bool done
```

```
    int lastTouchBar
```

```
var fvgArray = array.new<Fvg>()
```

```
lastRequestedHighTouchBar = ta.valuwhen(requestedHighTouch, bar_index,  
0)
```

```
lastRequestedLowTouchBar = ta.valuwhen(requestedLowTouch, bar_index,  
0)
```

```
latestRequestedTouchBar = na(lastRequestedHighTouchBar) ?
```

```
lastRequestedLowTouchBar : na(lastRequestedLowTouchBar) ?
```

```
lastRequestedHighTouchBar : math.max(lastRequestedHighTouchBar,  
lastRequestedLowTouchBar)
```

```
if bar_index >= 2
```

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

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

```

    if high < low[2]

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

    if array.size(fvgArray) > 100

        array.shift(fvgArray)

var table setupTable = table.new(position.bottom_right, 1, 1, bgcolor =
color.new(color.black, 70), border_width = 1, border_color = color.gray)

atrValue = ta.atr(14)

var string lastIntention = "

if bullBreachLogic

    lastIntention := 'Bullish'

else if bearBreachLogic

    lastIntention := 'Bearish'

bool fvg123SetupDetected = false

bool keyLevelResponseDetected = false

bool buyFvg123Only = false

bool sellFvg123Only = 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
    afterCreation = bar_index > fvgObj.createdBar
    if afterCreation 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 afterCreation and (fvgObj.isBull ? close < fvgObj.bottom : close >
fvgObj.top)
            fvgObj.done := true
            hasTouch = fvgObj.touched and not na(fvgObj.lastTouchBar) and
bar_index > fvgObj.lastTouchBar and afterCreation
            breakout = (fvgObj.isBull and hasTouch and close > fvgObj.top) or (not
fvgObj.isBull and hasTouch and close < fvgObj.bottom)
            if breakout
                flowSetup = fvgObj.isBull ? lastIntention == 'Bullish' : lastIntention
== 'Bearish'
                sweepSetup = fvgObj.isBull ? lastIntention == 'Bearish' and
recentLowTouch : lastIntention == 'Bullish' and recentHighTouch
                acceptanceSetup = fvgObj.isBull ? not na(lastBullBreachBar) and
bar_index - lastBullBreachBar <= 10 : not na(lastBearBreachBar) and
bar_index - lastBearBreachBar <= 10
                flowEngaged = fvgObj.isBull ? flowSetup and recentHighTouch :
flowSetup and recentLowTouch

```

sweepEngaged = sweepSetup

fvgValid = bar_index - fvgObj.createdBar <= fvgMaxAgeInput and
math.abs(fvgObj.top - fvgObj.bottom) / syminfo.mintick >= fvgMinWidthInput

fvgBeyondDisplacedLevel = fvgObj.isBull ? not
na(lastBullBreachLevel) and not na(lastBullBreachBar) and lastBullBreachBar
<= fvgObj.createdBar and fvgObj.bottom > lastBullBreachLevel : not
na(lastBearBreachLevel) and not na(lastBearBreachBar) and
lastBearBreachBar <= fvgObj.createdBar and fvgObj.top <
lastBearBreachLevel

continuationSetup = (acceptanceSetup or flowEngaged) and
fvgBeyondDisplacedLevel

keyTouchBeforeFvg = not na(fvgObj.keyTouchBar) and
fvgObj.keyTouchBar < fvgObj.createdBar and bar_index - fvgObj.keyTouchBar
<= levelTouchLookbackInput

keyTouchGate = onlyIbResponsesInput ? recentSessionTouch :
recentKeyTouch

keyResponse = continuationSetup and (onlyFvgAfterKeyTouchInput
? keyTouchBeforeFvg and keyTouchGate : true)

setupAllowed = fvgValid and barstate.isconfirmed and keyResponse

if setupAllowed

 fvg123SetupDetected := true

 if fvgObj.isBull

 buyFvg123Only := true

 else

 sellFvg123Only := true

 if keyResponse

```

        keyLevelResponseDetected := true

        entryPrice = close

        stopPrice = fvgObj.isBull ? fvgObj.bottom - syminfo.mintick :
fvgObj.top + syminfo.mintick

        risk = math.abs(entryPrice - stopPrice)

        targetPrice = fvgObj.isBull ? entryPrice + risk * 2 : entryPrice - risk *
2

        setupName = flowEngaged ? 'Flow' : acceptanceSetup ?
'Acceptance' : 'FVG-123'

        labelY = fvgObj.isBull ? low - atrValue * 2.8 : high + atrValue * 2.8

        if showLabelsInput

            label.new(bar_index, labelY, setupName + (fvgObj.isBull ? ' ▲' : '
▼'), color = #00000000, textcolor = #5b9cf6, style = fvgObj.isBull ?
label.style_label_up : label.style_label_down, size = size.normal)

        if showTradeLinesInput

            line.new(bar_index, entryPrice, bar_index +
tradeProjectionInput, entryPrice, color = #5b9cf6, style = line.style_dotted,
width = entryLineWidthInput)

            line.new(bar_index, stopPrice, bar_index + tradeProjectionInput,
stopPrice, color = #f23645, style = line.style_dotted, width =
entryLineWidthInput)

            line.new(bar_index, targetPrice, bar_index +
tradeProjectionInput, targetPrice, color = #089981, style = line.style_dotted,
width = entryLineWidthInput)

        table.cell(setupTable, 0, 0, 'Most Recent: ' + setupName, text_color
= #5b9cf6, text_size = size.normal)

```

```
        alert('Universal MFB - ' + setupName + ' key-level trade  
confirmation on ' + syminfo.ticker, alert.freq_once_per_bar_close)
```

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

```
if not (timeframe.isintraday and timeframe.isminutes and  
(timeframe.multiplier == 1 or timeframe.multiplier == 3 or  
timeframe.multiplier == 5 or timeframe.multiplier == 15 or  
timeframe.multiplier == 30 or timeframe.multiplier == 60)) and barstate.islast  
    runtime.error('Designed for 1m, 3m, 5m, 15m, 30m, and 1h only.')
```

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

```
alertcondition(fvg123SetupDetected, 'Universal MFB I.B. entry', 'Universal  
MFB I.B. trade entry confirmation on {{ticker}}')
```

```
alertcondition(bullChd, 'Bullish Chd', 'Bullish key-level Chd detected on  
{{ticker}}')
```

```
alertcondition(bearChd, 'Bearish Chd', 'Bearish key-level Chd detected on  
{{ticker}}')
```

```
alertcondition(fvg123SetupDetected, 'FVG-123 Key-Level Setup', 'Key-level  
FVG-123 setup confirmed on {{ticker}}')
```

```
alertcondition(buyFvg123Only, 'Buy FVG-123 only', 'Buy key-level FVG-123  
confirmation on {{ticker}}')
```

```
alertcondition(sellFvg123Only, 'Sell FVG-123 only', 'Sell key-level FVG-123  
confirmation on {{ticker}}')
```

```
alertcondition(bull5mCisd, 'Bullish Session CISD', 'Bullish session-range CISD  
detected on {{ticker}}')
```

```
alertcondition(bear5mCisd, 'Bearish Session CISD', 'Bearish session-range  
CISD detected on {{ticker}}')
```

```
alertcondition(keyLevelResponseDetected, 'Any I.B. Key-Level Response',  
'Initial Balance key-level response confirmed on {{ticker}}')
```

```
alertcondition(bull5mCisd or bear5mCisd or keyLevelResponseDetected, 'I.B.  
CISD / Key-Level FVG', 'Initial Balance CISD or key-level FVG-123 confirmed on  
{{ticker}}')
```