

```
// 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 - Liquidity Sweep Trading - Higher Time", shorttitle = "MFB -  
Sweep HTF", overlay = true, max_lines_count = 30, max_labels_count = 50)
```

```
// --- Constants ---
```

```
string TOOLTIP_PIVOT = "Bars used on each side to confirm swing highs and  
lows. Lower values detect nearer-term levels."
```

```
string TOOLTIP_BUFFER = "Minimum distance price must trade beyond a level  
before the sweep is recognized."
```

```
string TOOLTIP_WINDOW = "Maximum number of bars allowed between a  
liquidity sweep and its market structure shift."
```

```
string TOOLTIP_WICK = "Minimum rejection wick as a percentage of the full  
candle range."
```

```
string TOOLTIP_CLOSE_POSITION = "Bearish candles must close at or below  
this fraction of the candle range. Bullish candles use the mirrored minimum."
```

```
string TOOLTIP_ATR_FILTER = "When enabled, the key-level sweep depth must  
be between the selected minimum and maximum ATR multiples."
```

```
string SESSION_TIMEZONE = "America/New_York"
```

```
color BULL_COLOR = #089981
```

```
color BEAR_COLOR = #f23645
```

```
color LEVEL_COLOR = #5b9cf6
```

```
// --- Inputs ---
```

```
int pivotLeftInput = input.int(3, "Pivot left bars", minval = 1, group =  
"Detection", tooltip = TOOLTIP_PIVOT)
```

```
int pivotRightInput = input.int(3, "Pivot right bars", minval = 1, group =  
"Detection", tooltip = TOOLTIP_PIVOT)
```

```
float sweepBufferInput = input.float(0.0, "Sweep buffer (ATR)", minval = 0.0,  
step = 0.05, group = "Detection", tooltip = TOOLTIP_BUFFER)
```

```
int confirmationWindowInput = input.int(20, "MSS confirmation window",  
minval = 1, maxval = 100, group = "Confirmation", tooltip =  
TOOLTIP_WINDOW)
```

```
bool requireMssInput = input.bool(true, "Require market structure shift",  
group = "Confirmation", tooltip = "When enabled, standard entry markers  
require price to break the internal swing in the sweep direction after the  
rejection.")
```

```
int atrLengthInput = input.int(14, "ATR length", minval = 1, group =  
"Confirmation", tooltip = "ATR length used to calculate the optional sweep  
buffer and HTF PWR ATR filter.")
```

```
bool useDailyLevelsInput = input.bool(true, "Use PDH / PDL", group = "Key  
Levels", tooltip = "Uses the prior completed TradingView daily high and low as  
key liquidity levels.")
```

```
bool useWeeklyLevelsInput = input.bool(true, "Use PWH / PWL", group = "Key  
Levels", tooltip = "Uses the prior completed TradingView weekly high and low  
as key liquidity levels.")
```

```
bool useAsiaLevelsInput = input.bool(true, "Use Asia high / low", group = "Key  
Levels", tooltip = "Uses the prior completed Asia session, defined as 6:00 PM  
to midnight New York time.")
```

```
bool useLondonLevelsInput = input.bool(true, "Use London high / low", group  
= "Key Levels", tooltip = "Uses the prior completed London session, defined as  
midnight to 6:00 AM New York time.")
```

bool enableKeyLevelAlertsInput = input.bool(true, "Enable key-level entry alerts", group = "Key Levels", tooltip = "Enables alerts for a key-level sweep followed by a close beyond the most recent confirmed pre-sweep swing.")

float rejectionWickRatioInput = input.float(0.35, "Minimum rejection wick ratio", minval = 0.0, maxval = 1.0, step = 0.05, group = "HTF PWR", tooltip = TOOLTIP_WICK)

float bearishClosePositionInput = input.float(0.40, "Bearish close position maximum", minval = 0.0, maxval = 1.0, step = 0.05, group = "HTF PWR", tooltip = TOOLTIP_CLOSE_POSITION)

float bullishClosePositionInput = input.float(0.60, "Bullish close position minimum", minval = 0.0, maxval = 1.0, step = 0.05, group = "HTF PWR", tooltip = TOOLTIP_CLOSE_POSITION)

bool useAtrPowerFilterInput = input.bool(false, "Use ATR - HTF Key Level Power Candle", group = "HTF PWR", tooltip = TOOLTIP_ATR_FILTER)

float minimumNormalizedSweepInput = input.float(0.05, "Minimum sweep depth (ATR)", minval = 0.0, step = 0.05, group = "HTF PWR", tooltip = "Minimum distance the candle must pierce beyond the key level, measured in ATR.")

float maximumNormalizedSweepInput = input.float(0.40, "Maximum sweep depth (ATR)", minval = 0.0, step = 0.05, group = "HTF PWR", tooltip = "Maximum distance the candle may pierce beyond the key level, measured in ATR.")

bool enablePowerReversalAlertsInput = input.bool(true, "Enable HTF PWR alerts", group = "HTF PWR", tooltip = "Enables alerts when a key level is swept and the same confirmed candle closes back inside with the required rejection characteristics.")

bool showPivotPointsInput = input.bool(false, "Show confirmed pivots", group = "Style", tooltip = "Displays the confirmed swing points used as liquidity levels.")

```
bool showSweepLevelsInput = input.bool(true, "Show sweep levels", group =  
"Style", tooltip = "Displays the latest bullish and bearish liquidity levels as  
dashed lines.")
```

```
bool showPowerReversalInput = input.bool(true, "Show HTF PWR labels",  
group = "Style", tooltip = "Displays PWR to the left of the usual Sweep marker  
when a key-level power reversal candle is confirmed.")
```

```
bool hidePowerReversalOnOneMinuteInput = input.bool(false, "Hide HTF  
PWR labels on 1m", group = "Style", tooltip = "Automatically hides PWR labels  
on the 1-minute chart while keeping the PWR alert conditions active.")
```

```
color bullishColorInput = input.color(BULL_COLOR, "Bullish color", group =  
"Style", tooltip = "Color used for bullish sweep and entry signals.")
```

```
color bearishColorInput = input.color(BEAR_COLOR, "Bearish color", group =  
"Style", tooltip = "Color used for bearish sweep and entry signals.")
```

```
color levelColorInput = input.color(LEVEL_COLOR, "Level color", group =  
"Style", tooltip = "Color used for liquidity level lines.")
```

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

```
[previousDayHigh, previousDayLow] = request.security(syminfo.tickerid, "D",  
[high[1], low[1]], gaps = barmerge.gaps_off, lookahead =  
barmerge.lookahead_on)
```

```
[previousWeekHigh, previousWeekLow] = request.security(syminfo.tickerid,  
"W", [high[1], low[1]], gaps = barmerge.gaps_off, lookahead =  
barmerge.lookahead_on)
```

```
// --- Completed Asia and London session levels ---
```

```
int currentNewYorkHour = hour(time, SESSION_TIMEZONE)
```

```
bool inAsiaSession = currentNewYorkHour >= 18
```

```
bool inLondonSession = currentNewYorkHour < 6
```

```
var bool asiaSessionActive = false
```

```
var float asiaSessionHigh = na
```

```
var float asiaSessionLow = na
```

```
var float previousAsiaHigh = na
```

```
var float previousAsiaLow = na
```

```
var bool londonSessionActive = false
```

```
var float londonSessionHigh = na
```

```
var float londonSessionLow = na
```

```
var float previousLondonHigh = na
```

```
var float previousLondonLow = na
```

```
if inAsiaSession
```

```
    if not asiaSessionActive
```

```
        asiaSessionHigh := high
```

```
        asiaSessionLow := low
```

```
        asiaSessionActive := true
```

```
    else
```

```
        asiaSessionHigh := math.max(asiaSessionHigh, high)
```

```
        asiaSessionLow := math.min(asiaSessionLow, low)
```

```
else if asiaSessionActive
```

```
    previousAsiaHigh := asiaSessionHigh
```

```
previousAsiaLow := asiaSessionLow
```

```
asiaSessionActive := false
```

```
if inLondonSession
```

```
    if not londonSessionActive
```

```
        londonSessionHigh := high
```

```
        londonSessionLow := low
```

```
        londonSessionActive := true
```

```
    else
```

```
        londonSessionHigh := math.max(londonSessionHigh, high)
```

```
        londonSessionLow := math.min(londonSessionLow, low)
```

```
else if londonSessionActive
```

```
    previousLondonHigh := londonSessionHigh
```

```
    previousLondonLow := londonSessionLow
```

```
    londonSessionActive := false
```

```
// --- Confirmed pivots and volatility buffer ---
```

```
float pivotHigh = ta.pivohigh(high, pivotLeftInput, pivotRightInput)
```

```
float pivotLow = ta.pivotlow(low, pivotLeftInput, pivotRightInput)
```

```
float atrValue = ta.atr(atrLengthInput)
```

```
float sweepBuffer = atrValue * sweepBufferInput
```

```
var float lastSwingHighPrice = na
```

```
var int lastSwingHighIndex = na
var float lastSwingLowPrice = na
var int lastSwingLowIndex = na
```

```
// --- Standard sweep state ---
```

```
var int lastSweptHighIndex = na
var int lastSweptLowIndex = na
var bool pendingBullish = false
var bool pendingBearish = false
var int bullishSweepIndex = na
var int bearishSweepIndex = na
var float bullishTriggerPrice = na
var float bearishTriggerPrice = na
var line bullishLevelLine = na
var line bearishLevelLine = na
```

```
// --- Standard liquidity sweep detection ---
```

```
bool freshHighLevel = na(lastSweptHighIndex) ? true : lastSweptHighIndex !=
lastSwingHighIndex

bool freshLowLevel = na(lastSweptLowIndex) ? true : lastSweptLowIndex !=
lastSwingLowIndex

bool bullishSweep = not na(lastSwingLowPrice) and freshLowLevel and low <
lastSwingLowPrice - sweepBuffer and close > lastSwingLowPrice
```

```
bool bearishSweep = not na(lastSwingHighPrice) and freshHighLevel and high  
> lastSwingHighPrice + sweepBuffer and close < lastSwingHighPrice
```

```
// A candle that sweeps both sides is ignored to keep the directional signal  
unambiguous.
```

```
bool bullishSweepEvent = bullishSweep and not bearishSweep
```

```
bool bearishSweepEvent = bearishSweep and not bullishSweep
```

```
if bullishSweepEvent
```

```
    lastSweptLowIndex := lastSwingLowIndex
```

```
    pendingBullish := true
```

```
    pendingBearish := false
```

```
    bullishSweepIndex := bar_index
```

```
    bullishTriggerPrice := lastSwingHighPrice
```

```
    if not na(bullishLevelLine)
```

```
        line.delete(bullishLevelLine)
```

```
    if showSweepLevelsInput
```

```
        bullishLevelLine := line.new(lastSwingLowIndex, lastSwingLowPrice,  
bar_index, lastSwingLowPrice, xloc = xloc.bar_index, extend = extend.right,  
color = color.new(levelColorInput, 15), style = line.style_dashed, width = 1)
```

```
if bearishSweepEvent
```

```
    lastSweptHighIndex := lastSwingHighIndex
```

```
    pendingBearish := true
```

```

pendingBullish := false

bearishSweepIndex := bar_index

bearishTriggerPrice := lastSwingLowPrice

if not na(bearishLevelLine)
    line.delete(bearishLevelLine)

if showSweepLevelsInput
    bearishLevelLine := line.new(lastSwingHighIndex, lastSwingHighPrice,
    bar_index, lastSwingHighPrice, xloc = xloc.bar_index, extend = extend.right,
    color = color.new(levelColorInput, 15), style = line.style_dashed, width = 1)

// --- Key-level sweep candidates ---

bool pdhSweep = useDailyLevelsInput and not na(previousDayHigh) and high
> previousDayHigh + sweepBuffer and close < previousDayHigh

bool pdlSweep = useDailyLevelsInput and not na(previousDayLow) and low <
previousDayLow - sweepBuffer and close > previousDayLow

bool pwhSweep = useWeeklyLevelsInput and not na(previousWeekHigh) and
high > previousWeekHigh + sweepBuffer and close < previousWeekHigh

bool pwlSweep = useWeeklyLevelsInput and not na(previousWeekLow) and
low < previousWeekLow - sweepBuffer and close > previousWeekLow

bool asiaHighSweep = useAsiaLevelsInput and not na(previousAsiaHigh) and
high > previousAsiaHigh + sweepBuffer and close < previousAsiaHigh

bool asiaLowSweep = useAsiaLevelsInput and not na(previousAsiaLow) and
low < previousAsiaLow - sweepBuffer and close > previousAsiaLow

bool londonHighSweep = useLondonLevelsInput and not
na(previousLondonHigh) and high > previousLondonHigh + sweepBuffer and
close < previousLondonHigh

```

```
bool londonLowSweep = useLondonLevelsInput and not  
na(previousLondonLow) and low < previousLondonLow - sweepBuffer and  
close > previousLondonLow
```

```
var string lastBullishKeyName = ""
```

```
var float lastBullishKeyPrice = na
```

```
var string lastBearishKeyName = ""
```

```
var float lastBearishKeyPrice = na
```

```
bool pdllsNew = lastBullishKeyName != "PDL" ? true : lastBullishKeyPrice !=  
previousDayLow
```

```
bool pwllsNew = lastBullishKeyName != "PWL" ? true : lastBullishKeyPrice !=  
previousWeekLow
```

```
bool asiaLowlsNew = lastBullishKeyName != "Asia Low" ? true :  
lastBullishKeyPrice != previousAsiaLow
```

```
bool londonLowlsNew = lastBullishKeyName != "London Low" ? true :  
lastBullishKeyPrice != previousLondonLow
```

```
bool pdhlsNew = lastBearishKeyName != "PDH" ? true : lastBearishKeyPrice !=  
previousDayHigh
```

```
bool pwhlsNew = lastBearishKeyName != "PWH" ? true : lastBearishKeyPrice  
!= previousWeekHigh
```

```
bool asiaHighlsNew = lastBearishKeyName != "Asia High" ? true :  
lastBearishKeyPrice != previousAsiaHigh
```

```
bool londonHighlsNew = lastBearishKeyName != "London High" ? true :  
lastBearishKeyPrice != previousLondonHigh
```

```
float bullishKeyCandidatePrice = na
string bullishKeyCandidateName = ""
if pdlSweep and pdllsNew
    bullishKeyCandidatePrice := previousDayLow
    bullishKeyCandidateName := "PDL"
else if pwlSweep and pwllsNew
    bullishKeyCandidatePrice := previousWeekLow
    bullishKeyCandidateName := "PWL"
else if asiaLowSweep and asiaLowsNew
    bullishKeyCandidatePrice := previousAsiaLow
    bullishKeyCandidateName := "Asia Low"
else if londonLowSweep and londonLowsNew
    bullishKeyCandidatePrice := previousLondonLow
    bullishKeyCandidateName := "London Low"
```

```
float bearishKeyCandidatePrice = na
string bearishKeyCandidateName = ""
if pdhSweep and pdhlsNew
    bearishKeyCandidatePrice := previousDayHigh
    bearishKeyCandidateName := "PDH"
else if pwhSweep and pwhlsNew
    bearishKeyCandidatePrice := previousWeekHigh
    bearishKeyCandidateName := "PWH"
```

else if asiaHighSweep and asiaHighIsNew

 bearishKeyCandidatePrice := previousAsiaHigh

 bearishKeyCandidateName := "Asia High"

else if londonHighSweep and londonHighIsNew

 bearishKeyCandidatePrice := previousLondonHigh

 bearishKeyCandidateName := "London High"

// A candle that sweeps key levels on both sides is ignored for the key-level state machine.

bool bullishKeySweepEvent = bullishKeyCandidateName != "" and
bearishKeyCandidateName == ""

bool bearishKeySweepEvent = bearishKeyCandidateName != "" and
bullishKeyCandidateName == ""

// --- HTF Key Level Power Reversal Candle ---

float candleRange = high - low

float upperWick = high - math.max(open, close)

float lowerWick = math.min(open, close) - low

float closePosition = candleRange > 0.0 ? (close - low) / candleRange : na

float bearishNormalizedSweep = not na(bearishKeyCandidatePrice) and not
na(atrValue) and atrValue > 0.0 ? (high - bearishKeyCandidatePrice) / atrValue
: na

float bullishNormalizedSweep = not na(bullishKeyCandidatePrice) and not
na(atrValue) and atrValue > 0.0 ? (bullishKeyCandidatePrice - low) / atrValue :
na

```
bool bearishAtrQuality = not useAtrPowerFilterInput or not  
na(bearishNormalizedSweep) and bearishNormalizedSweep >=  
minimumNormalizedSweepInput and bearishNormalizedSweep <=  
maximumNormalizedSweepInput
```

```
bool bullishAtrQuality = not useAtrPowerFilterInput or not  
na(bullishNormalizedSweep) and bullishNormalizedSweep >=  
minimumNormalizedSweepInput and bullishNormalizedSweep <=  
maximumNormalizedSweepInput
```

```
// The candle must sweep the key level, reclaim it on close, show a dominant  
rejection wick,
```

```
// and finish in the opposite portion of its range. PWR is evaluated only on a  
confirmed close.
```

```
bool bearishPowerReversal = barstate.isconfirmed and  
bearishKeySweepEvent and candleRange > 0.0 and upperWick / candleRange  
>= rejectionWickRatioInput and closePosition <= bearishClosePositionInput  
and bearishAtrQuality
```

```
bool bullishPowerReversal = barstate.isconfirmed and bullishKeySweepEvent  
and candleRange > 0.0 and lowerWick / candleRange >=  
rejectionWickRatioInput and closePosition >= bullishClosePositionInput and  
bullishAtrQuality
```

```
bool bullishPowerReversalAlert = enablePowerReversalAlertsInput and  
bullishPowerReversal
```

```
bool bearishPowerReversalAlert = enablePowerReversalAlertsInput and  
bearishPowerReversal
```

```
bool anyPowerReversalAlert = bullishPowerReversalAlert or  
bearishPowerReversalAlert
```

```

// --- Key-level sweep state ---

var bool pendingBullishKey = false
var bool pendingBearishKey = false
var int bullishKeySweepIndex = na
var int bearishKeySweepIndex = na
var float bullishKeyTriggerPrice = na
var float bearishKeyTriggerPrice = na
var string bullishKeyName = ""
var string bearishKeyName = ""
var line bullishKeyLevelLine = na
var line bearishKeyLevelLine = na

if bullishKeySweepEvent
    lastBullishKeyName := bullishKeyCandidateName
    lastBullishKeyPrice := bullishKeyCandidatePrice
    pendingBullishKey := true
    pendingBearishKey := false
    bullishKeySweepIndex := bar_index
    bullishKeyTriggerPrice := lastSwingHighPrice
    bullishKeyName := bullishKeyCandidateName
    if not na(bullishKeyLevelLine)
        line.delete(bullishKeyLevelLine)
    if showSweepLevelsInput

```

```
    bullishKeyLevelLine := line.new(bar_index, bullishKeyCandidatePrice,  
bar_index, bullishKeyCandidatePrice, xloc = xloc.bar_index, extend =  
extend.right, color = color.new(bullishColorInput, 25), style =  
line.style_dotted, width = 2)
```

```
if bearishKeySweepEvent
```

```
    lastBearishKeyName := bearishKeyCandidateName
```

```
    lastBearishKeyPrice := bearishKeyCandidatePrice
```

```
    pendingBearishKey := true
```

```
    pendingBullishKey := false
```

```
    bearishKeySweepIndex := bar_index
```

```
    bearishKeyTriggerPrice := lastSwingLowPrice
```

```
    bearishKeyName := bearishKeyCandidateName
```

```
    if not na(bearishKeyLevelLine)
```

```
        line.delete(bearishKeyLevelLine)
```

```
    if showSweepLevelsInput
```

```
        bearishKeyLevelLine := line.new(bar_index, bearishKeyCandidatePrice,  
bar_index, bearishKeyCandidatePrice, xloc = xloc.bar_index, extend =  
extend.right, color = color.new(bearishColorInput, 25), style =  
line.style_dotted, width = 2)
```

```
// --- Update liquidity levels after processing the current bar ---
```

```
if not na(pivotHigh)
```

```
    lastSwingHighPrice := pivotHigh
```

```
    lastSwingHighIndex := bar_index - pivotRightInput
```

```

if not na(pivotLow)

    lastSwingLowPrice := pivotLow

    lastSwingLowIndex := bar_index - pivotRightInput


// --- Market structure shift confirmation ---

bool bullishMssCross = ta.crossover(close, bullishTriggerPrice)
bool bearishMssCross = ta.crossunder(close, bearishTriggerPrice)
bool bullishKeyMssCross = ta.crossover(close, bullishKeyTriggerPrice)
bool bearishKeyMssCross = ta.crossunder(close, bearishKeyTriggerPrice)
bool bullishWithinWindow = not na(bullishSweepIndex) and bar_index -
bullishSweepIndex <= confirmationWindowInput
bool bearishWithinWindow = not na(bearishSweepIndex) and bar_index -
bearishSweepIndex <= confirmationWindowInput
bool bullishKeyWithinWindow = not na(bullishKeySweepIndex) and bar_index
- bullishKeySweepIndex <= confirmationWindowInput
bool bearishKeyWithinWindow = not na(bearishKeySweepIndex) and
bar_index - bearishKeySweepIndex <= confirmationWindowInput


bool bullishEntrySignal = requireMssInput ? pendingBullish and
bullishWithinWindow and bar_index > bullishSweepIndex and
bullishMssCross : bullishSweepEvent

bool bearishEntrySignal = requireMssInput ? pendingBearish and
bearishWithinWindow and bar_index > bearishSweepIndex and
bearishMssCross : bearishSweepEvent

```

bool bullishKeyEntrySignal = pendingBullishKey and bullishKeyWithinWindow
and bar_index > bullishKeySweepIndex and bullishKeyMssCross

bool bearishKeyEntrySignal = pendingBearishKey and
bearishKeyWithinWindow and bar_index > bearishKeySweepIndex and
bearishKeyMssCross

bool bullishKeyEntryAlert = enableKeyLevelAlertsInput and
bullishKeyEntrySignal

bool bearishKeyEntryAlert = enableKeyLevelAlertsInput and
bearishKeyEntrySignal

bool anyKeyLevelEntryAlert = bullishKeyEntryAlert or bearishKeyEntryAlert

if bullishEntrySignal

 pendingBullish := false

if bearishEntrySignal

 pendingBearish := false

if bullishKeyEntrySignal

 pendingBullishKey := false

if bearishKeyEntrySignal

 pendingBearishKey := false

if pendingBullish and not bullishWithinWindow

 pendingBullish := false

if pendingBearish and not bearishWithinWindow

pendingBearish := false

if pendingBullishKey and not bullishKeyWithinWindow

pendingBullishKey := false

if pendingBearishKey and not bearishKeyWithinWindow

pendingBearishKey := false

// --- Visual elements ---

bool showPowerReversalLabel = showPowerReversalInput and (not
hidePowerReversalOnOneMinuteInput or timeframe.period != "1")

plotshape(showPivotPointsInput and not na(pivotHigh), title = "Confirmed
swing high", style = shape.circle, location = location.abovebar, offset = -
pivotRightInput, color = bearishColorInput, size = size.tiny)

plotshape(showPivotPointsInput and not na(pivotLow), title = "Confirmed
swing low", style = shape.circle, location = location.belowbar, offset = -
pivotRightInput, color = bullishColorInput, size = size.tiny)

plotshape(bullishSweepEvent, title = "Bullish liquidity sweep", style =
shape.circle, location = location.belowbar, color = bullishColorInput, text =
"Sweep", textcolor = chart.fg_color, size = size.tiny)

plotshape(bearishSweepEvent, title = "Bearish liquidity sweep", style =
shape.circle, location = location.abovebar, color = bearishColorInput, text =
"Sweep", textcolor = chart.fg_color, size = size.tiny)

```
plotshape(bullishEntrySignal, title = "Bullish sweep entry confirmation", style
= shape.triangleup, location = location.belowbar, color = bullishColorInput,
text = "LONG", textcolor = chart.fg_color, size = size.small)
```

```
plotshape(bearishEntrySignal, title = "Bearish sweep entry confirmation", style
= shape.triangledown, location = location.abovebar, color =
bearishColorInput, text = "SHORT", textcolor = chart.fg_color, size =
size.small)
```

```
// label.style_label_right anchors the label at the candle and places its text to
the left.
```

```
// Both labels are positioned above the candle with a directional background
and key-level name.
```

```
float powerLabelOffset = not na(atrValue) ? math.max(candleRange * 0.25,
atrValue * 0.15) : math.max(candleRange * 0.25, syminfo.mintick * 10.0)
```

```
float powerLabelPrice = high + powerLabelOffset
```

```
if showPowerReversalLabel and bullishPowerReversal
```

```
    label.new(bar_index, powerLabelPrice, "PWR " +
bullishKeyCandidateName, xloc = xloc.bar_index, yloc = yloc.price, color =
color.new(bullishColorInput, 15), style = label.style_label_right, textcolor =
chart.fg_color, size = size.small)
```

```
if showPowerReversalLabel and bearishPowerReversal
```

```
    label.new(bar_index, powerLabelPrice, "PWR " +
bearishKeyCandidateName, xloc = xloc.bar_index, yloc = yloc.price, color =
color.new(bearishColorInput, 15), style = label.style_label_right, textcolor =
chart.fg_color, size = size.small)
```

```
// --- Dynamic key-level alerts ---
```

```
if bullishKeyEntryAlert
```

```
    alert("Bullish key-level sweep + MSS confirmed at " + bullishKeyName + " on  
" + syminfo.ticker + " " + timeframe.period + ".", alert.freq_once_per_bar_close)
```

```
if bearishKeyEntryAlert
```

```
    alert("Bearish key-level sweep + MSS confirmed at " + bearishKeyName + "  
on " + syminfo.ticker + " " + timeframe.period + ".",  
alert.freq_once_per_bar_close)
```

```
if bullishPowerReversalAlert
```

```
    alert("Bullish HTF PWR at " + bullishKeyCandidateName + " on " +  
syminfo.ticker + " " + timeframe.period + ".", alert.freq_once_per_bar_close)
```

```
if bearishPowerReversalAlert
```

```
    alert("Bearish HTF PWR at " + bearishKeyCandidateName + " on " +  
syminfo.ticker + " " + timeframe.period + ".", alert.freq_once_per_bar_close)
```

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

```
alertcondition(bullishSweepEvent, title = "Bullish liquidity sweep", message =  
"Bullish liquidity sweep detected on {{ticker}} {{interval}}.")
```

```
alertcondition(bearishSweepEvent, title = "Bearish liquidity sweep", message  
= "Bearish liquidity sweep detected on {{ticker}} {{interval}}.")
```

```
alertcondition(bullishEntrySignal, title = "Bullish sweep entry confirmation",  
message = "Bullish liquidity sweep confirmed by market structure shift on  
{{ticker}} {{interval}}.")
```

```
alertcondition(bearishEntrySignal, title = "Bearish sweep entry confirmation",  
message = "Bearish liquidity sweep confirmed by market structure shift on  
{{ticker}} {{interval}}.")
```

```
alertcondition(bullishKeyEntryAlert, title = "Bullish key-level sweep entry  
confirmation", message = "Bullish key-level sweep and close above the most  
recent confirmed pre-sweep swing high on {{ticker}} {{interval}}.")
```

```
alertcondition(bearishKeyEntryAlert, title = "Bearish key-level sweep entry  
confirmation", message = "Bearish key-level sweep and close below the most  
recent confirmed pre-sweep swing low on {{ticker}} {{interval}}.")
```

```
alertcondition(anyKeyLevelEntryAlert, title = "Any key-level sweep entry  
confirmation", message = "Bullish or bearish key-level sweep and pre-sweep  
structure confirmation detected on {{ticker}} {{interval}}.")
```

```
alertcondition(bullishPowerReversalAlert, title = "Bullish Sweep PWR Candle",  
message = "Bullish Sweep PWR Candle detected on {{ticker}} {{interval}}.")
```

```
alertcondition(bearishPowerReversalAlert, title = "Bearish Sweep PWR  
Candle", message = "Bearish Sweep PWR Candle detected on {{ticker}}  
{{interval}}.")
```

```
alertcondition(anyPowerReversalAlert, title = "Any Sweep PWR Candle",  
message = "Bullish or bearish Sweep PWR Candle detected on {{ticker}}  
{{interval}}.")
```