```
// 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 - One chart - 1 min", shorttitle = "MFB - One chart - 1 min", overlay = true, max_lines_count = 200, max_labels_count = 200)


// --- 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 useOpeningRangeLevelsInput = input.bool(true, "Use 9:00–10:00 ET high / low", group = "Key Levels", tooltip = "Adds the current New York 9:00–10:00 AM range high and low as key levels after the window closes. Automatically follows Eastern Time DST changes.")

bool showKeyLevelEntrySignalsInput = input.bool(true, "Show key-level entry markers", group = "Key-Level RR", tooltip = "Shows arrows when a key-level sweep is confirmed by a market structure shift.")

bool showKeyLevelRiskRewardInput = input.bool(true, "Show key-level entry/stop/target lines", group = "Key-Level RR", tooltip = "Draws projected entry, sweep-extreme stop, and target guides for every confirmed key-level MSS signal.")

float keyLevelRewardRiskInput = input.float(2.0, "Key-level target risk/reward", minval = 0.1, step = 0.1, group = "Key-Level RR", tooltip = "Target distance as a multiple of entry-to-stop risk. 2.0 means a 1:2 risk/reward target.")

bool showKeyLevelRiskRewardLabelsInput = input.bool(false, "Show key-level Entry/SL/TP tags", group = "Key-Level RR", tooltip = "Adds text labels to key-level entry, stop, and configured target guides.")

int keyLevelProjectionBarsInput = input.int(10, "Key-level guide length (bars)", minval = 1, maxval = 100, group = "Key-Level RR", tooltip = "Number of bars to project key-level entry, stop, and target guides to the right of each signal.")

int keyLevelRiskRewardLineWidthInput = input.int(3, "Key-level trade line width", minval = 1, maxval = 5, group = "Key-Level RR", tooltip = "Thickness of key-level entry, stop, and configured target guides.")

int maxDisplayedKeyLevelSetupsInput = input.int(5, "Maximum displayed key-level setups", minval = 1, maxval = 20, group = "Key-Level RR", tooltip = "Limits how many recent key-level entry/stop/target guide sets remain visible.")

float keyLevelStopBufferAtrInput = input.float(0.10, "Stop buffer beyond sweep (ATR)", minval = 0.0, step = 0.05, group = "Key-Level RR", tooltip = "Adds
```

a buffer beyond the lowest/highest price from the key-level sweep through entry. At least one tick is always used.")

int keyArrowFvgMaxAgeInput = input.int(100, "Maximum FVG age for key arrow (bars)", minval = 1, group = "Key-Level RR", tooltip = "Maximum age of the active, same-direction FVG used to qualify a key-level arrow alert.")

bool showSweepRiskRewardInput = input.bool(true, "Show LONG/SHORT entry/stop/1R lines", group = "Sweep Risk/Reward", tooltip = "Draws entry, sweep-extreme stop, and 1R target guides for regular pivot-sweep signals.")

bool showSweepRiskRewardLabelsInput = input.bool(false, "Show LONG/SHORT Entry/SL/TP tags", group = "Sweep Risk/Reward", tooltip = "Adds text tags to regular pivot-sweep entry, stop, and 1R target guides.")

int sweepRiskRewardProjectionBarsInput = input.int(10, "Sweep guide length (bars)", minval = 1, maxval = 100, group = "Sweep Risk/Reward", tooltip = "Number of bars to project regular pivot-sweep entry, stop, and target guides.")

int sweepRiskRewardLineWidthInput = input.int(3, "Sweep trade line width", minval = 1, maxval = 5, group = "Sweep Risk/Reward", tooltip = "Thickness of regular pivot-sweep entry, stop, and target guides.")

int maxDisplayedSweepSetupsInput = input.int(5, "Maximum displayed sweep setups", minval = 1, maxval = 20, group = "Sweep Risk/Reward", tooltip = "Limits how many recent regular pivot-sweep entry/stop/target guide sets remain visible.")

float sweepStopBufferAtrInput = input.float(0.10, "Stop buffer beyond sweep (ATR)", minval = 0.0, step = 0.05, group = "Sweep Risk/Reward", tooltip = "Adds a buffer beyond the sweep extreme through entry. At least one tick is always used.")

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 enableFvg123Input = input.bool(true, "Enable FVG-123 signals", group = "FVG-123", tooltip = "Detects a three-candle fair value gap, its later retest, and a confirmed close breaking out in the gap direction.")

bool requireFvg123EmaCrossInput = input.bool(true, "Require 9/20 EMA cross", group = "FVG-123", tooltip = "Requires the matching-direction EMA cross after the qualifying key-level touch. One FVG-123 signal is allowed per EMA cross.")

bool requireFvg123KeyTouchInput = input.bool(true, "Require prior key-level touch", group = "FVG-123", tooltip = "Requires price to touch an enabled PDH/PDL, PWH/PWL, or completed Asia/London extreme before the FVG forms.")

bool requireFvg123OpeningRangeBreakInput = input.bool(true, "Require 9:30–9:45 ET OR breakout", group = "FVG-123", tooltip = "Only allows FVG-123 entries after the 9:30–9:45 AM ET 15-minute opening range completes, with bullish closes above its high and bearish closes below its low.")
```

```
int fvg123KeyTouchLookbackInput = input.int(30, "Key-level touch lookback (bars)", minval = 1, group = "FVG-123", tooltip = "Maximum bars between the prior key-level touch and the FVG-123 breakout signal.")

int fvg123MaxAgeInput = input.int(100, "FVG-123 maximum age (bars)", minval = 1, group = "FVG-123", tooltip = "Maximum age of a fair value gap when its breakout signal occurs.")

float fvg123MinWidthTicksInput = input.float(0.0, "FVG-123 minimum width (ticks)", minval = 0.0, group = "FVG-123", tooltip = "Optional minimum gap width in ticks. Zero allows any positive-width FVG.")

int fvgFastEmaLengthInput = input.int(9, "Fast EMA length", minval = 1, group = "FVG-123", tooltip = "Fast EMA used for FVG-123 trend and cross confirmation.")

int fvgSlowEmaLengthInput = input.int(20, "Slow EMA length", minval = 1, group = "FVG-123", tooltip = "Slow EMA used for FVG-123 trend and cross confirmation.")

float fvgRewardRiskInput = input.float(2.0, "Target risk/reward multiple", minval = 0.1, step = 0.1, group = "FVG-123 Risk/Reward", tooltip = "Target distance as a multiple of entry-to-stop risk. Default 2.0 means a 1:2 risk/reward target.")

int fvgProjectionBarsInput = input.int(10, "Risk/reward line length (bars)", minval = 1, maxval = 100, group = "FVG-123 Risk/Reward", tooltip = "Number of bars to project the compact dotted entry, stop, and target guides to the right of each signal.")

int fvgRiskRewardLineWidthInput = input.int(3, "Trade line width", minval = 1, maxval = 5, group = "FVG-123 Risk/Reward", tooltip = "Line thickness for FVG-123 and second-CISD entry, stop, and target guides.")

int maxDisplayedFvgSetupsInput = input.int(5, "Maximum displayed FVG setups", minval = 1, maxval = 20, group = "FVG-123 Risk/Reward", tooltip = "Limits how many recent FVG-123 entry/stop/target guides remain visible.")
```

```
bool enableFiveMinuteCisdInput = input.bool(true, "Enable 5-minute second-CISD signals", group = "5-minute CISD", tooltip = "Signals only on a 5-minute chart when the second CISD event occurs at the same enabled key-level.")

bool enableFifteenMinuteCisdInput = input.bool(true, "Enable 15-minute key-level CISD", group = "15-minute CISD", tooltip = "Uses confirmed 15-minute closes at enabled key levels and requires the 9/20 EMA alignment to agree with signal direction. Plots on a 15-minute or lower chart.")

bool showFifteenMinuteCisdSignalsInput = input.bool(true, "Show 15-minute CISD markers", group = "15-minute CISD", tooltip = "Shows 15m REV and BRK labels below 15 minutes; on the 15-minute chart, shows qualified CISD entry markers instead.")

bool showFifteenMinuteCisdRiskRewardInput = input.bool(true, "Show 1:1 trade guides on 15-minute chart", group = "15-minute CISD", tooltip = "Draws entry, stop, and 1R target lines only on a 15-minute chart for qualified CISD trade signals.")

int fifteenMinuteCisdProjectionBarsInput = input.int(10, "15-minute guide length (bars)", minval = 1, maxval = 100, group = "15-minute CISD", tooltip = "Number of 15-minute bars to project the entry, stop, and 1R target guides.")

int maxDisplayedFifteenMinuteCisdSetupsInput = input.int(5, "Maximum displayed 15-minute CISD setups", minval = 1, maxval = 20, group = "15-minute CISD", tooltip = "Limits how many recent 15-minute CISD entry/stop/target guide sets remain visible.")

string cisd15TextSizeInput = input.string(size.normal, "15-minute CISD text size", options = [size.tiny, size.small, size.normal, size.large, size.huge], group = "15-minute CISD", tooltip = "Controls only the size of the CISD text; Normal is the default and the triangle arrow size is controlled separately.")

string cisd15ArrowSizeInput = input.string(size.tiny, "15-minute CISD arrow size", options = [size.tiny, size.small, size.normal], group = "15-minute CISD", tooltip = "Controls only the size of the CISD triangle arrow, independently from the text. Tiny matches the smaller chart arrows.")
```

```
float cisd15TextOffsetAtrInput = input.float(0.7, "15-minute CISD text spacing (ATR)", minval = 0.1, maxval = 2.0, step = 0.1, group = "15-minute CISD", tooltip = "Separates the large CISD text from its arrow and nearby markers; raise the value for more spacing.")

int buyCisdSweepWindowInput = input.int(20, "Buy CISD to key-level sweep window (1m bars)", minval = 1, maxval = 240, group = "15-minute CISD", tooltip = "Maximum 1-minute bars after a confirmed bullish 15-minute CISD for a new bullish key-level sweep. The MSS arrow may confirm afterward within the existing MSS window.")

int cisdProjectionBarsInput = input.int(10, "CISD guide length (bars)", minval = 1, maxval = 100, group = "5-minute CISD", tooltip = "Number of 5-minute bars to project CISD entry, structural stop, and target guides.")

int maxDisplayedCisdSetupsInput = input.int(5, "Maximum displayed CISD setups", minval = 1, maxval = 20, group = "5-minute CISD", tooltip = "Limits the number of recent CISD entry/stop/target guide sets retained.")

bool showFiveMinuteCisdSignalsInput = input.bool(true, "Show second-CISD signal markers", group = "5-minute CISD", tooltip = "Displays a marker only on the second qualified CISD event at a given key-level price.")

bool showCisdRiskRewardInput = input.bool(true, "Show CISD entry/stop/1R/2R lines", group = "5-minute CISD", tooltip = "Draws the signal entry, structural stop, and both 1R and 2R target guides for the second CISD event.")

bool showCisdRiskRewardLabelsInput = input.bool(false, "Show CISD Entry/SL/TP text tags", group = "5-minute CISD", tooltip = "Adds text labels to the CISD entry, stop, and target lines. Off by default for a cleaner chart.")

bool showFvg123SignalsInput = input.bool(true, "Show FVG-123 signal markers", group = "Style", tooltip = "Displays bullish and bearish FVG-123 breakout markers on the signal bars.")
```

```
bool showFvgRiskRewardInput = input.bool(true, "Show FVG-123 entry/stop/target lines", group = "Style", tooltip = "Draws projected entry, stop, and configured risk/reward target guides for recent FVG-123 signals.")

bool showFvgRiskRewardLabelsInput = input.bool(false, "Show entry/SL/TP text tags", group = "Style", tooltip = "Adds Entry, SL, and target-multiple text tags to the FVG-123 risk/reward lines. Off by default for a cleaner chart.")

bool showSweepMarkersInput = input.bool(false, "Show sweep labels", group = "Style", tooltip = "Displays the bullish and bearish liquidity-sweep markers on price bars.")

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

bool showSweepLevelsInput = input.bool(false, "Show blue swing sweep-reference lines", group = "Style", tooltip = "Displays the blue horizontal lines at confirmed swing highs and lows used by the standard liquidity-sweep detector.")

bool showKeyLevelSweepLinesInput = input.bool(true, "Show swept key-level lines", group = "Style", tooltip = "Displays the latest swept prior-day, prior-week, Asia, London, and 9:00–10:00 ET key levels.")

bool showOpeningRangeLevelsInput = input.bool(true, "Show 9:00–10:00 ET high / low", group = "Style", tooltip = "Draws the completed New York 9:00–10:00 AM range high and low from 10:00 AM through 1:00 PM ET.")

bool showPowerReversalInput = input.bool(false, "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 PWR detection 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.")

color openingRangeColorInput = input.color(BEAR_COLOR, "9:00–10:00 ET line color", group = "Style", tooltip = "Color used for the 9:00–10:00 ET high and low 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

int currentNewYorkDate = year(time, SESSION_TIMEZONE) * 10000 + month(time, SESSION_TIMEZONE) * 100 + dayofmonth(time, SESSION_TIMEZONE)

bool inOpeningRangeSession = currentNewYorkHour == 9
```

```
var int openingRangeDate = na
var float openingRangeHigh = na
var float openingRangeLow = na
var bool openingRangeActive = false
var bool openingRangeComplete = false
var array<line> openingRangeLinesArray = array.new<line>()

if na(openingRangeDate) or currentNewYorkDate != openingRangeDate
   openingRangeDate := currentNewYorkDate
   openingRangeHigh := na
   openingRangeLow := na
   openingRangeActive := false
   openingRangeComplete := false

if inOpeningRangeSession
   if not openingRangeActive
      openingRangeHigh := high
      openingRangeLow := low
      openingRangeActive := true
   else
      openingRangeHigh := math.max(openingRangeHigh, high)
      openingRangeLow := math.min(openingRangeLow, low)
else if openingRangeActive
```

```
    openingRangeActive := false
    openingRangeComplete := true
    if showOpeningRangeLevelsInput and not na(openingRangeHigh) and not
na(openingRangeLow)
        int openingRangeStartTime = timestamp(SESSION_TIMEZONE, year(time,
SESSION_TIMEZONE), month(time, SESSION_TIMEZONE), dayofmonth(time,
SESSION_TIMEZONE), 10, 0)
        int openingRangeEndTime = timestamp(SESSION_TIMEZONE, year(time,
SESSION_TIMEZONE), month(time, SESSION_TIMEZONE), dayofmonth(time,
SESSION_TIMEZONE), 13, 0)
        line openingRangeHighLine = line.new(openingRangeStartTime,
openingRangeHigh, openingRangeEndTime, openingRangeHigh, xloc =
xloc.bar_time, color = color.new(openingRangeColorInput, 5), style =
line.style_solid, width = 2)
        line openingRangeLowLine = line.new(openingRangeStartTime,
openingRangeLow, openingRangeEndTime, openingRangeLow, xloc =
xloc.bar_time, color = color.new(openingRangeColorInput, 5), style =
line.style_solid, width = 2)
        array.push(openingRangeLinesArray, openingRangeHighLine)
        array.push(openingRangeLinesArray, openingRangeLowLine)
        while array.size(openingRangeLinesArray) > 100
            line.delete(array.shift(openingRangeLinesArray))

// --- 9:30–9:45 ET 15-minute opening range for FVG-123 filtering ---
int currentNewYorkMinute = minute(time, SESSION_TIMEZONE)
bool inFvgOpeningRange15Session = currentNewYorkHour == 9 and
currentNewYorkMinute >= 30 and currentNewYorkMinute < 45
```

```
var int fvgOpeningRange15Date = na
var float fvgOpeningRange15High = na
var float fvgOpeningRange15Low = na
var bool fvgOpeningRange15Active = false
var bool fvgOpeningRange15Complete = false

if na(fvgOpeningRange15Date) or currentNewYorkDate !=
fvgOpeningRange15Date
   fvgOpeningRange15Date := currentNewYorkDate
   fvgOpeningRange15High := na
   fvgOpeningRange15Low := na
   fvgOpeningRange15Active := false
   fvgOpeningRange15Complete := false

if inFvgOpeningRange15Session
   if not fvgOpeningRange15Active
      fvgOpeningRange15High := high
      fvgOpeningRange15Low := low
      fvgOpeningRange15Active := true
   else
      fvgOpeningRange15High := math.max(fvgOpeningRange15High, high)
      fvgOpeningRange15Low := math.min(fvgOpeningRange15Low, low)
else if fvgOpeningRange15Active
```

```
    fvgOpeningRange15Active := false
    fvgOpeningRange15Complete := true

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.pivothigh(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 float bullishSweepExtreme = na
var float bearishSweepExtreme = 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

   bullishSweepExtreme := low

   bearishSweepExtreme := na

   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
   bearishSweepExtreme := high
   bullishSweepExtreme := na
   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)


if pendingBullish and not bullishSweepEvent and not
na(bullishSweepExtreme)
   bullishSweepExtreme := math.min(bullishSweepExtreme, low)
if pendingBearish and not bearishSweepEvent and not
na(bearishSweepExtreme)
   bearishSweepExtreme := math.max(bearishSweepExtreme, high)


// --- 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

bool openingRangeHighSweep = useOpeningRangeLevelsInput and openingRangeComplete and not na(openingRangeHigh) and high > openingRangeHigh + sweepBuffer and close < openingRangeHigh

bool openingRangeLowSweep = useOpeningRangeLevelsInput and openingRangeComplete and not na(openingRangeLow) and low < openingRangeLow - sweepBuffer and close > openingRangeLow


var string lastBullishKeyName = ""

var float lastBullishKeyPrice = na

var string lastBearishKeyName = ""

var float lastBearishKeyPrice = na
```

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

bool pwlIsNew = lastBullishKeyName != "PWL" ? true : lastBullishKeyPrice != previousWeekLow

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

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

bool openingRangeLowIsNew = lastBullishKeyName != "9-10 ET Low" ? true : lastBullishKeyPrice != openingRangeLow

bool pdhIsNew = lastBearishKeyName != "PDH" ? true : lastBearishKeyPrice != previousDayHigh

bool pwhIsNew = lastBearishKeyName != "PWH" ? true : lastBearishKeyPrice != previousWeekHigh

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

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

bool openingRangeHighIsNew = lastBearishKeyName != "9-10 ET High" ? true : lastBearishKeyPrice != openingRangeHigh


float bullishKeyCandidatePrice = na

string bullishKeyCandidateName = ""

if pdlSweep and pdlIsNew

   bullishKeyCandidatePrice := previousDayLow
```

```
    bullishKeyCandidateName := "PDL"
else if pwlSweep and pwlIsNew
    bullishKeyCandidatePrice := previousWeekLow
    bullishKeyCandidateName := "PWL"
else if asiaLowSweep and asiaLowIsNew
    bullishKeyCandidatePrice := previousAsiaLow
    bullishKeyCandidateName := "Asia Low"
else if londonLowSweep and londonLowIsNew
    bullishKeyCandidatePrice := previousLondonLow
    bullishKeyCandidateName := "London Low"
else if openingRangeLowSweep and openingRangeLowIsNew
    bullishKeyCandidatePrice := openingRangeLow
    bullishKeyCandidateName := "9-10 ET Low"

float bearishKeyCandidatePrice = na
string bearishKeyCandidateName = ""
if pdhSweep and pdhIsNew
    bearishKeyCandidatePrice := previousDayHigh
    bearishKeyCandidateName := "PDH"
else if pwhSweep and pwhIsNew
    bearishKeyCandidatePrice := previousWeekHigh
    bearishKeyCandidateName := "PWH"
else if asiaHighSweep and asiaHighIsNew
```

```
    bearishKeyCandidatePrice := previousAsiaHigh
    bearishKeyCandidateName := "Asia High"
else if londonHighSweep and londonHighIsNew
    bearishKeyCandidatePrice := previousLondonHigh
    bearishKeyCandidateName := "London High"
else if openingRangeHighSweep and openingRangeHighIsNew
    bearishKeyCandidatePrice := openingRangeHigh
    bearishKeyCandidateName := "9-10 ET 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


// --- 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 float bullishKeySweepExtreme = na
var float bearishKeySweepExtreme = 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
   bullishKeySweepExtreme := low
   bearishKeySweepExtreme := na
   bullishKeyName := bullishKeyCandidateName
   if not na(bullishKeyLevelLine)
      line.delete(bullishKeyLevelLine)
```

```
    if showKeyLevelSweepLinesInput
        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
    bearishKeySweepExtreme := high
    bullishKeySweepExtreme := na
    bearishKeyName := bearishKeyCandidateName
    if not na(bearishKeyLevelLine)
        line.delete(bearishKeyLevelLine)
    if showKeyLevelSweepLinesInput
        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)
```

```
if pendingBullishKey and not bullishKeySweepEvent and not
na(bullishKeySweepExtreme)
   bullishKeySweepExtreme := math.min(bullishKeySweepExtreme, low)
if pendingBearishKey and not bearishKeySweepEvent and not
na(bearishKeySweepExtreme)
   bearishKeySweepExtreme := math.max(bearishKeySweepExtreme, high)


// --- FVG-123 key-touch and EMA context ---
bool fvgPdhTouch = useDailyLevelsInput and not na(previousDayHigh) and
high >= previousDayHigh and low <= previousDayHigh
bool fvgPdlTouch = useDailyLevelsInput and not na(previousDayLow) and high
>= previousDayLow and low <= previousDayLow
bool fvgPwhTouch = useWeeklyLevelsInput and not na(previousWeekHigh)
and high >= previousWeekHigh and low <= previousWeekHigh
bool fvgPwlTouch = useWeeklyLevelsInput and not na(previousWeekLow) and
high >= previousWeekLow and low <= previousWeekLow
bool fvgAsiaHighTouch = useAsiaLevelsInput and not na(previousAsiaHigh)
and high >= previousAsiaHigh and low <= previousAsiaHigh
bool fvgAsiaLowTouch = useAsiaLevelsInput and not na(previousAsiaLow) and
high >= previousAsiaLow and low <= previousAsiaLow
bool fvgLondonHighTouch = useLondonLevelsInput and not
na(previousLondonHigh) and high >= previousLondonHigh and low <=
previousLondonHigh
bool fvgLondonLowTouch = useLondonLevelsInput and not
na(previousLondonLow) and high >= previousLondonLow and low <=
previousLondonLow
```

```
bool fvgOpeningRangeHighTouch = useOpeningRangeLevelsInput and openingRangeComplete and not na(openingRangeHigh) and high >= openingRangeHigh and low <= openingRangeHigh

bool fvgOpeningRangeLowTouch = useOpeningRangeLevelsInput and openingRangeComplete and not na(openingRangeLow) and high >= openingRangeLow and low <= openingRangeLow

bool fvgKeyLevelTouch = fvgPdhTouch or fvgPdlTouch or fvgPwhTouch or fvgPwlTouch or fvgAsiaHighTouch or fvgAsiaLowTouch or fvgLondonHighTouch or fvgLondonLowTouch or fvgOpeningRangeHighTouch or fvgOpeningRangeLowTouch

var int lastFvgKeyLevelTouchBar = na

if fvgKeyLevelTouch

   lastFvgKeyLevelTouchBar := bar_index


float fvgFastEma = ta.ema(close, fvgFastEmaLengthInput)

float fvgSlowEma = ta.ema(close, fvgSlowEmaLengthInput)

bool fvgBullEmaCross = ta.crossover(fvgFastEma, fvgSlowEma)

bool fvgBearEmaCross = ta.crossunder(fvgFastEma, fvgSlowEma)

var int lastFvgBullEmaCrossBar = na

var int lastFvgBearEmaCrossBar = na

if fvgBullEmaCross

   lastFvgBullEmaCrossBar := bar_index

if fvgBearEmaCross

   lastFvgBearEmaCrossBar := bar_index

bool fvgBullEmaContext = not na(lastFvgBullEmaCrossBar) and bar_index >= lastFvgBullEmaCrossBar and fvgFastEma > fvgSlowEma
```

```
bool fvgBearEmaContext = not na(lastFvgBearEmaCrossBar) and bar_index >=
lastFvgBearEmaCrossBar and fvgFastEma < fvgSlowEma


// --- FVG-123 setup tracking ---
type Fvg123Setup
   float top
   float bottom
   bool isBull
   int createdBar
   int keyTouchBar
   bool touched
   int touchBar
   bool done

type KeyArrowFvgZone
   float top
   float bottom
   bool isBull
   int createdBar
   bool active

var array<Fvg123Setup> fvg123SetupsArray = array.new<Fvg123Setup>()
var array<KeyArrowFvgZone> keyArrowFvgZonesArray =
array.new<KeyArrowFvgZone>()
```

```
var array<line> keyLevelRiskRewardLinesArray = array.new<line>()
var array<label> keyLevelRiskRewardLabelsArray = array.new<label>()
var array<line> sweepRiskRewardLinesArray = array.new<line>()
var array<label> sweepRiskRewardLabelsArray = array.new<label>()
var array<line> cisdRiskRewardLinesArray = array.new<line>()
var array<label> cisdRiskRewardLabelsArray = array.new<label>()
var array<line> cisd15RiskRewardLinesArray = array.new<line>()
var bool cisdBearPending = false
var float cisdBearKeyLevel = na
var float cisdBearBreachLow = na
var float cisdBearStructureHigh = na
var int cisdBearStartBar = na
var float lastBearCisdCountLevel = na
var int bearCisdEventCount = 0
var bool cisdBullPending = false
var float cisdBullKeyLevel = na
var float cisdBullBreachHigh = na
var float cisdBullStructureLow = na
var int cisdBullStartBar = na
var float lastBullCisdCountLevel = na
var int bullCisdEventCount = 0
bool cisdBullSignal = false
bool cisdBearSignal = false
```

```
float cisdSignalEntry = na

float cisdSignalStop = na

float cisdSignalTargetOneR = na

float cisdSignalTargetTwoR = na

var int fvg123EmaCrossUsedBar = na

var int buyCisdFollowupCisdBar = na

var array<line> fvgRiskRewardLinesArray = array.new<line>()

var array<label> fvgRiskRewardLabelsArray = array.new<label>()

bool fvgLongSignal = false

bool fvgShortSignal = false

float fvgSignalEntry = na

float fvgSignalStop = na

float fvgSignalTarget = na

bool bullishArrowFvgConfluence = false

bool bearishArrowFvgConfluence = false


if bar_index >= 2

   if low > high[2]

      array.push(keyArrowFvgZonesArray, KeyArrowFvgZone.new(low, high[2],
true, bar_index, true))

   if high < low[2]

      array.push(keyArrowFvgZonesArray, KeyArrowFvgZone.new(low[2], high,
false, bar_index, true))

   if array.size(keyArrowFvgZonesArray) > 100
```

```
    array.shift(keyArrowFvgZonesArray)
  if array.size(keyArrowFvgZonesArray) > 0
    for i = array.size(keyArrowFvgZonesArray) - 1 to 0
      KeyArrowFvgZone zone = array.get(keyArrowFvgZonesArray, i)
      if zone.active
        bool zoneExpired = bar_index - zone.createdBar > keyArrowFvgMaxAgeInput
        bool zoneInvalidated = zone.isBull ? low <= zone.bottom : high >= zone.top
        if zoneExpired or zoneInvalidated
          zone.active := false
        else if bar_index > zone.createdBar
          bool candleTouchesZone = high >= zone.bottom and low <= zone.top
          if zone.isBull and candleTouchesZone and close > zone.top and low > zone.bottom and fvgFastEma > fvgSlowEma and close > fvgFastEma and close > fvgSlowEma
            bullishArrowFvgConfluence := true
          if not zone.isBull and candleTouchesZone and close < zone.bottom and high < zone.top and fvgFastEma < fvgSlowEma and close < fvgFastEma and close < fvgSlowEma
            bearishArrowFvgConfluence := true
        array.set(keyArrowFvgZonesArray, i, zone)


if enableFvg123Input and bar_index >= 2
  if low > high[2]
```

```
    array.push(fvg123SetupsArray, Fvg123Setup.new(low, high[2], true, bar_index, lastFvgKeyLevelTouchBar, false, na, false))
  if high < low[2]
    array.push(fvg123SetupsArray, Fvg123Setup.new(low[2], high, false, bar_index, lastFvgKeyLevelTouchBar, false, na, false))
  if array.size(fvg123SetupsArray) > 100
    array.shift(fvg123SetupsArray)
  if array.size(fvg123SetupsArray) > 0
    for i = array.size(fvg123SetupsArray) - 1 to 0
      Fvg123Setup fvgObj = array.get(fvg123SetupsArray, i)
      if not fvgObj.done
        bool fullyFilled = fvgObj.isBull ? low <= fvgObj.bottom : high >= fvgObj.top
        if fullyFilled
          fvgObj.done := true
        else
          bool 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.touchBar := bar_index
          if afterCreation and (fvgObj.isBull ? close < fvgObj.bottom : close > fvgObj.top)
            fvgObj.done := true
```

```
        bool hasLaterTouch = fvgObj.touched and not na(fvgObj.touchBar) and bar_index > fvgObj.touchBar

        bool fvgBreakout = (fvgObj.isBull and hasLaterTouch and close > fvgObj.top) or (not fvgObj.isBull and hasLaterTouch and close < fvgObj.bottom)

        bool fvgAgeValid = bar_index - fvgObj.createdBar <= fvg123MaxAgeInput

        bool fvgWidthValid = math.abs(fvgObj.top - fvgObj.bottom) / syminfo.mintick >= fvg123MinWidthTicksInput

        bool keyTouchBeforeFvg = not na(fvgObj.keyTouchBar) and fvgObj.keyTouchBar < fvgObj.createdBar and bar_index - fvgObj.keyTouchBar <= fvg123KeyTouchLookbackInput

        bool fvgKeyTouchAllowed = not requireFvg123KeyTouchInput or keyTouchBeforeFvg

        bool fvgOpeningRangeCloseAllowed = not requireFvg123OpeningRangeBreakInput or (fvgOpeningRange15Complete and not na(fvgOpeningRange15High) and not na(fvgOpeningRange15Low) and (fvgObj.isBull ? close > fvgOpeningRange15High : close < fvgOpeningRange15Low))

        if fvgBreakout

            bool fvgTrendFlowAllowed = fvgObj.isBull ? fvgFastEma > fvgSlowEma and close > fvgFastEma and close > fvgSlowEma : fvgFastEma < fvgSlowEma and close < fvgFastEma and close < fvgSlowEma

            bool fvgCrossAfterKeyTouch = not requireFvg123KeyTouchInput or not na(fvgObj.keyTouchBar) and not na(fvgObj.isBull ? lastFvgBullEmaCrossBar : lastFvgBearEmaCrossBar) and fvgObj.keyTouchBar < (fvgObj.isBull ? lastFvgBullEmaCrossBar : lastFvgBearEmaCrossBar)
```

```
        bool fvgEmaCrossAvailable = fvgObj.isBull ? fvgBullEmaContext and fvgCrossAfterKeyTouch and (na(fvg123EmaCrossUsedBar) or fvg123EmaCrossUsedBar != lastFvgBullEmaCrossBar) : fvgBearEmaContext and fvgCrossAfterKeyTouch and (na(fvg123EmaCrossUsedBar) or fvg123EmaCrossUsedBar != lastFvgBearEmaCrossBar)

        bool fvgEmaCrossAllowed = not requireFvg123EmaCrossInput or fvgEmaCrossAvailable

        if fvgAgeValid and fvgWidthValid and fvgKeyTouchAllowed and fvgOpeningRangeCloseAllowed and fvgTrendFlowAllowed and fvgEmaCrossAllowed and barstate.isconfirmed and not fvgLongSignal and not fvgShortSignal

            float entryPrice = close

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

            float risk = fvgObj.isBull ? entryPrice - stopPrice : stopPrice - entryPrice

            if risk > 0.0

              fvgSignalEntry := entryPrice

              fvgSignalStop := stopPrice

              fvgSignalTarget := fvgObj.isBull ? entryPrice + risk * fvgRewardRiskInput : entryPrice - risk * fvgRewardRiskInput

              if requireFvg123EmaCrossInput

                fvg123EmaCrossUsedBar := fvgObj.isBull ? lastFvgBullEmaCrossBar : lastFvgBearEmaCrossBar

              if fvgObj.isBull

                fvgLongSignal := true

              else
```

```
                    fvgShortSignal := true
            fvgObj.done := true
        array.set(fvg123SetupsArray, i, fvgObj)

// --- Five-minute CISD and same-direction FVG confluence ---
bool isFiveMinuteChart = timeframe.isintraday and timeframe.isminutes and
timeframe.multiplier == 5
bool cisdPdhCrossUp = ta.crossover(close, previousDayHigh)
bool cisdPwhCrossUp = ta.crossover(close, previousWeekHigh)
bool cisdAsiaHighCrossUp = ta.crossover(close, previousAsiaHigh)
bool cisdLondonHighCrossUp = ta.crossover(close, previousLondonHigh)
bool cisdPdlCrossDown = ta.crossunder(close, previousDayLow)
bool cisdPwlCrossDown = ta.crossunder(close, previousWeekLow)
bool cisdAsiaLowCrossDown = ta.crossunder(close, previousAsiaLow)
bool cisdLondonLowCrossDown = ta.crossunder(close, previousLondonLow)

float cisdHighBreachLevel = na
if useDailyLevelsInput and cisdPdhCrossUp
   cisdHighBreachLevel := previousDayHigh
if useWeeklyLevelsInput and cisdPwhCrossUp
   cisdHighBreachLevel := na(cisdHighBreachLevel) ? previousWeekHigh :
math.max(cisdHighBreachLevel, previousWeekHigh)
if useAsiaLevelsInput and cisdAsiaHighCrossUp
```

```
   cisdHighBreachLevel := na(cisdHighBreachLevel) ? previousAsiaHigh :
math.max(cisdHighBreachLevel, previousAsiaHigh)

if useLondonLevelsInput and cisdLondonHighCrossUp

   cisdHighBreachLevel := na(cisdHighBreachLevel) ? previousLondonHigh :
math.max(cisdHighBreachLevel, previousLondonHigh)


float cisdLowBreachLevel = na

if useDailyLevelsInput and cisdPdlCrossDown

   cisdLowBreachLevel := previousDayLow

if useWeeklyLevelsInput and cisdPwlCrossDown

   cisdLowBreachLevel := na(cisdLowBreachLevel) ? previousWeekLow :
math.min(cisdLowBreachLevel, previousWeekLow)

if useAsiaLevelsInput and cisdAsiaLowCrossDown

   cisdLowBreachLevel := na(cisdLowBreachLevel) ? previousAsiaLow :
math.min(cisdLowBreachLevel, previousAsiaLow)

if useLondonLevelsInput and cisdLondonLowCrossDown

   cisdLowBreachLevel := na(cisdLowBreachLevel) ? previousLondonLow :
math.min(cisdLowBreachLevel, previousLondonLow)


if enableFiveMinuteCisdInput and isFiveMinuteChart and not
na(cisdHighBreachLevel)

   cisdBearPending := true

   cisdBearKeyLevel := cisdHighBreachLevel

   cisdBearBreachLow := low

   cisdBearStructureHigh := high
```

```
    cisdBearStartBar := bar_index


if enableFiveMinuteCisdInput and isFiveMinuteChart and not
na(cisdLowBreachLevel)
    cisdBullPending := true
    cisdBullKeyLevel := cisdLowBreachLevel
    cisdBullBreachHigh := high
    cisdBullStructureLow := low
    cisdBullStartBar := bar_index


if enableFiveMinuteCisdInput and isFiveMinuteChart
    if cisdBearPending
        cisdBearStructureHigh := math.max(cisdBearStructureHigh, high)
    if cisdBullPending
        cisdBullStructureLow := math.min(cisdBullStructureLow, low)


bool cisdBearReaccepted = cisdBearPending and not na(cisdBearKeyLevel)
and close > cisdBearKeyLevel and low > cisdBearBreachLow and bar_index -
cisdBearStartBar > 10
bool cisdBullReaccepted = cisdBullPending and not na(cisdBullKeyLevel) and
close < cisdBullKeyLevel and high < cisdBullBreachHigh and bar_index -
cisdBullStartBar > 10
if cisdBearReaccepted
    cisdBearPending := false
if cisdBullReaccepted
```

```
    cisdBullPending := false


bool rawFiveMinuteBearCisd = enableFiveMinuteCisdInput and
isFiveMinuteChart and cisdBearPending and not na(cisdBearKeyLevel) and
close < cisdBearKeyLevel and close < cisdBearBreachLow and
barstate.isconfirmed

bool rawFiveMinuteBullCisd = enableFiveMinuteCisdInput and
isFiveMinuteChart and cisdBullPending and not na(cisdBullKeyLevel) and
close > cisdBullKeyLevel and close > cisdBullBreachHigh and
barstate.isconfirmed

bool qualifiedFiveMinuteBearCisd = rawFiveMinuteBearCisd and not
rawFiveMinuteBullCisd

bool qualifiedFiveMinuteBullCisd = rawFiveMinuteBullCisd and not
rawFiveMinuteBearCisd


// Count completed CISD events by the selected key-level price; only event
two creates a trade setup.

if rawFiveMinuteBearCisd

    if qualifiedFiveMinuteBearCisd

        if na(lastBearCisdCountLevel) or cisdBearKeyLevel !=
lastBearCisdCountLevel

            lastBearCisdCountLevel := cisdBearKeyLevel

            bearCisdEventCount := 1

        else

            bearCisdEventCount += 1

        if bearCisdEventCount == 2
```

```
        float entryPrice = close
        float stopPrice = cisdBearStructureHigh + syminfo.mintick
        float risk = stopPrice - entryPrice
        if risk > 0.0
          cisdSignalEntry := entryPrice
          cisdSignalStop := stopPrice
          cisdSignalTargetOneR := entryPrice - risk
          cisdSignalTargetTwoR := entryPrice - risk * 2.0
          cisdBearSignal := true
    cisdBearPending := false

if rawFiveMinuteBullCisd
    if qualifiedFiveMinuteBullCisd
      if na(lastBullCisdCountLevel) or cisdBullKeyLevel !=
lastBullCisdCountLevel
        lastBullCisdCountLevel := cisdBullKeyLevel
        bullCisdEventCount := 1
      else
        bullCisdEventCount += 1
      if bullCisdEventCount == 2
        float entryPrice = close
        float stopPrice = cisdBullStructureLow - syminfo.mintick
        float risk = entryPrice - stopPrice
```

```
        if risk > 0.0
            cisdSignalEntry := entryPrice
            cisdSignalStop := stopPrice
            cisdSignalTargetOneR := entryPrice + risk
            cisdSignalTargetTwoR := entryPrice + risk * 2.0
            cisdBullSignal := true
    cisdBullPending := false


// --- Confirmed 15-minute key-level CISD ---
[cisd15High, cisd15Low, cisd15Close, cisd15PreviousClose, cisd15EmaFast, cisd15EmaSlow] = request.security(syminfo.tickerid, "15", [high, low, close, close[1], ta.ema(close, fvgFastEmaLengthInput), ta.ema(close, fvgSlowEmaLengthInput)], gaps = barmerge.gaps_off, lookahead = barmerge.lookahead_off)
bool is15MinuteOrLowerChart = timeframe.isintraday and timeframe.in_seconds() <= 900
bool isFifteenMinuteChart = timeframe.isintraday and timeframe.in_seconds() == 900
bool isOneMinuteChart = timeframe.isminutes and timeframe.multiplier == 1
bool is15MinuteCandleClose = is15MinuteOrLowerChart and barstate.isconfirmed and time_close == time_close("15")
bool cisd15BullEmaFlow = not na(cisd15EmaFast) and not na(cisd15EmaSlow) and cisd15EmaFast > cisd15EmaSlow
bool cisd15BearEmaFlow = not na(cisd15EmaFast) and not na(cisd15EmaSlow) and cisd15EmaFast < cisd15EmaSlow
bool cisd15BullReversalSignal = false
```

```
bool cisd15BearReversalSignal = false

bool cisd15BullBreakoutSignal = false

bool cisd15BearBreakoutSignal = false

bool cisd15BullTradeSignal = false

bool cisd15BearTradeSignal = false

float cisd15TradeEntry = na

float cisd15TradeStop = na

float cisd15TradeTarget = na


if enableFifteenMinuteCisdInput and is15MinuteCandleClose and not
na(cisd15Close) and not na(cisd15PreviousClose)

   bool cisd15BearSweepAtHigh = (useDailyLevelsInput and not
na(previousDayHigh) and cisd15High > previousDayHigh and cisd15Close <
previousDayHigh) or (useWeeklyLevelsInput and not na(previousWeekHigh)
and cisd15High > previousWeekHigh and cisd15Close < previousWeekHigh)
or (useAsiaLevelsInput and not na(previousAsiaHigh) and cisd15High >
previousAsiaHigh and cisd15Close < previousAsiaHigh) or
(useLondonLevelsInput and not na(previousLondonHigh) and cisd15High >
previousLondonHigh and cisd15Close < previousLondonHigh) or
(useOpeningRangeLevelsInput and openingRangeComplete and not
na(openingRangeHigh) and cisd15High > openingRangeHigh and cisd15Close
< openingRangeHigh)

   bool cisd15BullSweepAtLow = (useDailyLevelsInput and not
na(previousDayLow) and cisd15Low < previousDayLow and cisd15Close >
previousDayLow) or (useWeeklyLevelsInput and not na(previousWeekLow)
and cisd15Low < previousWeekLow and cisd15Close > previousWeekLow) or
(useAsiaLevelsInput and not na(previousAsiaLow) and cisd15Low <
previousAsiaLow and cisd15Close > previousAsiaLow) or
```

(useLondonLevelsInput and not na(previousLondonLow) and cisd15Low < previousLondonLow and cisd15Close > previousLondonLow) or (useOpeningRangeLevelsInput and openingRangeComplete and not na(openingRangeLow) and cisd15Low < openingRangeLow and cisd15Close > openingRangeLow)

bool cisd15BullCloseThrough = (useDailyLevelsInput and not na(previousDayHigh) and cisd15PreviousClose <= previousDayHigh and cisd15Close > previousDayHigh) or (useWeeklyLevelsInput and not na(previousWeekHigh) and cisd15PreviousClose <= previousWeekHigh and cisd15Close > previousWeekHigh) or (useAsiaLevelsInput and not na(previousAsiaHigh) and cisd15PreviousClose <= previousAsiaHigh and cisd15Close > previousAsiaHigh) or (useLondonLevelsInput and not na(previousLondonHigh) and cisd15PreviousClose <= previousLondonHigh and cisd15Close > previousLondonHigh) or (useOpeningRangeLevelsInput and openingRangeComplete and not na(openingRangeHigh) and cisd15PreviousClose <= openingRangeHigh and cisd15Close > openingRangeHigh)

bool cisd15BearCloseThrough = (useDailyLevelsInput and not na(previousDayLow) and cisd15PreviousClose >= previousDayLow and cisd15Close < previousDayLow) or (useWeeklyLevelsInput and not na(previousWeekLow) and cisd15PreviousClose >= previousWeekLow and cisd15Close < previousWeekLow) or (useAsiaLevelsInput and not na(previousAsiaLow) and cisd15PreviousClose >= previousAsiaLow and cisd15Close < previousAsiaLow) or (useLondonLevelsInput and not na(previousLondonLow) and cisd15PreviousClose >= previousLondonLow and cisd15Close < previousLondonLow) or (useOpeningRangeLevelsInput and openingRangeComplete and not na(openingRangeLow) and cisd15PreviousClose >= openingRangeLow and cisd15Close < openingRangeLow)

cisd15BearReversalSignal := cisd15BearSweepAtHigh and cisd15BearEmaFlow

```
    cisd15BullReversalSignal := cisd15BullSweepAtLow and cisd15BullEmaFlow

    cisd15BullBreakoutSignal := cisd15BullCloseThrough and cisd15BullEmaFlow

    cisd15BearBreakoutSignal := cisd15BearCloseThrough and cisd15BearEmaFlow

    bool cisd15BullEntryCandidate = cisd15BullReversalSignal or cisd15BullBreakoutSignal

    bool cisd15BearEntryCandidate = cisd15BearReversalSignal or cisd15BearBreakoutSignal

    if cisd15BullEntryCandidate and not cisd15BearEntryCandidate

        float bullEntryPrice15 = cisd15Close

        float bullStopPrice15 = cisd15Low - syminfo.mintick

        float bullRisk15 = bullEntryPrice15 - bullStopPrice15

        if bullRisk15 > 0.0

            cisd15BullTradeSignal := true

            cisd15TradeEntry := bullEntryPrice15

            cisd15TradeStop := bullStopPrice15

            cisd15TradeTarget := bullEntryPrice15 + bullRisk15

    else if cisd15BearEntryCandidate and not cisd15BullEntryCandidate

        float bearEntryPrice15 = cisd15Close

        float bearStopPrice15 = cisd15High + syminfo.mintick

        float bearRisk15 = bearStopPrice15 - bearEntryPrice15

        if bearRisk15 > 0.0

            cisd15BearTradeSignal := true
```

```
        cisd15TradeEntry := bearEntryPrice15
        cisd15TradeStop := bearStopPrice15
        cisd15TradeTarget := bearEntryPrice15 - bearRisk15

// --- 15-minute CISD 1:1 risk/reward guides (15-minute chart only) ---
if isFifteenMinuteChart and showFifteenMinuteCisdRiskRewardInput and
(cisd15BullTradeSignal or cisd15BearTradeSignal)
   int cisd15ObjectLimit = maxDisplayedFifteenMinuteCisdSetupsInput * 3
   while array.size(cisd15RiskRewardLinesArray) + 3 > cisd15ObjectLimit
      for i = 0 to 2
         line.delete(array.shift(cisd15RiskRewardLinesArray))
   int cisd15GuideEndBar = bar_index + fifteenMinuteCisdProjectionBarsInput
   line cisd15EntryLine = line.new(bar_index, cisd15TradeEntry,
cisd15GuideEndBar, cisd15TradeEntry, xloc = xloc.bar_index, color =
color.new(color.gray, 5), style = line.style_dotted, width =
fvgRiskRewardLineWidthInput)
   line cisd15StopLine = line.new(bar_index, cisd15TradeStop,
cisd15GuideEndBar, cisd15TradeStop, xloc = xloc.bar_index, color =
color.new(bearishColorInput, 5), style = line.style_dotted, width =
fvgRiskRewardLineWidthInput)
   line cisd15TargetLine = line.new(bar_index, cisd15TradeTarget,
cisd15GuideEndBar, cisd15TradeTarget, xloc = xloc.bar_index, color =
color.new(bullishColorInput, 5), style = line.style_dotted, width =
fvgRiskRewardLineWidthInput)
   array.push(cisd15RiskRewardLinesArray, cisd15EntryLine)
   array.push(cisd15RiskRewardLinesArray, cisd15StopLine)
```

```
    array.push(cisd15RiskRewardLinesArray, cisd15TargetLine)

// --- 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 sweepBullSignal = false
bool sweepBearSignal = false
float sweepSignalEntry = na
float sweepSignalStop = na
float sweepSignalTarget = na
bool keyLevelBullSignal = false
bool keyLevelBearSignal = false
float keyLevelSignalEntry = na
float keyLevelSignalStop = na
float keyLevelSignalTarget = na
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

float sweepStopBuffer = math.max(syminfo.mintick, nz(atrValue,
syminfo.mintick) * sweepStopBufferAtrInput)

if bullishEntrySignal and barstate.isconfirmed and not
na(bullishSweepExtreme)

   float sweepBullEntryPrice = close

   float sweepBullStopPrice = bullishSweepExtreme - sweepStopBuffer

   float sweepBullRisk = sweepBullEntryPrice - sweepBullStopPrice

   if sweepBullRisk > 0.0
```

```
    sweepSignalEntry := sweepBullEntryPrice
    sweepSignalStop := sweepBullStopPrice
    sweepSignalTarget := sweepBullEntryPrice + sweepBullRisk
    sweepBullSignal := true
if bearishEntrySignal and barstate.isconfirmed and not
na(bearishSweepExtreme)
  float sweepBearEntryPrice = close
  float sweepBearStopPrice = bearishSweepExtreme + sweepStopBuffer
  float sweepBearRisk = sweepBearStopPrice - sweepBearEntryPrice
  if sweepBearRisk > 0.0
    sweepSignalEntry := sweepBearEntryPrice
    sweepSignalStop := sweepBearStopPrice
    sweepSignalTarget := sweepBearEntryPrice - sweepBearRisk
    sweepBearSignal := true
bool bullishKeyEntrySignal = pendingBullishKey and bullishKeyWithinWindow
and bar_index > bullishKeySweepIndex and bullishKeyMssCross
bool bearishKeyEntrySignal = pendingBearishKey and
bearishKeyWithinWindow and bar_index > bearishKeySweepIndex and
bearishKeyMssCross
float keyLevelStopBuffer = math.max(syminfo.mintick, nz(atrValue,
syminfo.mintick) * keyLevelStopBufferAtrInput)
if bullishKeyEntrySignal and barstate.isconfirmed and not
na(bullishKeySweepExtreme)
  float bullKeyEntryPrice = close
  float bullKeyStopPrice = bullishKeySweepExtreme - keyLevelStopBuffer
```

```
    float bullKeyRisk = bullKeyEntryPrice - bullKeyStopPrice
    if bullKeyRisk > 0.0
        keyLevelSignalEntry := bullKeyEntryPrice
        keyLevelSignalStop := bullKeyStopPrice
        keyLevelSignalTarget := bullKeyEntryPrice + bullKeyRisk *
keyLevelRewardRiskInput
        keyLevelBullSignal := true
if bearishKeyEntrySignal and barstate.isconfirmed and not
na(bearishKeySweepExtreme)
    float bearKeyEntryPrice = close
    float bearKeyStopPrice = bearishKeySweepExtreme + keyLevelStopBuffer
    float bearKeyRisk = bearKeyStopPrice - bearKeyEntryPrice
    if bearKeyRisk > 0.0
        keyLevelSignalEntry := bearKeyEntryPrice
        keyLevelSignalStop := bearKeyStopPrice
        keyLevelSignalTarget := bearKeyEntryPrice - bearKeyRisk *
keyLevelRewardRiskInput
        keyLevelBearSignal := true


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

// --- Regular sweep 1R risk/reward guides ---
if showSweepRiskRewardInput and (sweepBullSignal or sweepBearSignal)
    int sweepObjectLimit = maxDisplayedSweepSetupsInput * 3
    while array.size(sweepRiskRewardLinesArray) + 3 > sweepObjectLimit
```

```
    for i = 0 to 2
        line.delete(array.shift(sweepRiskRewardLinesArray))
    if array.size(sweepRiskRewardLabelsArray) >= 3
        for i = 0 to 2
            label.delete(array.shift(sweepRiskRewardLabelsArray))
  int sweepGuideEndBar = bar_index + sweepRiskRewardProjectionBarsInput
  line sweepEntryLine = line.new(bar_index, sweepSignalEntry,
sweepGuideEndBar, sweepSignalEntry, xloc = xloc.bar_index, color =
color.new(color.gray, 10), style = line.style_dotted, width =
sweepRiskRewardLineWidthInput)
  line sweepStopLine = line.new(bar_index, sweepSignalStop,
sweepGuideEndBar, sweepSignalStop, xloc = xloc.bar_index, color =
color.new(bearishColorInput, 5), style = line.style_dotted, width =
sweepRiskRewardLineWidthInput)
  line sweepTargetLine = line.new(bar_index, sweepSignalTarget,
sweepGuideEndBar, sweepSignalTarget, xloc = xloc.bar_index, color =
color.new(bullishColorInput, 5), style = line.style_dotted, width =
sweepRiskRewardLineWidthInput)
  array.push(sweepRiskRewardLinesArray, sweepEntryLine)
  array.push(sweepRiskRewardLinesArray, sweepStopLine)
  array.push(sweepRiskRewardLinesArray, sweepTargetLine)
  if showSweepRiskRewardLabelsInput
    label sweepEntryLabel = label.new(sweepGuideEndBar,
sweepSignalEntry, "Entry", xloc = xloc.bar_index, yloc = yloc.price, color =
color.new(color.gray, 15), style = label.style_label_left, textcolor =
chart.fg_color, size = size.tiny)
```

```
    label sweepStopLabel = label.new(sweepGuideEndBar, sweepSignalStop, "SL", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(bearishColorInput, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)

    label sweepTargetLabel = label.new(sweepGuideEndBar, sweepSignalTarget, "TP 1R", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(bullishColorInput, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)

    array.push(sweepRiskRewardLabelsArray, sweepEntryLabel)

    array.push(sweepRiskRewardLabelsArray, sweepStopLabel)

    array.push(sweepRiskRewardLabelsArray, sweepTargetLabel)


// --- Key-level risk/reward guides ---

if showKeyLevelRiskRewardInput and (keyLevelBullSignal or keyLevelBearSignal)

  int keyLevelObjectLimit = maxDisplayedKeyLevelSetupsInput * 3

  while array.size(keyLevelRiskRewardLinesArray) + 3 > keyLevelObjectLimit

    for i = 0 to 2

      line.delete(array.shift(keyLevelRiskRewardLinesArray))

    if array.size(keyLevelRiskRewardLabelsArray) >= 3

      for i = 0 to 2

        label.delete(array.shift(keyLevelRiskRewardLabelsArray))

  int keyLevelGuideEndBar = bar_index + keyLevelProjectionBarsInput

  line keyLevelEntryLine = line.new(bar_index, keyLevelSignalEntry, keyLevelGuideEndBar, keyLevelSignalEntry, xloc = xloc.bar_index, color =
```

```
color.new(color.gray, 10), style = line.style_dotted, width =
keyLevelRiskRewardLineWidthInput)

   line keyLevelStopLine = line.new(bar_index, keyLevelSignalStop,
keyLevelGuideEndBar, keyLevelSignalStop, xloc = xloc.bar_index, color =
color.new(bearishColorInput, 5), style = line.style_dotted, width =
keyLevelRiskRewardLineWidthInput)

   line keyLevelTargetLine = line.new(bar_index, keyLevelSignalTarget,
keyLevelGuideEndBar, keyLevelSignalTarget, xloc = xloc.bar_index, color =
color.new(bullishColorInput, 5), style = line.style_dotted, width =
keyLevelRiskRewardLineWidthInput)

   array.push(keyLevelRiskRewardLinesArray, keyLevelEntryLine)

   array.push(keyLevelRiskRewardLinesArray, keyLevelStopLine)

   array.push(keyLevelRiskRewardLinesArray, keyLevelTargetLine)

   if showKeyLevelRiskRewardLabelsInput

      label keyLevelEntryLabel = label.new(keyLevelGuideEndBar,
keyLevelSignalEntry, "Key Entry", xloc = xloc.bar_index, yloc = yloc.price, color
= color.new(color.gray, 15), style = label.style_label_left, textcolor =
chart.fg_color, size = size.tiny)

      label keyLevelStopLabel = label.new(keyLevelGuideEndBar,
keyLevelSignalStop, "Key SL", xloc = xloc.bar_index, yloc = yloc.price, color =
color.new(bearishColorInput, 15), style = label.style_label_left, textcolor =
chart.fg_color, size = size.tiny)

      label keyLevelTargetLabel = label.new(keyLevelGuideEndBar,
keyLevelSignalTarget, "Key TP " + str.tostring(keyLevelRewardRiskInput, "#.##")
+ "R", xloc = xloc.bar_index, yloc = yloc.price, color =
color.new(bullishColorInput, 15), style = label.style_label_left, textcolor =
chart.fg_color, size = size.tiny)

      array.push(keyLevelRiskRewardLabelsArray, keyLevelEntryLabel)
```

```
    array.push(keyLevelRiskRewardLabelsArray, keyLevelStopLabel)
    array.push(keyLevelRiskRewardLabelsArray, keyLevelTargetLabel)


// --- FVG-123 risk/reward guides ---
if showFvgRiskRewardInput and (fvgLongSignal or fvgShortSignal)
  int objectLimit = maxDisplayedFvgSetupsInput * 3
  while array.size(fvgRiskRewardLinesArray) + 3 > objectLimit
    for i = 0 to 2
      line.delete(array.shift(fvgRiskRewardLinesArray))
    if array.size(fvgRiskRewardLabelsArray) >= 3
      for i = 0 to 2
        label.delete(array.shift(fvgRiskRewardLabelsArray))
  int guideEndBar = bar_index + fvgProjectionBarsInput
  line entryLine = line.new(bar_index, fvgSignalEntry, guideEndBar,
fvgSignalEntry, xloc = xloc.bar_index, color = color.new(color.gray, 10), style =
line.style_dotted, width = fvgRiskRewardLineWidthInput)
  line stopLine = line.new(bar_index, fvgSignalStop, guideEndBar,
fvgSignalStop, xloc = xloc.bar_index, color = color.new(bearishColorInput, 5),
style = line.style_dotted, width = fvgRiskRewardLineWidthInput)
  line targetLine = line.new(bar_index, fvgSignalTarget, guideEndBar,
fvgSignalTarget, xloc = xloc.bar_index, color = color.new(bullishColorInput, 5),
style = line.style_dotted, width = fvgRiskRewardLineWidthInput)
  array.push(fvgRiskRewardLinesArray, entryLine)
  array.push(fvgRiskRewardLinesArray, stopLine)
  array.push(fvgRiskRewardLinesArray, targetLine)
```

```
    if showFvgRiskRewardLabelsInput
        label entryLabel = label.new(guideEndBar, fvgSignalEntry, "Entry", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(color.gray, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)
        label stopLabel = label.new(guideEndBar, fvgSignalStop, "SL", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(bearishColorInput, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)
        label targetLabel = label.new(guideEndBar, fvgSignalTarget, "TP " + str.tostring(fvgRewardRiskInput, "#.##") + "R", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(bullishColorInput, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)
        array.push(fvgRiskRewardLabelsArray, entryLabel)
        array.push(fvgRiskRewardLabelsArray, stopLabel)
        array.push(fvgRiskRewardLabelsArray, targetLabel)

// --- Five-minute CISD risk/reward guides ---
if showCisdRiskRewardInput and (cisdBullSignal or cisdBearSignal)
    int cisdObjectLimit = maxDisplayedCisdSetupsInput * 4
    while array.size(cisdRiskRewardLinesArray) + 4 > cisdObjectLimit
        for i = 0 to 3
            line.delete(array.shift(cisdRiskRewardLinesArray))
        if array.size(cisdRiskRewardLabelsArray) >= 4
            for i = 0 to 3
                label.delete(array.shift(cisdRiskRewardLabelsArray))
    int cisdGuideEndBar = bar_index + cisdProjectionBarsInput
```

```
    line cisdEntryLine = line.new(bar_index, cisdSignalEntry, cisdGuideEndBar, cisdSignalEntry, xloc = xloc.bar_index, color = color.new(color.gray, 10), style = line.style_dotted, width = fvgRiskRewardLineWidthInput)

    line cisdStopLine = line.new(bar_index, cisdSignalStop, cisdGuideEndBar, cisdSignalStop, xloc = xloc.bar_index, color = color.new(bearishColorInput, 5), style = line.style_dotted, width = fvgRiskRewardLineWidthInput)

    line cisdTargetOneRLine = line.new(bar_index, cisdSignalTargetOneR, cisdGuideEndBar, cisdSignalTargetOneR, xloc = xloc.bar_index, color = color.new(LEVEL_COLOR, 5), style = line.style_dotted, width = fvgRiskRewardLineWidthInput)

    line cisdTargetTwoRLine = line.new(bar_index, cisdSignalTargetTwoR, cisdGuideEndBar, cisdSignalTargetTwoR, xloc = xloc.bar_index, color = color.new(bullishColorInput, 5), style = line.style_dotted, width = fvgRiskRewardLineWidthInput)

    array.push(cisdRiskRewardLinesArray, cisdEntryLine)

    array.push(cisdRiskRewardLinesArray, cisdStopLine)

    array.push(cisdRiskRewardLinesArray, cisdTargetOneRLine)

    array.push(cisdRiskRewardLinesArray, cisdTargetTwoRLine)

    if showCisdRiskRewardLabelsInput

        label cisdEntryLabel = label.new(cisdGuideEndBar, cisdSignalEntry, "CISD Entry", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(color.gray, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)

        label cisdStopLabel = label.new(cisdGuideEndBar, cisdSignalStop, "CISD SL", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(bearishColorInput, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)
```

```
    label cisdTargetOneRLabel = label.new(cisdGuideEndBar, cisdSignalTargetOneR, "CISD TP 1R", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(LEVEL_COLOR, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)

    label cisdTargetTwoRLabel = label.new(cisdGuideEndBar, cisdSignalTargetTwoR, "CISD TP 2R", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(bullishColorInput, 15), style = label.style_label_left, textcolor = chart.fg_color, size = size.tiny)

    array.push(cisdRiskRewardLabelsArray, cisdEntryLabel)

    array.push(cisdRiskRewardLabelsArray, cisdStopLabel)

    array.push(cisdRiskRewardLabelsArray, cisdTargetOneRLabel)

    array.push(cisdRiskRewardLabelsArray, cisdTargetTwoRLabel)


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

plotshape(showSweepMarkersInput and 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)

plotshape(showKeyLevelEntrySignalsInput and keyLevelBullSignal, title = "Bullish key-level MSS 1R", style = shape.triangleup, location = location.belowbar, color = bullishColorInput, text = "KEY LONG", textcolor = chart.fg_color, size = size.small)

plotshape(showKeyLevelEntrySignalsInput and keyLevelBearSignal, title = "Bearish key-level MSS 1R", style = shape.triangledown, location = location.abovebar, color = bearishColorInput, text = "KEY SHORT", textcolor = chart.fg_color, size = size.small)

plotshape(showFvg123SignalsInput and fvgLongSignal, title = "Bullish FVG-123 2R signal", style = shape.triangleup, location = location.belowbar, color = bullishColorInput, text = "FVG-123", textcolor = chart.fg_color, size = size.small)

plotshape(showFvg123SignalsInput and fvgShortSignal, title = "Bearish FVG-123 2R signal", style = shape.triangledown, location = location.abovebar, color = bearishColorInput, text = "FVG-123", textcolor = chart.fg_color, size = size.small)

plotshape(showFiveMinuteCisdSignalsInput and cisdBullSignal, title = "Bullish Second Key-Level CISD", style = shape.triangleup, location = location.belowbar, color = bullishColorInput, text = "2nd CISD", textcolor = chart.fg_color, size = size.small)

plotshape(showFiveMinuteCisdSignalsInput and cisdBearSignal, title = "Bearish Second Key-Level CISD", style = shape.triangledown, location =
```

```
location.abovebar, color = bearishColorInput, text = "2nd CISD", textcolor =
chart.fg_color, size = size.small)

plotshape(showFifteenMinuteCisdSignalsInput and not isFifteenMinuteChart
and not (cisd15BullTradeSignal or cisd15BearTradeSignal) and
cisd15BullReversalSignal, title = "Bullish 15-minute CISD sweep reclaim",
style = shape.triangleup, location = location.abovebar, color =
bullishColorInput, text = "15m REV", textcolor = chart.fg_color, size = size.tiny)

plotshape(showFifteenMinuteCisdSignalsInput and not isFifteenMinuteChart
and not (cisd15BullTradeSignal or cisd15BearTradeSignal) and
cisd15BearReversalSignal, title = "Bearish 15-minute CISD sweep reclaim",
style = shape.triangledown, location = location.belowbar, color =
bearishColorInput, text = "15m REV", textcolor = chart.fg_color, size = size.tiny)

plotshape(showFifteenMinuteCisdSignalsInput and not isFifteenMinuteChart
and not (cisd15BullTradeSignal or cisd15BearTradeSignal) and
cisd15BullBreakoutSignal, title = "Bullish 15-minute CISD close through level",
style = shape.triangleup, location = location.abovebar, color =
bullishColorInput, text = "15m BRK", textcolor = chart.fg_color, size = size.tiny)

plotshape(showFifteenMinuteCisdSignalsInput and not isFifteenMinuteChart
and not (cisd15BullTradeSignal or cisd15BearTradeSignal) and
cisd15BearBreakoutSignal, title = "Bearish 15-minute CISD close through
level", style = shape.triangledown, location = location.belowbar, color =
bearishColorInput, text = "15m BRK", textcolor = chart.fg_color, size = size.tiny)

float cisd15MarkerTextOffset = not na(atrValue) ? math.max(atrValue *
cisd15TextOffsetAtrInput, syminfo.mintick * 8.0) : syminfo.mintick * 8.0

if showFifteenMinuteCisdSignalsInput and cisd15BullTradeSignal

   float cisd15BullArrowPrice = isFifteenMinuteChart ? low : high

   float cisd15BullTextPrice = isFifteenMinuteChart ? high +
cisd15MarkerTextOffset : low - cisd15MarkerTextOffset
```

```
    label.new(bar_index, cisd15BullArrowPrice, "", xloc = xloc.bar_index, yloc = isFifteenMinuteChart ? yloc.belowbar : yloc.abovebar, color = bullishColorInput, style = label.style_triangleup, textcolor = chart.fg_color, size = cisd15ArrowSizeInput)

    label.new(bar_index, cisd15BullTextPrice, "CISD", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(bullishColorInput, 100), style = label.style_none, textcolor = bullishColorInput, size = cisd15TextSizeInput)

if showFifteenMinuteCisdSignalsInput and cisd15BearTradeSignal

    float cisd15BearArrowPrice = isFifteenMinuteChart ? high : low

    float cisd15BearTextPrice = isFifteenMinuteChart ? low - cisd15MarkerTextOffset : high + cisd15MarkerTextOffset

    label.new(bar_index, cisd15BearArrowPrice, "", xloc = xloc.bar_index, yloc = isFifteenMinuteChart ? yloc.abovebar : yloc.belowbar, color = bearishColorInput, style = label.style_triangledown, textcolor = chart.fg_color, size = cisd15ArrowSizeInput)

    label.new(bar_index, cisd15BearTextPrice, "CISD", xloc = xloc.bar_index, yloc = yloc.price, color = color.new(bearishColorInput, 100), style = label.style_none, textcolor = bearishColorInput, size = cisd15TextSizeInput)


// 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)

// --- Combined FVG and HTF/LTF alerts ---
bool anyFvg123KeyEmaSignal = requireFvg123KeyTouchInput and requireFvg123EmaCrossInput and (fvgLongSignal or fvgShortSignal)
bool anyKeyLevelArrowFvgSignal = (keyLevelBullSignal and bullishArrowFvgConfluence) or (keyLevelBearSignal and bearishArrowFvgConfluence)
alertcondition(anyFvg123KeyEmaSignal, title = "Any FVG-123 after Key-Level Touch + 9/20 EMA Cross", message = "FVG-123 confirmed after a qualifying key-level touch and 9/20 EMA cross on {{ticker}} {{interval}}.")
alertcondition(anyKeyLevelArrowFvgSignal, title = "Any Key-Level Arrow + FVG/9-20 Confirmation", message = "Key-level sweep/MSS arrow confirmed with a same-direction active FVG retest and 9/20 EMA trend flow on {{ticker}} {{interval}}.")
bool oneMinuteBullTradeEntrySignal = isOneMinuteChart and barstate.isconfirmed and (bullishEntrySignal or keyLevelBullSignal or fvgLongSignal)
```

```
bool oneMinuteBearTradeEntrySignal = isOneMinuteChart and barstate.isconfirmed and (bearishEntrySignal or keyLevelBearSignal or fvgShortSignal)

bool alignedHtfLtfBullSignal = cisd15BullTradeSignal and oneMinuteBullTradeEntrySignal

bool alignedHtfLtfBearSignal = cisd15BearTradeSignal and oneMinuteBearTradeEntrySignal

alertcondition(alignedHtfLtfBullSignal or alignedHtfLtfBearSignal, title = "One Chart agreement HTF/LTF", message = "A confirmed 15-minute BRK/REV aligns with a same-direction 1-minute trade entry signal on {{ticker}}. Check the chart marker for direction.")

alertcondition(cisd15BullTradeSignal or cisd15BearTradeSignal, title = "15-minute CISD with 9/20 EMA", message = "A confirmed 15-minute CISD trade setup aligned with 9/20 EMA flow on {{ticker}} {{interval}}.")

bool cisd15Fvg123BullAgreement = isOneMinuteChart and cisd15BullTradeSignal and fvgLongSignal

bool cisd15Fvg123BearAgreement = isOneMinuteChart and cisd15BearTradeSignal and fvgShortSignal

alertcondition(cisd15Fvg123BullAgreement or cisd15Fvg123BearAgreement, title = "15 min CISD with 1 min FVG-123 - 9/20 EMA agreement", message = "A confirmed 15-minute CISD trade setup and 1-minute FVG-123 signal agree in direction with 9/20 EMA alignment on {{ticker}}.")


if isOneMinuteChart and (cisd15BearTradeSignal or keyLevelBearSignal)

   buyCisdFollowupCisdBar := na

else if isOneMinuteChart and cisd15BullTradeSignal and barstate.isconfirmed

   buyCisdFollowupCisdBar := bar_index
```

```
bool buyCisdThenKeyArrowSignal = isOneMinuteChart and barstate.isconfirmed and not na(buyCisdFollowupCisdBar) and keyLevelBullSignal and not na(bullishKeySweepIndex) and bullishKeySweepIndex > buyCisdFollowupCisdBar and bullishKeySweepIndex - buyCisdFollowupCisdBar <= buyCisdSweepWindowInput

bool buyCisdFollowupExpired = isOneMinuteChart and not na(buyCisdFollowupCisdBar) and bar_index - buyCisdFollowupCisdBar > buyCisdSweepWindowInput + confirmationWindowInput

if buyCisdThenKeyArrowSignal or buyCisdFollowupExpired

    buyCisdFollowupCisdBar := na


alertcondition(buyCisdThenKeyArrowSignal, title = "15-minute Buy CISD followed by key-level Buy Arrow", message = "A confirmed 15-minute bullish CISD was followed by a new bullish key-level sweep and MSS arrow within the configured window on {{ticker}} {{interval}}.")

alertcondition(keyLevelBullSignal or keyLevelBearSignal, title = "Key-Level Sweep + Arrow", message = "A confirmed key-level sweep was followed by a market-structure-shift arrow on {{ticker}} {{interval}}. Check the chart marker for direction.")

bool oneMinuteBullSweepArrowSignal = isOneMinuteChart and barstate.isconfirmed and (bullishEntrySignal or keyLevelBullSignal)

bool oneMinuteBearSweepArrowSignal = isOneMinuteChart and barstate.isconfirmed and (bearishEntrySignal or keyLevelBearSignal)

bool bullCisdSweepArrowAgreement = cisd15BullTradeSignal and oneMinuteBullSweepArrowSignal

bool bearCisdSweepArrowAgreement = cisd15BearTradeSignal and oneMinuteBearSweepArrowSignal
```

```
alertcondition(bullCisdSweepArrowAgreement or bearCisdSweepArrowAgreement, title = "15m CISD + 1m Sweep/Arrow Agreement", message = "A confirmed 15-minute CISD and same-direction 1-minute sweep/MSS arrow printed on the same bar for {{ticker}}. Check the chart for direction.")
```