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

//@version=6

indicator("MFB - 15min Accept-Rev-FVG", "MFB - 15min Accept-Rev-FVG", overlay = true, max_labels_count = 500)


// --- Constants ---

const color BULL_COLOR = #089981

const color BEAR_COLOR = #f23645

const color LEVEL_COLOR = #5b9cf6

const string SESSION_TZ = "America/New_York"


// --- Inputs ---

bool usePriorDayLevelsInput = input.bool(true, "Monitor prior-day high / low", tooltip = "Use the previous daily candle high and low as PDH and PDL.", group = "Key Levels")

bool usePriorWeekLevelsInput = input.bool(true, "Monitor prior-week high / low", tooltip = "Use the previous weekly candle high and low as PWH and PWL.", group = "Key Levels")

bool useVahInput = input.bool(false, "Use manual VAH", tooltip = "Enable the entered value-area high; this indicator does not calculate volume-profile levels.", group = "Key Levels")

float vahInput = input.float(0.0, "VAH price", tooltip = "Enter the value-area high when manual VAH is enabled.", group = "Key Levels")

bool useValInput = input.bool(false, "Use manual VAL", tooltip = "Enable the entered value-area low; this indicator does not calculate volume-profile levels.", group = "Key Levels")

float valInput = input.float(0.0, "VAL price", tooltip = "Enter the value-area low when manual VAL is enabled.", group = "Key Levels")

bool showKeyLevelsInput = input.bool(false, "Plot monitored levels", tooltip = "Display the enabled daily, weekly, session, initial-balance, and manual value-area levels. Plotting does not control whether levels are monitored.", group = "Key Levels")
```

string contextTfInput = input.timeframe("15", "Evaluation timeframe", tooltip = "Completed timeframe used for Acceptance and Reversal evaluation.", group = "Timeframes")

float directionalClosePctInput = input.float(67.0, "Directional closing strength (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Minimum directional close location within the candle range for Acceptance and Reversal.", group = "Acceptance / Reversal")

float acceptanceBodyPctInput = input.float(60.0, "Acceptance minimum body / range (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Minimum real-body size as a percentage of the candle range for Acceptance.", group = "Acceptance / Reversal")

float acceptanceBeyondPctInput = input.float(50.0, "Acceptance minimum range beyond level (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Minimum candle range on the breakout side of the key level for Acceptance.", group = "Acceptance / Reversal")

int reversalSweepTicksInput = input.int(2, "Reversal minimum sweep (ticks)", minval = 1, maxval = 1000, tooltip = "Minimum excursion through the key level before a Reversal reclaim, in whole instrument ticks.", group = "Acceptance / Reversal")

float reversalReclaimPctInput = input.float(25.0, "Reversal minimum reclaim (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Minimum close distance back beyond the level as a percentage of candle range.", group = "Acceptance / Reversal")

bool filterMinorAcceptanceInput = input.bool(true, "Filter minor Acceptance signals", tooltip = "When enabled, Acceptance must also meet the stricter thresholds below. Disable to use only the standard thresholds above.", group = "Signal Filters")

float filteredAcceptanceClosePctInput = input.float(75.0, "Filtered Acceptance close strength (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Higher minimum directional close strength used when minor Acceptance filtering is enabled.", group = "Signal Filters")

float filteredAcceptanceBodyPctInput = input.float(70.0, "Filtered Acceptance body / range (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Higher minimum body-to-range ratio used when minor Acceptance filtering is enabled.", group = "Signal Filters")

float filteredAcceptanceBeyondPctInput = input.float(60.0, "Filtered Acceptance range beyond level (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Higher minimum share of the candle range beyond the key level used when filtering Acceptance.", group = "Signal Filters")

bool filterMinorReversalInput = input.bool(true, "Filter minor Reversal signals", tooltip = "When enabled, Reversal must also meet the stricter thresholds below. Disable to use only the standard thresholds above.", group = "Signal Filters")

float filteredReversalClosePctInput = input.float(75.0, "Filtered Reversal close strength (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Higher minimum directional close strength used when minor Reversal filtering is enabled.", group = "Signal Filters")

int filteredReversalSweepTicksInput = input.int(4, "Filtered Reversal minimum sweep (ticks)", minval = 1, maxval = 1000, tooltip = "Larger minimum excursion through the level, in instrument ticks, used when filtering Reversal.", group = "Signal Filters")

float filteredReversalReclaimPctInput = input.float(35.0, "Filtered Reversal minimum reclaim (%)", minval = 0.0, maxval = 100.0, step = 1.0, tooltip = "Higher minimum close distance back beyond the level used when filtering Reversal.", group = "Signal Filters")

bool requireReversalCisdInput = input.bool(true, "Require CISD confirmation for Reversal", tooltip = "After a 15-minute sweep-and-reclaim candle, wait for a later confirmed close beyond that candle's opposite extreme at the same key level.", group = "Signal Filters")

bool requireAcceptanceFollowThroughInput = input.bool(true, "Require follow-through for Acceptance", tooltip = "Delay Acceptance labels until a later 15-minute candle confirms continuation beyond the same key level.", group = "Signal Filters")

string acceptanceConfirmationInput = input.string("Either", "Acceptance confirmation type", options = ["Follow-through close", "Same-direction 15m FVG", "Either"], tooltip = "Internal Acceptance filter only: follow-through requires a later close beyond the Acceptance candle's close; the FVG option requires a same-direction 15-minute gap while price remains beyond the level. It never creates a separate FVG label or alert.", group = "Signal Filters")

int responseConfirmWindowInput = input.int(3, "Confirmation window (15m bars)", minval = 1, maxval = 10, tooltip = "Maximum completed 15-minute bars allowed for CISD or Acceptance follow-through confirmation.", group = "Signal Filters")

bool limitAcceptanceToOneMoveInput = input.bool(true, "Limit Acceptance to one per move", tooltip = "Show and alert only the first Acceptance in a directional move. Resets after a close back through its triggering level, a confirmed Reversal, or an opposite-direction Acceptance.", group = "Signal Filters")

bool showResponseLabelsInput = input.bool(true, "Show Acceptance / Reversal labels", tooltip = "Display one selected Acceptance or Reversal label per completed evaluation candle.", group = "Display")

```
string labelSizeInput = input.string("Normal", "Label text size", options = ["Tiny", "Small", "Normal", "Large", "Huge"], tooltip = "Choose the text size for Acceptance and Reversal labels.", group = "Display")

bool enableAlertsInput = input.bool(true, "Enable alert conditions", tooltip = "Enable the Acceptance and Reversal alert conditions.", group = "Alerts")


// --- Session range tracking ---

f_completedSessionRange(string sessionHours) =>
    bool inSession = not na(time(timeframe.period, sessionHours, SESSION_TZ))
    var float buildingHigh = na
    var float buildingLow = na
    var float completedHigh = na
    var float completedLow = na
    if inSession
        if not inSession[1]
            buildingHigh := high
            buildingLow := low
        else
            buildingHigh := math.max(buildingHigh, high)
            buildingLow := math.min(buildingLow, low)
    else if inSession[1]
        completedHigh := buildingHigh
        completedLow := buildingLow
    [completedHigh, completedLow]


f_sessionLevels() =>
    [asiaHigh, asiaLow] = f_completedSessionRange("1800-2359")
```

```
   [londonHigh, londonLow] = f_completedSessionRange("0000-0559")

   [newYorkHigh, newYorkLow] = f_completedSessionRange("0930-1559")

   [asiaIbHigh, asiaIbLow] = f_completedSessionRange("1800-1959")

   [londonIbHigh, londonIbLow] = f_completedSessionRange("0000-0059")

   [newYorkIbHigh, newYorkIbLow] = f_completedSessionRange("0930-1029")

   [asiaHigh, asiaLow, londonHigh, londonLow, newYorkHigh, newYorkLow, asiaIbHigh,
asiaIbLow, londonIbHigh, londonIbLow, newYorkIbHigh, newYorkIbLow]


f_labelSize(string sizeChoice) =>

   sizeChoice == "Tiny" ? size.tiny : sizeChoice == "Small" ? size.small : sizeChoice ==
"Normal" ? size.normal : sizeChoice == "Large" ? size.large : size.huge


// --- Confirmed daily / weekly levels and completed session levels ---

float priorDayHigh = request.security(syminfo.tickerid, "D", high[1], gaps =
barmerge.gaps_off, lookahead = barmerge.lookahead_on)

float priorDayLow = request.security(syminfo.tickerid, "D", low[1], gaps =
barmerge.gaps_off, lookahead = barmerge.lookahead_on)

float priorWeekHigh = request.security(syminfo.tickerid, "W", high[1], gaps =
barmerge.gaps_off, lookahead = barmerge.lookahead_on)

float priorWeekLow = request.security(syminfo.tickerid, "W", low[1], gaps =
barmerge.gaps_off, lookahead = barmerge.lookahead_on)

float pdh = usePriorDayLevelsInput ? priorDayHigh : na

float pdl = usePriorDayLevelsInput ? priorDayLow : na

float pwh = usePriorWeekLevelsInput ? priorWeekHigh : na

float pwl = usePriorWeekLevelsInput ? priorWeekLow : na

float vah = useVahInput and vahInput > 0 ? vahInput : na

float val = useValInput and valInput > 0 ? valInput : na
```

[asiaHighPlot, asiaLowPlot, londonHighPlot, londonLowPlot, newYorkHighPlot, newYorkLowPlot, asiaIbHighPlot, asiaIbLowPlot, londonIbHighPlot, londonIbLowPlot, newYorkIbHighPlot, newYorkIbLowPlot] = request.security(syminfo.tickerid, contextTfInput, f_sessionLevels(), gaps = barmerge.gaps_off, lookahead = barmerge.lookahead_on)

plot(showKeyLevelsInput ? pdh : na, "PDH", color = color.new(BEAR_COLOR, 10), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? pdl : na, "PDL", color = color.new(BULL_COLOR, 10), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? pwh : na, "PWH", color = color.new(BEAR_COLOR, 35), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? pwl : na, "PWL", color = color.new(BULL_COLOR, 35), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? asiaHighPlot : na, "Asia High", color = color.new(#f59e0b, 15), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? asiaLowPlot : na, "Asia Low", color = color.new(#f59e0b, 15), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? londonHighPlot : na, "London High", color = color.new(#5b9cf6, 15), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? londonLowPlot : na, "London Low", color = color.new(#5b9cf6, 15), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? newYorkHighPlot : na, "New York High", color = color.new(#b16ce8, 15), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? newYorkLowPlot : na, "New York Low", color = color.new(#b16ce8, 15), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? asiaIbHighPlot : na, "Asia IB High", color = color.new(#f59e0b, 50), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? asiaIbLowPlot : na, "Asia IB Low", color = color.new(#f59e0b, 50), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? londonIbHighPlot : na, "London IB High", color = color.new(#5b9cf6, 50), linewidth = 1, style = plot.style_linebr)

```
plot(showKeyLevelsInput ? londonIbLowPlot : na, "London IB Low", color = color.new(#5b9cf6, 50), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? newYorkIbHighPlot : na, "New York IB High", color = color.new(#b16ce8, 50), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? newYorkIbLowPlot : na, "New York IB Low", color = color.new(#b16ce8, 50), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? vah : na, "VAH", color = color.new(#f59e0b, 20), linewidth = 1, style = plot.style_linebr)

plot(showKeyLevelsInput ? val : na, "VAL", color = color.new(LEVEL_COLOR, 20), linewidth = 1, style = plot.style_linebr)


// --- Shared key-level array ---

f_buildLevels(float pdhLevel, float pdlLevel, float pwhLevel, float pwlLevel, float vahLevel, float valLevel, float asiaHigh, float asiaLow, float londonHigh, float londonLow, float newYorkHigh, float newYorkLow, float asiaIbHigh, float asiaIbLow, float londonIbHigh, float londonIbLow, float newYorkIbHigh, float newYorkIbLow) =>

   array<float> levels = array.from(pdhLevel, pdlLevel, pwhLevel, pwlLevel, asiaHigh, asiaLow, londonHigh, londonLow, newYorkHigh, newYorkLow, asiaIbHigh, asiaIbLow, londonIbHigh, londonIbLow, newYorkIbHigh, newYorkIbLow, vahLevel, valLevel)

   array<string> ids = array.from("PDH", "PDL", "PWH", "PWL", "Asia High", "Asia Low", "London High", "London Low", "New York High", "New York Low", "Asia IB High", "Asia IB Low", "London IB High", "London IB Low", "New York IB High", "New York IB Low", "VAH", "VAL")

   [levels, ids]


// --- Acceptance / Reversal response logic ---

f_responseAtLevel(float level) =>

   float candleRange = high - low

   int responseCode = 0

   if not na(level) and candleRange > 0
```

```
    float bullCloseStrength = (close - low) / candleRange * 100.0

    float bearCloseStrength = (high - close) / candleRange * 100.0

    float bodyPct = math.abs(close - open) / candleRange * 100.0

    float rangeAbove = (high - math.max(low, level)) / candleRange * 100.0

    float rangeBelow = (math.min(high, level) - low) / candleRange * 100.0

    float reclaimAbove = (close - level) / candleRange * 100.0

    float reclaimBelow = (level - close) / candleRange * 100.0

    float acceptanceCloseThreshold = filterMinorAcceptanceInput ?
math.max(directionalClosePctInput, filteredAcceptanceClosePctInput) :
directionalClosePctInput

    float acceptanceBodyThreshold = filterMinorAcceptanceInput ?
math.max(acceptanceBodyPctInput, filteredAcceptanceBodyPctInput) :
acceptanceBodyPctInput

    float acceptanceBeyondThreshold = filterMinorAcceptanceInput ?
math.max(acceptanceBeyondPctInput, filteredAcceptanceBeyondPctInput) :
acceptanceBeyondPctInput

    float reversalCloseThreshold = filterMinorReversalInput ?
math.max(directionalClosePctInput, filteredReversalClosePctInput) :
directionalClosePctInput

    int reversalSweepThreshold = filterMinorReversalInput ?
math.max(reversalSweepTicksInput, filteredReversalSweepTicksInput) :
reversalSweepTicksInput

    float reversalReclaimThreshold = filterMinorReversalInput ?
math.max(reversalReclaimPctInput, filteredReversalReclaimPctInput) :
reversalReclaimPctInput

    bool bullAcceptance = close[1] <= level and close > level and bullCloseStrength >=
acceptanceCloseThreshold and bodyPct >= acceptanceBodyThreshold and rangeAbove >=
acceptanceBeyondThreshold and close > open

    bool bearAcceptance = close[1] >= level and close < level and bearCloseStrength >=
acceptanceCloseThreshold and bodyPct >= acceptanceBodyThreshold and rangeBelow >=
acceptanceBeyondThreshold and close < open
```

```
    bool bullReversal = close[1] >= level and low <= level - reversalSweepThreshold *
syminfo.mintick and close > level and bullCloseStrength >= reversalCloseThreshold and
reclaimAbove >= reversalReclaimThreshold and close > open

    bool bearReversal = close[1] <= level and high >= level + reversalSweepThreshold *
syminfo.mintick and close < level and bearCloseStrength >= reversalCloseThreshold and
reclaimBelow >= reversalReclaimThreshold and close < open

    responseCode := bullReversal ? 3 : bearReversal ? 4 : bullAcceptance ? 1 :
bearAcceptance ? 2 : 0

  responseCode


f_response15(float pdhLevel, float pdlLevel, float pwhLevel, float pwlLevel, float vahLevel,
float valLevel) =>

  [asiaHigh, asiaLow, londonHigh, londonLow, newYorkHigh, newYorkLow, asiaIbHigh,
asiaIbLow, londonIbHigh, londonIbLow, newYorkIbHigh, newYorkIbLow] = f_sessionLevels()

  [levels, ids] = f_buildLevels(pdhLevel, pdlLevel, pwhLevel, pwlLevel, vahLevel, valLevel,
asiaHigh, asiaLow, londonHigh, londonLow, newYorkHigh, newYorkLow, asiaIbHigh,
asiaIbLow, londonIbHigh, londonIbLow, newYorkIbHigh, newYorkIbLow)

  var array<int> pendingCodes = array.new_int(18, 0)

  var array<int> pendingBars = array.new_int(18, 0)

  var array<float> pendingHighs = array.new_float(18, na)

  var array<float> pendingLows = array.new_float(18, na)

  var array<float> pendingLevels = array.new_float(18, na)

  var array<float> pendingCloses = array.new_float(18, na)

  bool bullAcceptanceAny = false

  bool bearAcceptanceAny = false

  bool bullReversalAny = false

  bool bearReversalAny = false

  int selectedClass = 0

  int selectedDirection = 0
```

```
  string selectedLevelId = ""
  float selectedLevelPrice = na
  float selectedDistance = na
  for i = 0 to array.size(levels) - 1
    float level = array.get(levels, i)
    int resolvedCode = 0
    float resolvedLevelPrice = level
    int pendingCode = array.get(pendingCodes, i)
    if pendingCode != 0
      int pendingAge = bar_index - array.get(pendingBars, i)
      if pendingAge > responseConfirmWindowInput
        array.set(pendingCodes, i, 0)
      else if pendingAge >= 1
        float pendingLevel = array.get(pendingLevels, i)
        float pendingClose = array.get(pendingCloses, i)
        bool pendingBullish = pendingCode == 1 or pendingCode == 3
        bool holdsBeyondLevel = pendingBullish ? close > pendingLevel : close < pendingLevel
        bool followThroughClose = pendingBullish ? close > pendingClose : close < pendingClose
        bool sameDirectionFvg = pendingBullish ? low > high[2] : high < low[2]
        bool acceptanceConfirmed = holdsBeyondLevel and (acceptanceConfirmationInput == "Follow-through close" ? followThroughClose : acceptanceConfirmationInput == "Same-direction 15m FVG" ? sameDirectionFvg : followThroughClose or sameDirectionFvg)
        bool reversalConfirmed = pendingCode == 3 ? close > array.get(pendingHighs, i) : close < array.get(pendingLows, i)
```

```
            bool confirmationMet = pendingCode == 1 or pendingCode == 2 ?
acceptanceConfirmed : reversalConfirmed
            if confirmationMet
              resolvedCode := pendingCode
              resolvedLevelPrice := pendingLevel
              array.set(pendingCodes, i, 0)
        int code = f_responseAtLevel(level)
        if code == 1 or code == 2
          if requireAcceptanceFollowThroughInput
            array.set(pendingCodes, i, code)
            array.set(pendingBars, i, bar_index)
            array.set(pendingHighs, i, high)
            array.set(pendingLows, i, low)
            array.set(pendingLevels, i, level)
            array.set(pendingCloses, i, close)
          else
            resolvedCode := code
        else if code == 3 or code == 4
          if requireReversalCisdInput
            array.set(pendingCodes, i, code)
            array.set(pendingBars, i, bar_index)
            array.set(pendingHighs, i, high)
            array.set(pendingLows, i, low)
            array.set(pendingLevels, i, level)
            array.set(pendingCloses, i, close)
          else
```

```
            resolvedCode := code

        bullAcceptanceAny := bullAcceptanceAny or resolvedCode == 1

        bearAcceptanceAny := bearAcceptanceAny or resolvedCode == 2

        bullReversalAny := bullReversalAny or resolvedCode == 3

        bearReversalAny := bearReversalAny or resolvedCode == 4

        int candidateClass = resolvedCode == 3 or resolvedCode == 4 ? 2 : resolvedCode > 0 ? 1 : 0

        float candidateDistance = candidateClass > 0 ? math.abs(close - resolvedLevelPrice) : na

        if candidateClass > selectedClass or (candidateClass == selectedClass and candidateClass > 0 and candidateDistance < selectedDistance)

            selectedClass := candidateClass

            selectedDirection := resolvedCode == 1 or resolvedCode == 3 ? 1 : -1

            selectedLevelId := array.get(ids, i)

            selectedLevelPrice := resolvedLevelPrice

            selectedDistance := candidateDistance

    [bullAcceptanceAny[1], bearAcceptanceAny[1], bullReversalAny[1], bearReversalAny[1], selectedClass[1], selectedDirection[1], selectedLevelId[1], selectedLevelPrice[1]]


// --- Confirmed evaluation-timeframe events ---

bool chartTfSupported = timeframe.in_seconds() <= timeframe.in_seconds(contextTfInput)

bool newContextPeriod = ta.change(time(contextTfInput)) != 0

[bullAcceptanceRaw, bearAcceptanceRaw, bullReversalRaw, bearReversalRaw, selectedClassRaw, selectedDirectionRaw, selectedLevelIdRaw, selectedLevelPriceRaw] = request.security(syminfo.tickerid, contextTfInput, f_response15(pdh, pdl, pwh, pwl, vah, val), gaps = barmerge.gaps_off, lookahead = barmerge.lookahead_on)

bool responsePeriodEvent = chartTfSupported and newContextPeriod

bool bullReversalEvent = responsePeriodEvent and bullReversalRaw
```

```
bool bearReversalEvent = responsePeriodEvent and bearReversalRaw

bool selectedReversalEvent = responsePeriodEvent and selectedClassRaw == 2

bool rawSelectedAcceptanceEvent = responsePeriodEvent and selectedClassRaw == 1

var int acceptanceLockDirection = 0

var float acceptanceLockLevel = na

if responsePeriodEvent and limitAcceptanceToOneMoveInput and
acceptanceLockDirection != 0

   bool closeReset = acceptanceLockDirection == 1 ? close[1] <= acceptanceLockLevel :
close[1] >= acceptanceLockLevel

   bool oppositeAcceptanceReset = rawSelectedAcceptanceEvent and
selectedDirectionRaw != acceptanceLockDirection

   bool reversalReset = bullReversalEvent or bearReversalEvent

   if closeReset or oppositeAcceptanceReset or reversalReset

      acceptanceLockDirection := 0

      acceptanceLockLevel := na

bool selectedAcceptanceEvent = rawSelectedAcceptanceEvent and (not
limitAcceptanceToOneMoveInput or acceptanceLockDirection == 0)

if selectedAcceptanceEvent and limitAcceptanceToOneMoveInput

   acceptanceLockDirection := selectedDirectionRaw

   acceptanceLockLevel := selectedLevelPriceRaw

bool bullAcceptanceEvent = selectedAcceptanceEvent and selectedDirectionRaw == 1

bool bearAcceptanceEvent = selectedAcceptanceEvent and selectedDirectionRaw == -1

bool selectedResponseEvent = selectedReversalEvent or selectedAcceptanceEvent

bool anyResponseEvent = bullAcceptanceEvent or bearAcceptanceEvent or
bullReversalEvent or bearReversalEvent


// --- Acceptance / Reversal labels ---
```

```
if showResponseLabelsInput and selectedResponseEvent
    string responseLabelText = selectedClassRaw == 2 ? "Reversal" : "Acceptance"
    bool responseBullish = selectedDirectionRaw == 1
    label.new(bar_index, responseBullish ? low : high, responseLabelText, style = responseBullish ? label.style_label_up : label.style_label_down, color = color.new(responseBullish ? BULL_COLOR : BEAR_COLOR, 5), textcolor = color.white, size = f_labelSize(labelSizeInput))

// --- Acceptance / Reversal alert conditions ---
alertcondition(enableAlertsInput and bullAcceptanceEvent, "Bullish ACCEPTANCE", "Bullish ACCEPTANCE on {{ticker}} (15-minute evaluation).")
alertcondition(enableAlertsInput and bearAcceptanceEvent, "Bearish ACCEPTANCE", "Bearish ACCEPTANCE on {{ticker}} (15-minute evaluation).")
alertcondition(enableAlertsInput and bullReversalEvent, "Bullish REVERSAL", "Bullish REVERSAL on {{ticker}} (15-minute evaluation).")
alertcondition(enableAlertsInput and bearReversalEvent, "Bearish REVERSAL", "Bearish REVERSAL on {{ticker}} (15-minute evaluation).")
alertcondition(enableAlertsInput and anyResponseEvent, "15m Key Level Response", "A confirmed 15-minute key-level Acceptance or Reversal occurred on {{ticker}}. Use the script's alert() call for the selected classification, direction, level identifier, and price.")

if enableAlertsInput and selectedResponseEvent
    string responseName = selectedClassRaw == 2 ? "Reversal" : "Acceptance"
    string directionName = selectedDirectionRaw == 1 ? "Bullish" : "Bearish"
    string responseMessage = syminfo.ticker + " | " + contextTfInput + " | " + directionName + " " + responseName + " | " + selectedLevelIdRaw + " @ " + str.tostring(selectedLevelPriceRaw, format.mintick)
    alert(responseMessage, alert.freq_once_per_bar)
```