

// 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 - Levels Coherence Asia Trading', overlay = true,
max_boxes_count = 500, max_lines_count = 500, max_labels_count = 500)

// --- Inputs ---

groupSessions = 'Sessions (EST/EDT)'

enNy = input.bool(false, 'NY Open', group = groupSessions, inline = 'ny')

nySessTime = input.session('0930-1100', '', group = groupSessions, inline = 'ny')

enLon = input.bool(false, 'London Open', group = groupSessions, inline = 'lon')

lonSessTime = input.session('0300-0430', '', group = groupSessions, inline = 'lon')

enTok = input.bool(true, 'Tokyo Open', group = groupSessions, inline = 'tok')

tokSessTime = input.session('1900-0300', '', group = groupSessions, inline = 'tok')

enNya = input.bool(false, 'NY Afternoon', group = groupSessions, inline = 'nya')

nyaSessTime = input.session('1500-1630', '', group = groupSessions, inline = 'nya')

```
enComex = input.bool(false, 'Comex (Gold)', group = groupSessions, inline = 'comex')
```

```
comexSessTime = input.session('0820-0850', '', group = groupSessions, inline = 'comex')
```

```
asiaSessTime = input.session('1800-0000', 'Asia Session', group = groupSessions)
```

```
lonFullTime = input.session('0000-0600', 'London Full Session', group = groupSessions)
```

```
groupLogic = 'Strategy Logic'
```

```
pointsDist = input.float(12.0, 'Confluence Distance (Points)', group = groupLogic)
```

```
rrRatio = input.float(2.0, 'Risk/Reward Ratio', step = 0.5, group = groupLogic)
```

```
groupSignals = 'Signal Toggles'
```

```
tradingMode = input.string('Custom', 'Trading Mode Preset', options = ['Custom', '1 min 9 EMA with HTF Support', '5 min Signals Only'], group = groupSignals, tooltip = 'Select a preset to automatically configure signals, or select Custom to use the checkboxes below.')
```

```
onlyEma = input.bool(false, 'Show ONLY 9 EMA Retrace Signals', group = groupSignals, tooltip = 'Overrides all other options to only show 9 EMA Retrace signals.')
```

```
onlyCisd = input.bool(false, 'Show ONLY CISC Signals', group = groupSignals, tooltip = 'Overrides all other options to only show 5m CISC signals.')
```

```
showSweep = input.bool(false, 'Enable Sweep Signals (Custom)', group = groupSignals)
```

showPathway = input.bool(false, 'Enable Pathway Signals (Custom)', group = groupSignals)

showCisd = input.bool(true, 'Enable 5m Cisd Signals (Custom)', group = groupSignals)

showHtfShapes = input.bool(false, 'Show 5m Signals on Chart', group = groupSignals, tooltip = 'Plots the 5m timeframe signals visually on the current timeframe chart.')

groupEma = '9 EMA Retrace'

showEma = input.bool(true, 'Enable 9 EMA Retrace Signals (Custom)', group = groupEma)

emaFastLen = input.int(9, 'Fast EMA', group = groupEma)

emaSlowLen = input.int(20, 'Slow EMA', group = groupEma)

emaLineLength = input.int(25, 'EMA Trade Line Length', minval = 1, group = groupEma)

requireCross = input.bool(true, 'Require EMA Cross Before Retrace', group = groupEma)

filterEmaHtf = input.bool(true, 'Filter EMA Signals by HTF Box (Custom)', group = groupEma)

groupDisplay = 'Display Settings'

showHtfBox = input.bool(true, 'Show 5m HTF Reading Box', group = groupDisplay)

showRecentLines = input.bool(false, 'Only Show Most Recent Trade Lines', group = groupDisplay, tooltip = 'When enabled, previous entry/stop/target lines are removed when a new signal occurs.')

```
showInfoTable = input.bool(false, 'Show Discernment Table', group =
groupDisplay)

showAoiScore = input.bool(false, 'Show AOI Score Label', group =
groupDisplay)

show15mOr = input.bool(false, 'Show 15m OR Lines', group = groupDisplay)
or15Width = input.int(1, '15m OR Line Width', minval = 1, maxval = 10, group =
groupDisplay)

show30mOr = input.bool(true, 'Show 30m OR Lines', group = groupDisplay)
or30Width = input.int(1, '30m OR Line Width', minval = 1, maxval = 10, group =
groupDisplay)

show1hOr = input.bool(true, 'Show 1h OR Lines', group = groupDisplay)
or1hWidth = input.int(1, '1h OR Line Width', minval = 1, maxval = 10, group =
groupDisplay)

showAsiaKeyLevels = input.bool(true, 'Show NY and 6pm-8pm Levels', group =
groupDisplay, tooltip = 'Displays the previous NY session high/low and 6pm-
8pm high/low during the 7pm-3am Asia trading window.')

asiaKeyLevelWidth = input.int(2, 'Asia Key Level Width', minval = 1, maxval = 5,
group = groupDisplay, tooltip = 'Width of the NY and 6pm-8pm high/low
levels.')

nyKeyLevelColor = input.color(#5b9cf6, 'Previous NY High/Low Color', group =
groupDisplay, tooltip = 'Color used for the completed 9:30am-4pm NY session
high and low.')

eveningKeyLevelColor = input.color(#f23645, '6pm-8pm High/Low Color',
group = groupDisplay, tooltip = 'Color used for the completed 6pm-8pm range
high and low.')

// --- Mode Logic ---
```

is1minMode = tradingMode == '1 min 9 EMA with HTF Support'

is5minMode = tradingMode == '5 min Signals Only'

isCustomMode = tradingMode == 'Custom'

bool activeSweep = isCustomMode ? showSweep : is5minMode

bool activePathway = isCustomMode ? showPathway : is5minMode

bool activeCisd = isCustomMode ? showCisd : is5minMode

bool activeEma = isCustomMode ? showEma : (is1minMode or is5minMode)

bool activeEmaFilter = isCustomMode ? filterEmaHtf : is1minMode

if onlyEma

 activeSweep := false

 activePathway := false

 activeCisd := false

 activeEma := true

if onlyCisd

 activeSweep := false

 activePathway := false

 activeEma := false

 activeCisd := true

// --- Timezones & Sessions ---

```
tz = 'America/New_York'
```

```
inSession(string sessionInput) =>
```

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

```
isNewSession(string sessionInput) =>
```

```
  inSession(sessionInput) and not inSession(sessionInput)[1]
```

```
nyStart = enNy and isNewSession(nySessTime)
```

```
lonStart = enLon and isNewSession(lonSessTime)
```

```
tokStart = enTok and isNewSession(tokSessTime)
```

```
nyaStart = enNya and isNewSession(nyaSessTime)
```

```
comexStart = enComex and isNewSession(comexSessTime)
```

```
anyStart = nyStart or lonStart or tokStart or nyaStart or comexStart
```

```
// --- Session State & OR ---
```

```
var int sessStartTime = na
```

```
var float or15H = na
```

```
var float or15L = na
```

```
var float or30H = na
```

```
var float or30L = na
```

```
var float or1hH = na
```

```
var float or1hL = na
```

if anyStart

 sessStartTime := time

 or15H := high

 or15L := low

 or30H := high

 or30L := low

 or1hH := high

 or1hL := low

// Update ORs based on elapsed time from session start.

timeElapsedMins = (time - sessStartTime) / 60000

if timeElapsedMins < 15

 or15H := math.max(or15H, high)

 or15L := math.min(or15L, low)

if timeElapsedMins < 30

 or30H := math.max(or30H, high)

 or30L := math.min(or30L, low)

if timeElapsedMins < 60

 or1hH := math.max(or1hH, high)

 or1hL := math.min(or1hL, low)

```
// Extend NY OR until 12:00 PM ET.
```

```
inNyExtension = not na(time(timeframe.period, '0930-1200', tz))
```

```
inComexExtension = not na(time(timeframe.period, '0820-1200', tz))
```

```
var bool nyLinesActive = false
```

```
var bool lonLinesActive = false
```

```
var bool tokLinesActive = false
```

```
var bool nyaLinesActive = false
```

```
var bool comexLinesActive = false
```

```
if nyStart
```

```
    nyLinesActive := true
```

```
    lonLinesActive := false
```

```
    tokLinesActive := false
```

```
    nyaLinesActive := false
```

```
    comexLinesActive := false
```

```
else if lonStart
```

```
    lonLinesActive := true
```

```
    nyLinesActive := false
```

```
    tokLinesActive := false
```

```
    nyaLinesActive := false
```

```
    comexLinesActive := false
```

```
else if tokStart
```

```
tokLinesActive := true
nyLinesActive := false
lonLinesActive := false
nyaLinesActive := false
comexLinesActive := false
else if nyaStart
    nyaLinesActive := true
    nyLinesActive := false
    lonLinesActive := false
    tokLinesActive := false
    comexLinesActive := false
else if comexStart
    comexLinesActive := true
    nyLinesActive := false
    lonLinesActive := false
    tokLinesActive := false
    nyaLinesActive := false

if nyLinesActive and not inNyExtension
    nyLinesActive := false
if lonLinesActive and not inSession(lonSessTime)
    lonLinesActive := false
if tokLinesActive and not inSession(tokSessTime)
```

```
tokLinesActive := false
if nyaLinesActive and not inSession(nyaSessTime)
  nyaLinesActive := false
if comexLinesActive and not inComexExtension
  comexLinesActive := false

canExtendLines = nyLinesActive or lonLinesActive or tokLinesActive or
nyaLinesActive or comexLinesActive
signalsAllowed = canExtendLines

// --- Visualizing the ORs ---
var line or15HLine = na
var line or15LLLine = na
var line or30HLine = na
var line or30LLLine = na
var line or1hHLine = na
var line or1hLLLine = na

if anyStart
  or15HLine := na
  or15LLLine := na
  or30HLine := na
  or30LLLine := na
```

or1hHLine := na

or1hLLine := na

if show15mOr

if timeElapsedMins >= 15 and not na(or15H) and na(or15HLine)

or15HLine := line.new(bar_index - int(timeElapsedMins), or15H,
bar_index, or15H, color = color.red, style = line.style_solid, width = or15Width)

or15LLine := line.new(bar_index - int(timeElapsedMins), or15L, bar_index,
or15L, color = color.red, style = line.style_solid, width = or15Width)

else if timeElapsedMins >= 15 and canExtendLines and not na(or15HLine)

line.set_x2(or15HLine, bar_index)

line.set_x2(or15LLine, bar_index)

if show30mOr

if timeElapsedMins >= 30 and not na(or30H) and na(or30HLine)

or30HLine := line.new(bar_index - int(timeElapsedMins), or30H,
bar_index, or30H, color = color.red, style = line.style_solid, width = or30Width)

or30LLine := line.new(bar_index - int(timeElapsedMins), or30L, bar_index,
or30L, color = color.red, style = line.style_solid, width = or30Width)

else if timeElapsedMins >= 30 and canExtendLines and not na(or30HLine)

line.set_x2(or30HLine, bar_index)

line.set_x2(or30LLine, bar_index)

if show1hOr

```

if timeElapsedMins >= 60 and not na(or1hH) and na(or1hHLine)

    or1hHLine := line.new(bar_index - int(timeElapsedMins), or1hH,
bar_index, or1hH, color = color.red, style = line.style_solid, width = or1hWidth)

    or1hLLine := line.new(bar_index - int(timeElapsedMins), or1hL, bar_index,
or1hL, color = color.red, style = line.style_solid, width = or1hWidth)

else if timeElapsedMins >= 60 and canExtendLines and not na(or1hHLine)

    line.set_x2(or1hHLine, bar_index)

    line.set_x2(or1hLLine, bar_index)

```

```

// --- Key Levels (PDH, PDL, PWH, PWL, PMH, PML, Asia, London) ---

```

```

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

```

```

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

```

```

[pmh, pml] = request.security(syminfo.tickerid, 'M', [high[1], low[1]],
lookahead = barmerge.lookahead_on)

```

```

var float asiaH = na

```

```

var float asiaL = na

```

```

var float lonH = na

```

```

var float lonL = na

```

```

if isNewSession(asiaSessTime)

```

```

    asiaH := high

```

```

    asiaL := low

```

```
else if inSession(asiaSessTime)
    asiaH := math.max(asiaH, high)
    asiaL := math.min(asiaL, low)
```

```
if isNewSession(lonFullTime)
    lonH := high
    lonL := low
else if inSession(lonFullTime)
    lonH := math.max(lonH, high)
    lonL := math.min(lonL, low)
```

```
// --- NY and 6pm-8pm Asia key levels ---
```

```
getCompletedNyLevels() =>
    var float sessionHigh = na
    var float sessionLow = na
    var float completedHigh = na
    var float completedLow = na
    bool inNySession = not na(time(timeframe.period, '0930-1600:23456', tz))
    bool newNySession = inNySession and not inNySession[1]
    bool nySessionEnded = not inNySession and inNySession[1]
    if newNySession
        sessionHigh := high
        sessionLow := low
```

else if inNySession

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

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

if nySessionEnded

completedHigh := sessionHigh

completedLow := sessionLow

[completedHigh, completedLow]

getEveningLevels() =>

var float rangeHigh = na

var float rangeLow = na

bool inRange = not na(time(timeframe.period, '1800-2000:23456', tz))

bool newRange = inRange and not inRange[1]

if newRange

rangeHigh := high

rangeLow := low

else if inRange

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

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

[rangeHigh, rangeLow]

[previousNyHigh, previousNyLow] = getCompletedNyLevels()

[eveningHigh, eveningLow] = getEveningLevels()

```
nyLevelWindow = inSession('1900-0300')
```

```
eveningLevelWindow = inSession('2000-0300')
```

```
plot(showAsiaKeyLevels and nyLevelWindow ? previousNyHigh : na, 'Previous  
NY High (7pm-3am)', color = nyKeyLevelColor, style = plot.style_linebr,  
linewidth = asiaKeyLevelWidth)
```

```
plot(showAsiaKeyLevels and nyLevelWindow ? previousNyLow : na, 'Previous  
NY Low (7pm-3am)', color = nyKeyLevelColor, style = plot.style_linebr,  
linewidth = asiaKeyLevelWidth)
```

```
plot(showAsiaKeyLevels and eveningLevelWindow ? eveningHigh : na, '6pm-  
8pm High (8pm-3am)', color = eveningKeyLevelColor, style = plot.style_linebr,  
linewidth = asiaKeyLevelWidth)
```

```
plot(showAsiaKeyLevels and eveningLevelWindow ? eveningLow : na, '6pm-  
8pm Low (8pm-3am)', color = eveningKeyLevelColor, style = plot.style_linebr,  
linewidth = asiaKeyLevelWidth)
```

```
// Function to calculate confluence.
```

```
getConfluence(float price) =>
```

```
    float nearest = na
```

```
    float minDist = 999999.0
```

```
    int score = 0
```

```
    float[] levels = array.from(pdh, pdl, pwh, pwl, pmh, pml, asiaH, asiaL, lonH,  
lonL, previousNyHigh, previousNyLow, eveningHigh, eveningLow)
```

```
    for level in levels
```

```
        if not na(level)
```

```
            float distance = math.abs(price - level)
```

```

    if distance < minDist
        minDist := distance
        nearest := level
    if distance <= pointsDist
        score += 1
    [nearest, minDist, score]

[nearestToOrH, distH, scoreH] = getConfluence(or15H)
[nearestToOrL, distL, scoreL] = getConfluence(or15L)

isAoiH = distH <= pointsDist
isAoiL = distL <= pointsDist

// --- FVG-123 Logic ---
rawFvgBull = low > high[2] and close[1] > open[1]
rawFvgBear = high < low[2] and close[1] < open[1]

var float activeBullFvgTop = na
var float activeBullFvgBot = na
var bool bullRetrace = false

var float activeBearFvgBot = na
var float activeBearFvgTop = na

```

```
var bool bearRetrace = false
```

```
if rawFvgBull
```

```
    activeBullFvgTop := low
```

```
    activeBullFvgBot := high[2]
```

```
    bullRetrace := false
```

```
if rawFvgBear
```

```
    activeBearFvgBot := high
```

```
    activeBearFvgTop := low[2]
```

```
    bearRetrace := false
```

```
if not na(activeBullFvgTop) and low <= activeBullFvgTop
```

```
    bullRetrace := true
```

```
if not na(activeBearFvgBot) and high >= activeBearFvgBot
```

```
    bearRetrace := true
```

```
bool fvg123Bull = false
```

```
bool fvg123Bear = false
```

```
if bullRetrace and close > activeBullFvgTop
```

```
    fvg123Bull := true
```

```
activeBullFvgTop := na
```

```
if bearRetrace and close < activeBearFvgBot
```

```
    fvg123Bear := true
```

```
    activeBearFvgBot := na
```

```
// --- Sweep/Reversal Logic Implementation ---
```

```
var bool sweptH = false
```

```
var float sweepHVal = na
```

```
var bool sweptL = false
```

```
var float sweepLVal = na
```

```
if isAoiH and high > math.max(or15H, nearestToOrH)
```

```
    sweptH := true
```

```
    sweepHVal := na(sweepHVal) ? high : math.max(sweepHVal, high)
```

```
if isAoiL and low < math.min(or15L, nearestToOrL)
```

```
    sweptL := true
```

```
    sweepLVal := na(sweepLVal) ? low : math.min(sweepLVal, low)
```

```
longSweepSetup = sweptL and close > or15L and fvg123Bull and  
signalsAllowed
```

```
shortSweepSetup = sweptH and close < or15H and fvg123Bear and  
signalsAllowed
```

```
if longSweepSetup
```

```
    sweptL := false
```

```
    sweepLVal := na
```

```
if shortSweepSetup
```

```
    sweptH := false
```

```
    sweepHVal := na
```

```
// --- Pathway Logic (Continuation) ---
```

```
isPathwayH = distH > pointsDist
```

```
isPathwayL = distL > pointsDist
```

```
longPathwaySetup = isPathwayH and close > or15H and fvg123Bull and  
signalsAllowed
```

```
shortPathwaySetup = isPathwayL and close < or15L and fvg123Bear and  
signalsAllowed
```

```
// --- 5m CISD Logic ---
```

```
is5m = timeframe.isintraday and timeframe.multiplier == 5
```

```
var float lastBearFvgTop = na
```

```
var float lastBullFvgBot = na
```

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

```
    lastBearFvgTop := low[2]
```

if low > high[2]

lastBullFvgBot := high[2]

cisdBullCross = ta.crossover(close, lastBearFvgTop)

cisdBearCross = ta.crossunder(close, lastBullFvgBot)

var bool sweptOr301hH = false

var bool sweptOr301hL = false

if tokStart

sweptOr301hH := false

sweptOr301hL := false

bool customHighSweep = nyLevelWindow and ((not na(previousNyHigh) and high >= previousNyHigh) or (eveningLevelWindow and not na(eveningHigh) and high >= eveningHigh))

bool customLowSweep = nyLevelWindow and ((not na(previousNyLow) and low <= previousNyLow) or (eveningLevelWindow and not na(eveningLow) and low <= eveningLow))

if (not na(or30H) and high >= or30H) or (not na(or1hH) and high >= or1hH) or customHighSweep

sweptOr301hH := true

```
if (not na(or30L) and low <= or30L) or (not na(or1hL) and low <= or1hL) or  
customLowSweep
```

```
sweptOr301hL := true
```

```
bool closedInsideH = (not na(or30H) and close < or30H) or (not na(or1hH) and  
close < or1hH) or (not na(previousNyHigh) and close < previousNyHigh) or  
(eveningLevelWindow and not na(eveningHigh) and close < eveningHigh)
```

```
bool closedInsideL = (not na(or30L) and close > or30L) or (not na(or1hL) and  
close > or1hL) or (not na(previousNyLow) and close > previousNyLow) or  
(eveningLevelWindow and not na(eveningLow) and close > eveningLow)
```

```
// For the 5m check, request.security provides the higher-timeframe context.
```

```
bool bullCisdSetup = sweptOr301hL and closedInsideL and cisdBullCross  
and close > lastBearFvgTop and signalsAllowed
```

```
bool bearCisdSetup = sweptOr301hH and closedInsideH and cisdBearCross  
and close < lastBullFvgBot and signalsAllowed
```

```
if bullCisdSetup
```

```
sweptOr301hL := false
```

```
if bearCisdSetup
```

```
sweptOr301hH := false
```

```
// --- HTF 5m Signal Logic & Box (Calculated early for filtering) ---
```

```
[htfLs, htfSs, htfLp, htfSp, htfBcisd, htfBearcisd] =  
request.security(syminfo.tickerid, '5', [longSweepSetup, shortSweepSetup,  
longPathwaySetup, shortPathwaySetup, bullCisdSetup, bearCisdSetup])
```

```
var string lastHtfSignal = 'Awaiting 5m Signal'
```

```
var color lastHtfColor = color.gray
```

```
var int lastHtfDir = 0
```

```
if htfBcisd
```

```
    lastHtfSignal := '5m Bullish CISC'
```

```
    lastHtfColor := color.green
```

```
    lastHtfDir := 1
```

```
else if htfBearcisc
```

```
    lastHtfSignal := '5m Bearish CISC'
```

```
    lastHtfColor := color.red
```

```
    lastHtfDir := -1
```

```
else if htfLs
```

```
    lastHtfSignal := '5m Bullish Sweep'
```

```
    lastHtfColor := color.green
```

```
    lastHtfDir := 1
```

```
else if htfSs
```

```
    lastHtfSignal := '5m Bearish Sweep'
```

```
    lastHtfColor := color.red
```

```
    lastHtfDir := -1
```

```
else if htfLp
```

```
    lastHtfSignal := '5m Bullish Pathway'
```

```
lastHtfColor := color.green
```

```
lastHtfDir := 1
```

```
else if htfSp
```

```
lastHtfSignal := '5m Bearish Pathway'
```

```
lastHtfColor := color.red
```

```
lastHtfDir := -1
```

```
var table htfTable = table.new(position.top_right, 1, 2, bgcolor =  
color.new(color.black, 70), border_color = color.gray, border_width = 1)
```

```
if barstate.islast and showHtfBox
```

```
table.cell(htfTable, 0, 0, 'HTF Reading', text_color = color.white, text_size =  
size.small, bgcolor = color.new(color.blue, 70))
```

```
table.cell(htfTable, 0, 1, lastHtfSignal, text_color = lastHtfColor, text_size =  
size.normal)
```

```
// --- Visualizing HTF Signals on Chart ---
```

```
plotshape(showHtfShapes and htfLs, '5m Long Sweep', shape.triangleup,  
location.belowbar, color.green, size = size.small, text = '5m SW', textcolor =  
color.green)
```

```
plotshape(showHtfShapes and htfSs, '5m Short Sweep', shape.triangledown,  
location.abovebar, color.red, size = size.small, text = '5m SW', textcolor =  
color.red)
```

```
plotshape(showHtfShapes and htfLp, '5m Long Pathway', shape.arrowup,  
location.belowbar, color.lime, size = size.small, text = '5m PW', textcolor =  
color.lime)
```

```
plotshape(showHtfShapes and htfSp, '5m Short Pathway', shape.arrowdown,
location.abovebar, color.maroon, size = size.small, text = '5m PW', textcolor =
color.maroon)
```

```
plotshape(showHtfShapes and htfBcisd, title = '5m Bullish CISC', text = '5m
CISC', style = shape.diamond, location = location.belowbar, color = #089981,
textcolor = #089981, size = size.small)
```

```
plotshape(showHtfShapes and htfBearcisc, title = '5m Bearish CISC', text =
'5m CISC', style = shape.diamond, location = location.abovebar, color =
#f23645, textcolor = #f23645, size = size.small)
```

```
// --- 9 EMA Retrace Logic ---
```

```
emaFast = ta.ema(close, emaFastLen)
```

```
emaSlow = ta.ema(close, emaSlowLen)
```

```
var int longState = 0
```

```
var int shortState = 0
```

```
if ta.crossover(emaFast, emaSlow)
```

```
    longState := 1
```

```
    shortState := 0
```

```
else if ta.crossunder(emaFast, emaSlow)
```

```
    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

    emaLongSignal = longState == 2 and close > emaFast and activeEma
    emaShortSignal = shortState == 2 and close < emaFast and activeEma

// Apply HTF Box Filter if enabled.
if activeEmaFilter
    if lastHtfDir != 1
        emaLongSignal := false
    if lastHtfDir != -1
        emaShortSignal := false

    if emaLongSignal
        longState := 0
    if emaShortSignal
        shortState := 0

```

```
plot(activeEma ? emaFast : na, 'Fast EMA', color = color.blue)
```

```
plot(activeEma ? emaSlow : na, 'Slow EMA', color = color.red)
```

```
// --- Execution & Visualizing Entries & Lines ---
```

```
bool finalLongSweep = longSweepSetup and activeSweep
```

```
bool finalShortSweep = shortSweepSetup and activeSweep
```

```
bool finalLongPathway = longPathwaySetup and activePathway
```

```
bool finalShortPathway = shortPathwaySetup and activePathway
```

```
bool finalBullCisd = bullCisdSetup and is5m and activeCisd
```

```
bool finalBearCisd = bearCisdSetup and is5m and activeCisd
```

```
bool anyLongSignal = finalLongSweep or finalLongPathway or finalBullCisd
```

```
bool anyShortSignal = finalShortSweep or finalShortPathway or finalBearCisd
```

```
plotshape(finalLongSweep, 'Long Sweep Entry', shape.triangleup,  
location.belowbar, color.green, size = size.small)
```

```
plotshape(finalShortSweep, 'Short Sweep Entry', shape.triangledown,  
location.abovebar, color.red, size = size.small)
```

```
plotshape(finalLongPathway, 'Long Pathway Entry', shape.arrowup,  
location.belowbar, color.lime, size = size.small)
```

```
plotshape(finalShortPathway, 'Short Pathway Entry', shape.arrowdown,  
location.abovebar, color.maroon, size = size.small)
```

```
plotshape(finalBullCisd, title = 'Bullish Cisd', text = 'Cisd', style =  
shape.diamond, location = location.belowbar, color = #089981, textcolor =  
#089981, size = size.small)
```

```
plotshape(finalBearCisd, title = 'Bearish Cisd', text = 'Cisd', style =  
shape.diamond, location = location.abovebar, color = #f23645, textcolor =  
#f23645, size = size.small)
```

```
plotshape(emaLongSignal, 'Long EMA Entry', shape.triangleup,  
location.belowbar, color.new(color.green, 0), size = size.small)
```

```
plotshape(emaShortSignal, 'Short EMA Entry', shape.triangledown,  
location.abovebar, color.new(color.red, 0), size = size.small)
```

```
// State variables to hold recent lines.
```

```
var line lastEpLine = na
```

```
var line lastSlLine = na
```

```
var line lastTpLine = na
```

```
drawLines(float ep, float sl, float tp, int length, line inEp, line inSl, line inTp) =>
```

```
    if showRecentLines
```

```
        line.delete(inEp)
```

```
        line.delete(inSl)
```

```
        line.delete(inTp)
```

```
    line outEp = line.new(bar_index, ep, bar_index + length, ep, color =  
color.gray, style = line.style_dotted, width = 2)
```

```
    line outSl = line.new(bar_index, sl, bar_index + length, sl, color = color.red,  
style = line.style_dotted, width = 2)
```

```
    line outTp = line.new(bar_index, tp, bar_index + length, tp, color =  
color.green, style = line.style_dotted, width = 2)
```

```
    [outEp, outSl, outTp]
```

```
if finalBullCisd
```

```
    float ep = close
```

```
    float sl = low
```

```
    float risk = ep - sl
```

```
    float tp = ep + risk * rrRatio
```

```
    if risk > 0
```

```
        [newEp, newSl, newTp] = drawLines(ep, sl, tp, 10, lastEpLine, lastSlLine,  
lastTpLine)
```

```
        lastEpLine := newEp
```

```
        lastSlLine := newSl
```

```
        lastTpLine := newTp
```

```
if finalBearCisd
```

```
    float ep = close
```

```
    float sl = high
```

```
    float risk = sl - ep
```

```
    float tp = ep - risk * rrRatio
```

```
    if risk > 0
```

```
        [newEp, newSl, newTp] = drawLines(ep, sl, tp, 10, lastEpLine, lastSlLine,  
lastTpLine)
```

lastEpLine := newEp

lastSlLine := newSl

lastTpLine := newTp

if emaLongSignal

float ep = close

float sl = emaSlow

float risk = ep - sl

float tp = ep + risk * rrRatio

if risk > 0

[newEp, newSl, newTp] = drawLines(ep, sl, tp, emaLineLength, lastEpLine,
lastSlLine, lastTpLine)

lastEpLine := newEp

lastSlLine := newSl

lastTpLine := newTp

if emaShortSignal

float ep = close

float sl = emaSlow

float risk = sl - ep

float tp = ep - risk * rrRatio

if risk > 0

[newEp, newSl, newTp] = drawLines(ep, sl, tp, emaLineLength, lastEpLine,
lastSlLine, lastTpLine)

```

lastEpLine := newEp
lastSlLine := newSl
lastTpLine := newTp

// Move AOI labels to the right.

if finalLongSweep and scoreL > 0 and showAoiScore

    label.new(bar_index + 1, low, 'AOI Score: ' + str.tostring(scoreL), color =
#00000000, textcolor = color.green, style = label.style_label_left)

if finalShortSweep and scoreH > 0 and showAoiScore

    label.new(bar_index + 1, high, 'AOI Score: ' + str.tostring(scoreH), color =
#00000000, textcolor = color.red, style = label.style_label_left)

// --- Information Table ---

var table infoTable = table.new(position.bottom_center, 2, 3, bgcolor =
color.new(color.black, 70), border_color = color.gray, border_width = 1)

if barstate.islast and showInfoTable

    table.cell(infoTable, 0, 0, 'Trade Discernment', text_color = color.white,
text_size = size.small, text_halign = text.align_center, bgcolor =
color.new(color.blue, 70))

    table.merge_cells(infoTable, 0, 0, 1, 0)

    table.cell(infoTable, 0, 1, 'Confluence AOI', text_color = color.green, text_size
= size.small)

    table.cell(infoTable, 1, 1, 'OR & Ext Liq ≤ 12 pts (Reversal/Sweep)', text_color
= color.white, text_size = size.small, text_halign = text.align_left)

```

```
table.cell(infoTable, 0, 2, 'Pathway Mode', text_color = color.lime, text_size = size.small)
```

```
table.cell(infoTable, 1, 2, 'OR & Ext Liq > 12 pts (Continuation)', text_color = color.white, text_size = size.small, text_halign = text.align_left)
```

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

```
alertcondition(anyLongSignal or anyShortSignal or emaLongSignal or emaShortSignal, title = 'MFB - Coherence', message = 'MFB Coherence signal triggered on {{ticker}}')
```

```
bool emaHtfAgreement = (emaLongSignal and lastHtfDir == 1) or (emaShortSignal and lastHtfDir == -1)
```

```
alertcondition(emaHtfAgreement, title = 'MFB - 9 EMA & HTF Agreement', message = '1 min - 9 EMA - HTF and LTF in agreement')
```