```
// 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 - Three 5s Key-Level Story", "MFB Three 5s", overlay = true, max_bars_back = 5000, max_labels_count = 100)


// --- Constants ---

const string TIMEZONE = "America/New_York"

const string RTH_SESSION = "0930-1600:23456"

const string GLOBEX_SESSION = "1800-0930:23456"

const int COMPOSITE_PROFILE_DAYS = 5


// --- Inputs ---

int nyProfileTicksInput = input.int(4, "NY Checklist profile ticks per row", minval = 1, maxval = 100, group = "Reference Levels", tooltip = "Tick-based row size used to reproduce the NY Checklist prior-RTH and Globex profiles.")

int vrsRowsInput = input.int(50, "VRS profile rows", minval = 10, maxval = 100, group = "Reference Levels", tooltip = "Number of rows used to reproduce the Value Reading and five-day composite profiles.")

float valueAreaPercentInput = input.float(70.0, "Value area percentage", minval = 50.0, maxval = 90.0, step = 1.0, group = "Reference Levels", tooltip = "Volume percentage used for value-area boundaries in the reproduced profiles.")

float compositeHvnThresholdInput = input.float(25.0, "5-day HVN threshold (% of POC row)", minval = 5.0, maxval = 90.0, step = 5.0, group = "Reference Levels", tooltip = "Minimum five-day composite local-peak volume as a share of the composite POC row.")
```

int closesRequiredInput = input.int(3, "Consecutive closes required", minval = 2, maxval = 10, group = "Three 5s", tooltip = "Number of confirmed 5-minute closes beyond the balance-zone edge needed for acceptance. After a directional attempt, the same number of closes beyond the opposite edge marks a failed attempt.")

int touchProximityTicksInput = input.int(4, "Level touch proximity (ticks)", minval = 0, maxval = 1000, group = "Three 5s", tooltip = "A completed 5-minute candle starts or updates a sequence when its range comes within this many ticks of a reference.")

int balancedZoneTicksInput = input.int(8, "Balanced close zone (ticks)", minval = 1, maxval = 1000, group = "Three 5s", tooltip = "Closes within this distance on either side of a reference count toward balance. Returning to this zone resets a directional streak and starts a new balance count.")

int sequenceTimeoutBarsInput = input.int(12, "Sequence timeout (5-minute bars)", minval = 3, maxval = 100, group = "Three 5s", tooltip = "Expire an unfinished sequence after this many confirmed 5-minute candles.")

// --- Display Inputs ---

bool showDashboardInput = input.bool(true, "Show sequence dashboard", group = "Display", tooltip = "Show the active reference, close-by-close count, and latest completed reading.")

bool showProgressLabelsInput = input.bool(false, "Show progress labels", group = "Display", tooltip = "Show intermediate progress labels such as 1/3 and 2/3. Completed outcome labels remain visible.")

string dashboardPositionInput = input.string("Top Right", "Dashboard position", options = ["Top Right", "Bottom Right"], group = "Display", tooltip = "Right-side corner used for the compact Three 5s status panel.")

```
color bullColorInput = input.color(#089981, "Bullish / accepted color", group = "Style", tooltip = "Color for closes progressing above a reference and accepted outcomes.")

color bearColorInput = input.color(#f23645, "Bearish / failed color", group = "Style", tooltip = "Color for closes progressing below a reference and failed-auction outcomes.")

color neutralColorInput = input.color(#5b9cf6, "Neutral / balanced color", group = "Style", tooltip = "Color for balanced, waiting, and unavailable readings.")


// --- Profile Data Types ---

type SessionProfile

    float sessionHigh

    float sessionLow

    float pocPrice

    float vahPrice

    float valPrice

    float[] sourcePrices

    float[] sourceVolumes


// --- Profile Functions ---

f_mapProfileLevels(map<int, float> profileMap, float totalVolume, float rowSize, float valuePercent) =>

    float pocPrice = na

    float vahPrice = na
```

```
    float valPrice = na
    int rowCount = map.size(profileMap)
    if rowCount > 0 and totalVolume > 0 and rowSize > 0
        array<int> keys = map.keys(profileMap)
        array.sort(keys, order.ascending)
        int pocIndex = 0
        float maximumVolume = -1.0
        for index = 0 to rowCount - 1
            float rowVolume = nz(map.get(profileMap, array.get(keys, index)), 0.0)
            if rowVolume > maximumVolume
                maximumVolume := rowVolume
                pocIndex := index
        int lowerIndex = pocIndex
        int upperIndex = pocIndex
        float accumulatedVolume = nz(map.get(profileMap, array.get(keys,
pocIndex)), 0.0)
        float targetVolume = totalVolume * valuePercent
        for step = 0 to rowCount - 1
            if accumulatedVolume < targetVolume
                bool canExpandLower = lowerIndex > 0
                bool canExpandUpper = upperIndex < rowCount - 1
                if canExpandLower and canExpandUpper
                    float lowerVolume = nz(map.get(profileMap, array.get(keys,
lowerIndex - 1)), 0.0)
```

```
            float upperVolume = nz(map.get(profileMap, array.get(keys,
upperIndex + 1)), 0.0)
            if lowerVolume >= upperVolume
              lowerIndex -= 1
              accumulatedVolume += lowerVolume
            else
              upperIndex += 1
              accumulatedVolume += upperVolume
          else if canExpandLower
            lowerIndex -= 1
            accumulatedVolume += nz(map.get(profileMap, array.get(keys,
lowerIndex)), 0.0)
          else if canExpandUpper
            upperIndex += 1
            accumulatedVolume += nz(map.get(profileMap, array.get(keys,
upperIndex)), 0.0)
      pocPrice := (float(array.get(keys, pocIndex)) + 0.5) * rowSize
      valPrice := float(array.get(keys, lowerIndex)) * rowSize
      vahPrice := (float(array.get(keys, upperIndex)) + 1.0) * rowSize
    [pocPrice, vahPrice, valPrice]


f_arrayProfileLevels(float[] prices, float[] volumes, float sessionHigh, float
sessionLow, int rows, float valuePercent) =>
    float pocPrice = na
```

```
    float vahPrice = na
    float valPrice = na
    float priceRange = sessionHigh - sessionLow
    int sampleCount = array.size(prices)
    if priceRange > 0 and sampleCount > 0 and rows > 0
        float rowStep = priceRange / rows
        float[] rowVolumes = array.new_float(rows, 0.0)
        float totalVolume = 0.0
        for sampleIndex = 0 to sampleCount - 1
            float samplePrice = array.get(prices, sampleIndex)
            float sampleVolume = nz(array.get(volumes, sampleIndex), 0.0)
            int rowIndex = int(math.floor((samplePrice - sessionLow) / rowStep))
            rowIndex := math.max(0, math.min(rows - 1, rowIndex))
            array.set(rowVolumes, rowIndex, array.get(rowVolumes, rowIndex) +
sampleVolume)
            totalVolume += sampleVolume
        float maximumVolume = array.get(rowVolumes, 0)
        int pocIndex = 0
        for rowSearch = 1 to rows - 1
            float candidateVolume = array.get(rowVolumes, rowSearch)
            if candidateVolume > maximumVolume
                maximumVolume := candidateVolume
                pocIndex := rowSearch
```

```
    if maximumVolume > 0 and totalVolume > 0
        int lowerIndex = pocIndex
        int upperIndex = pocIndex
        float accumulatedVolume = maximumVolume
        float targetVolume = totalVolume * valuePercent
        int expansionSteps = 0
        while accumulatedVolume < targetVolume and (upperIndex < rows - 1 or lowerIndex > 0) and expansionSteps < rows
            expansionSteps += 1
            bool canExpandUpper = upperIndex < rows - 1
            bool canExpandLower = lowerIndex > 0
            float upperVolume = canExpandUpper ? array.get(rowVolumes, upperIndex + 1) : 0.0
            float lowerVolume = canExpandLower ? array.get(rowVolumes, lowerIndex - 1) : 0.0
            if canExpandUpper and (not canExpandLower or upperVolume >= lowerVolume)
                upperIndex += 1
                accumulatedVolume += upperVolume
            else if canExpandLower
                lowerIndex -= 1
                accumulatedVolume += lowerVolume
        pocPrice := sessionLow + (pocIndex + 0.5) * rowStep
        valPrice := sessionLow + lowerIndex * rowStep
```

```
        vahPrice := sessionLow + (upperIndex + 1) * rowStep
    [pocPrice, vahPrice, valPrice]


f_compositeProfile(array<SessionProfile> profiles, int rows, float
valuePercent, float peakPercent) =>
    float pocPrice = na
    float vahPrice = na
    float valPrice = na
    float peak1Price = na
    float peak2Price = na
    float peak3Price = na
    if array.size(profiles) >= COMPOSITE_PROFILE_DAYS
        float compositeHigh = na
        float compositeLow = na
        for profileIndex = 0 to COMPOSITE_PROFILE_DAYS - 1
            SessionProfile sourceProfile = array.get(profiles, profileIndex)
            compositeHigh := na(compositeHigh) ? sourceProfile.sessionHigh :
math.max(compositeHigh, sourceProfile.sessionHigh)
            compositeLow := na(compositeLow) ? sourceProfile.sessionLow :
math.min(compositeLow, sourceProfile.sessionLow)
        float compositeRange = compositeHigh - compositeLow
        if compositeRange > 0
            float rowStep = compositeRange / rows
            float[] rowVolumes = array.new_float(rows, 0.0)
```

```
        float totalVolume = 0.0
        for profileIndex = 0 to COMPOSITE_PROFILE_DAYS - 1
          SessionProfile sourceProfile = array.get(profiles, profileIndex)
          int sourceCount = array.size(sourceProfile.sourcePrices)
          if sourceCount > 0
            for sourceIndex = 0 to sourceCount - 1
              float sourcePrice = array.get(sourceProfile.sourcePrices,
sourceIndex)
              float sourceVolume = nz(array.get(sourceProfile.sourceVolumes,
sourceIndex), 0.0)
              int rowIndex = int(math.floor((sourcePrice - compositeLow) /
rowStep))
              rowIndex := math.max(0, math.min(rows - 1, rowIndex))
              array.set(rowVolumes, rowIndex, array.get(rowVolumes, rowIndex)
+ sourceVolume)
              totalVolume += sourceVolume
        float maximumVolume = array.get(rowVolumes, 0)
        int pocIndex = 0
        for rowSearch = 1 to rows - 1
          float candidateVolume = array.get(rowVolumes, rowSearch)
          if candidateVolume > maximumVolume
            maximumVolume := candidateVolume
            pocIndex := rowSearch
        if maximumVolume > 0 and totalVolume > 0
```

```
        int lowerIndex = pocIndex
        int upperIndex = pocIndex
        float accumulatedVolume = maximumVolume
        float targetVolume = totalVolume * valuePercent
        int expansionSteps = 0
        while accumulatedVolume < targetVolume and (upperIndex < rows - 1 or lowerIndex > 0) and expansionSteps < rows
            expansionSteps += 1
            bool canExpandUpper = upperIndex < rows - 1
            bool canExpandLower = lowerIndex > 0
            float upperVolume = canExpandUpper ? array.get(rowVolumes, upperIndex + 1) : 0.0
            float lowerVolume = canExpandLower ? array.get(rowVolumes, lowerIndex - 1) : 0.0
            if canExpandUpper and (not canExpandLower or upperVolume >= lowerVolume)
                upperIndex += 1
                accumulatedVolume += upperVolume
            else if canExpandLower
                lowerIndex -= 1
                accumulatedVolume += lowerVolume
        pocPrice := compositeLow + (pocIndex + 0.5) * rowStep
        valPrice := compositeLow + lowerIndex * rowStep
        vahPrice := compositeLow + (upperIndex + 1) * rowStep
```

```
    float minimumPeakVolume = maximumVolume * peakPercent
    int minimumPeakSpacing = math.max(2, int(math.round(rows * 0.06)))
    int firstPeakRow = -1
    float firstPeakVolume = 0.0
    for peakCandidate1 = 0 to rows - 1
      float candidateVolume1 = array.get(rowVolumes, peakCandidate1)
      float leftVolume1 = peakCandidate1 > 0 ? array.get(rowVolumes, peakCandidate1 - 1) : -1.0
      float rightVolume1 = peakCandidate1 < rows - 1 ? array.get(rowVolumes, peakCandidate1 + 1) : -1.0
      bool isPeak1 = candidateVolume1 >= minimumPeakVolume and candidateVolume1 >= leftVolume1 and candidateVolume1 >= rightVolume1 and (candidateVolume1 > leftVolume1 or candidateVolume1 > rightVolume1)
      if isPeak1 and candidateVolume1 > firstPeakVolume
        firstPeakVolume := candidateVolume1
        firstPeakRow := peakCandidate1
    int secondPeakRow = -1
    float secondPeakVolume = 0.0
    for peakCandidate2 = 0 to rows - 1
      float candidateVolume2 = array.get(rowVolumes, peakCandidate2)
      float leftVolume2 = peakCandidate2 > 0 ? array.get(rowVolumes, peakCandidate2 - 1) : -1.0
      float rightVolume2 = peakCandidate2 < rows - 1 ? array.get(rowVolumes, peakCandidate2 + 1) : -1.0
```

```
            bool isPeak2 = candidateVolume2 >= minimumPeakVolume and
candidateVolume2 >= leftVolume2 and candidateVolume2 >= rightVolume2
and (candidateVolume2 > leftVolume2 or candidateVolume2 > rightVolume2)

            bool spacedFromFirst = firstPeakRow < 0 or
math.abs(peakCandidate2 - firstPeakRow) >= minimumPeakSpacing

            if isPeak2 and spacedFromFirst and candidateVolume2 >
secondPeakVolume

                secondPeakVolume := candidateVolume2

                secondPeakRow := peakCandidate2

        int thirdPeakRow = -1

        float thirdPeakVolume = 0.0

        for peakCandidate3 = 0 to rows - 1

            float candidateVolume3 = array.get(rowVolumes, peakCandidate3)

            float leftVolume3 = peakCandidate3 > 0 ? array.get(rowVolumes,
peakCandidate3 - 1) : -1.0

            float rightVolume3 = peakCandidate3 < rows - 1 ?
array.get(rowVolumes, peakCandidate3 + 1) : -1.0

            bool isPeak3 = candidateVolume3 >= minimumPeakVolume and
candidateVolume3 >= leftVolume3 and candidateVolume3 >= rightVolume3
and (candidateVolume3 > leftVolume3 or candidateVolume3 > rightVolume3)

            bool spacedFromFirst3 = firstPeakRow < 0 or
math.abs(peakCandidate3 - firstPeakRow) >= minimumPeakSpacing

            bool spacedFromSecond3 = secondPeakRow < 0 or
math.abs(peakCandidate3 - secondPeakRow) >= minimumPeakSpacing

            if isPeak3 and spacedFromFirst3 and spacedFromSecond3 and
candidateVolume3 > thirdPeakVolume
```

```
            thirdPeakVolume := candidateVolume3
            thirdPeakRow := peakCandidate3
        peak1Price := firstPeakRow >= 0 ? compositeLow + (firstPeakRow + 0.5) * rowStep : na
        peak2Price := secondPeakRow >= 0 ? compositeLow + (secondPeakRow + 0.5) * rowStep : na
        peak3Price := thirdPeakRow >= 0 ? compositeLow + (thirdPeakRow + 0.5) * rowStep : na
    [pocPrice, vahPrice, valPrice, peak1Price, peak2Price, peak3Price]

// --- Session Detection ---
bool inRth = not na(time(timeframe.period, RTH_SESSION, TIMEZONE))
bool inGlobex = not na(time(timeframe.period, GLOBEX_SESSION, TIMEZONE))
bool isRthOpen = inRth and (bar_index == 0 or not inRth[1])
bool isRthClose = not inRth and bar_index > 0 and inRth[1]
bool isGlobexOpen = inGlobex and (bar_index == 0 or not inGlobex[1])
bool isGlobexClose = not inGlobex and bar_index > 0 and inGlobex[1]

// --- Profile Collection State ---
var map<int, float> rthTickProfile = map.new<int, float>()
var map<int, float> globexTickProfile = map.new<int, float>()
var float rthTickVolume = 0.0
var float globexTickVolume = 0.0
```

```
var float rthHigh = na
var float rthLow = na
var float globexHigh = na
var float globexLow = na
var float[] rthPrices = array.new_float()
var float[] rthVolumes = array.new_float()
var array<SessionProfile> completedRthProfiles =
array.new<SessionProfile>()
var float checklistPoc = na
var float checklistVah = na
var float checklistVal = na
var float vrsPoc = na
var float vrsVah = na
var float vrsVal = na
var float globexPoc = na
var float compositePoc = na
var float compositeVah = na
var float compositeVal = na
var float compositeHvn1 = na
var float compositeHvn2 = na
var float compositeHvn3 = na
float nyRowSize = syminfo.mintick * nyProfileTicksInput
```

```
if isRthOpen
    map.clear(rthTickProfile)
    rthTickVolume := 0.0
    array.clear(rthPrices)
    array.clear(rthVolumes)
    rthHigh := high
    rthLow := low
if inRth
    rthHigh := math.max(nz(rthHigh, high), high)
    rthLow := math.min(nz(rthLow, low), low)
    float currentVolume = nz(volume, 0.0)
    int profileBucket = int(math.floor(hlc3 / nyRowSize))
    map.put(rthTickProfile, profileBucket, nz(map.get(rthTickProfile,
profileBucket), 0.0) + currentVolume)
    rthTickVolume += currentVolume
    array.push(rthPrices, hlc3)
    array.push(rthVolumes, currentVolume)
if isRthClose
    [checklistPocResult, checklistVahResult, checklistValResult] =
f_mapProfileLevels(rthTickProfile, rthTickVolume, nyRowSize,
valueAreaPercentInput * 0.01)
    checklistPoc := checklistPocResult
    checklistVah := checklistVahResult
    checklistVal := checklistValResult
```

```
    [vrsPocResult, vrsVahResult, vrsValResult] = f_arrayProfileLevels(rthPrices,
rthVolumes, rthHigh, rthLow, vrsRowsInput, valueAreaPercentInput * 0.01)
    vrsPoc := vrsPocResult
    vrsVah := vrsVahResult
    vrsVal := vrsValResult
    SessionProfile newProfile = SessionProfile.new(rthHigh, rthLow,
vrsPocResult, vrsVahResult, vrsValResult, array.copy(rthPrices),
array.copy(rthVolumes))
    if not na(vrsPocResult)
        array.unshift(completedRthProfiles, newProfile)
        if array.size(completedRthProfiles) > COMPOSITE_PROFILE_DAYS
            array.pop(completedRthProfiles)
        [compositePocResult, compositeVahResult, compositeValResult,
compositeHvn1Result, compositeHvn2Result, compositeHvn3Result] =
f_compositeProfile(completedRthProfiles, vrsRowsInput,
valueAreaPercentInput * 0.01, compositeHvnThresholdInput * 0.01)
        compositePoc := compositePocResult
        compositeVah := compositeVahResult
        compositeVal := compositeValResult
        compositeHvn1 := compositeHvn1Result
        compositeHvn2 := compositeHvn2Result
        compositeHvn3 := compositeHvn3Result


if isGlobexOpen
    map.clear(globexTickProfile)
```

```
    globexTickVolume := 0.0
    globexHigh := high
    globexLow := low
if inGlobex
    globexHigh := math.max(nz(globexHigh, high), high)
    globexLow := math.min(nz(globexLow, low), low)
    float currentGlobexVolume = nz(volume, 0.0)
    int globexBucket = int(math.floor(hlc3 / nyRowSize))
    map.put(globexTickProfile, globexBucket, nz(map.get(globexTickProfile,
globexBucket), 0.0) + currentGlobexVolume)
    globexTickVolume += currentGlobexVolume
if isGlobexClose
    [globexPocResult, globexVahResult, globexValResult] =
f_mapProfileLevels(globexTickProfile, globexTickVolume, nyRowSize,
valueAreaPercentInput * 0.01)
    globexPoc := globexPocResult


// --- Confirmed 5-minute data ---
int chartSeconds = timeframe.in_seconds()
bool chartSupportsFiveMinute = not na(chartSeconds) and chartSeconds <=
300
int fiveMinuteBucketTime = time("5")
bool newFiveMinuteBucket = not na(fiveMinuteBucketTime) and
ta.change(fiveMinuteBucketTime) != 0
```

```
[fiveMinuteClosedTime, fiveMinuteOpen, fiveMinuteHigh, fiveMinuteLow,
fiveMinuteClose, fiveMinutePriorClose] = request.security(syminfo.tickerid,
"5", [time[1], open[1], high[1], low[1], close[1], close[2]], gaps =
barmerge.gaps_off, lookahead = barmerge.lookahead_on)


// --- Three 5s State ---

var bool sequenceActive = false

var float activeLevel = na

var string activeLevelName = "NO ACTIVE LEVEL"

var int sequenceDirection = 0

var int beyondCloseCount = 0

var int insideCloseCount = 0

var int balancedCloseCount = 0

var int barsInSequence = 0

var string latestStory = "WAITING FOR A REFERENCE TOUCH"

var string latestStoryDetail = "Confirmed 5-minute closes only."

bool acceptanceCompletedNow = false

bool failedAuctionCompletedNow = false

bool balancedCompletedNow = false

bool levelSequenceStartedNow = false

string progressLabelText = ""

float progressLabelPrice = na

color progressLabelColor = neutralColorInput
```

```
if chartSupportsFiveMinute and newFiveMinuteBucket and not
na(fiveMinuteClosedTime) and not na(fiveMinuteClose)

   float touchDistance = syminfo.mintick * touchProximityTicksInput

   float balanceDistance = syminfo.mintick * balancedZoneTicksInput

   float[] candidatePrices = array.from(checklistPoc, checklistVah,
checklistVal, vrsPoc, vrsVah, vrsVal, (not na(vrsVah) and not na(vrsVal)) ?
(vrsVah + vrsVal) * 0.5 : na, globexPoc, globexHigh, globexLow, compositePoc,
compositeVah, compositeVal, compositeHvn1, compositeHvn2,
compositeHvn3)

   string[] candidateNames = array.from("NY D-1 POC", "NY D-1 VAH", "NY D-1
VAL", "VRS D-1 POC", "VRS D-1 VAH", "VRS D-1 VAL", "VRS D-1 VA MID",
"GLOBEX POC", "GLOBEX HIGH", "GLOBEX LOW", "5D COMP POC", "5D COMP
VAH", "5D COMP VAL", "5D HVN 1", "5D HVN 2", "5D HVN 3")

   float nearestTouchedLevel = na

   string nearestTouchedName = "NO LEVEL"

   float nearestTouchedDistance = na

   for candidateIndex = 0 to array.size(candidatePrices) - 1

      float candidatePrice = array.get(candidatePrices, candidateIndex)

      bool candidateTouched = not na(candidatePrice) and fiveMinuteHigh >=
candidatePrice - touchDistance and fiveMinuteLow <= candidatePrice +
touchDistance

      if candidateTouched

         float candidateDistance = math.abs(fiveMinuteClose - candidatePrice)

         if na(nearestTouchedDistance) or candidateDistance <
nearestTouchedDistance - syminfo.mintick * 0.5

            nearestTouchedLevel := candidatePrice
```

```
            nearestTouchedName := array.get(candidateNames, candidateIndex)
            nearestTouchedDistance := candidateDistance
        else if math.abs(candidateDistance - nearestTouchedDistance) <= syminfo.mintick * 0.5 and not str.contains(nearestTouchedName, array.get(candidateNames, candidateIndex))
            nearestTouchedName += " / " + array.get(candidateNames, candidateIndex)


    bool candidateIsDifferent = not na(nearestTouchedLevel) and (na(activeLevel) or math.abs(nearestTouchedLevel - activeLevel) > syminfo.mintick * 0.5)
    bool startNewSequence = not na(nearestTouchedLevel) and (not sequenceActive or candidateIsDifferent)
    if startNewSequence
        sequenceActive := true
        activeLevel := nearestTouchedLevel
        activeLevelName := nearestTouchedName
        sequenceDirection := 0
        beyondCloseCount := 0
        insideCloseCount := 0
        balancedCloseCount := 0
        barsInSequence := 1
        levelSequenceStartedNow := true
        latestStory := "TESTING " + activeLevelName
        latestStoryDetail := "First confirmed 5-minute close is being classified."
```

```
    if math.abs(fiveMinuteClose - activeLevel) <= balanceDistance
        sequenceDirection := 0
        beyondCloseCount := 0
        insideCloseCount := 0
        balancedCloseCount := 1
        latestStory := "BALANCED 1/" + str.tostring(closesRequiredInput) + " | " + activeLevelName
        latestStoryDetail := "Close is inside the configured balance zone around the reference."
        progressLabelText := "1/" + str.tostring(closesRequiredInput)
        progressLabelPrice := activeLevel
        progressLabelColor := neutralColorInput
    else if fiveMinuteClose > activeLevel + balanceDistance
        sequenceDirection := 1
        beyondCloseCount := 1
        latestStory := "ABOVE 1/" + str.tostring(closesRequiredInput) + " | " + activeLevelName
        latestStoryDetail := "First close outside the upper edge of the balance zone."
        progressLabelText := "1/" + str.tostring(closesRequiredInput)
        progressLabelPrice := activeLevel
        progressLabelColor := bullColorInput
    else
        sequenceDirection := -1
```

```
        beyondCloseCount := 1
        latestStory := "BELOW 1/" + str.tostring(closesRequiredInput) + " | " + activeLevelName
        latestStoryDetail := "First close outside the lower edge of the balance zone."
        progressLabelText := "1/" + str.tostring(closesRequiredInput)
        progressLabelPrice := activeLevel
        progressLabelColor := bearColorInput
    else if sequenceActive
      barsInSequence += 1
      if barsInSequence > sequenceTimeoutBarsInput
        sequenceActive := false
        latestStory := "SEQUENCE EXPIRED | " + activeLevelName
        latestStoryDetail := "No three-close outcome within " + str.tostring(sequenceTimeoutBarsInput) + " confirmed 5-minute bars."
      else if sequenceDirection == 1
        if fiveMinuteClose > activeLevel + balanceDistance
          beyondCloseCount += 1
          insideCloseCount := 0
          balancedCloseCount := 0
          latestStory := "ABOVE " + str.tostring(math.min(beyondCloseCount, closesRequiredInput)) + "/" + str.tostring(closesRequiredInput) + " | " + activeLevelName
```

```
        latestStoryDetail := "Close " +
str.tostring(math.min(beyondCloseCount, closesRequiredInput)) + " of " +
str.tostring(closesRequiredInput) + " outside the upper balance band."

        progressLabelText := str.tostring(math.min(beyondCloseCount,
closesRequiredInput)) + "/" + str.tostring(closesRequiredInput)

        progressLabelPrice := activeLevel

        progressLabelColor := bullColorInput

        if beyondCloseCount >= closesRequiredInput

          acceptanceCompletedNow := true

          sequenceActive := false

          latestStory := "ACCEPTED ABOVE | " + activeLevelName

          latestStoryDetail := str.tostring(closesRequiredInput) + " consecutive
confirmed 5-minute closes above the balance band."

      else if fiveMinuteClose < activeLevel - balanceDistance

        insideCloseCount += 1

        beyondCloseCount := 0

        balancedCloseCount := 0

        latestStory := "BACK BELOW BAND " +
str.tostring(math.min(insideCloseCount, closesRequiredInput)) + "/" +
str.tostring(closesRequiredInput) + " | " + activeLevelName

        latestStoryDetail := "Close moved beyond the lower balance-band
edge after an above-level attempt."

        progressLabelText := str.tostring(math.min(insideCloseCount,
closesRequiredInput)) + "/" + str.tostring(closesRequiredInput)

        progressLabelPrice := activeLevel

        progressLabelColor := neutralColorInput
```

```
        if insideCloseCount >= closesRequiredInput
            failedAuctionCompletedNow := true
            sequenceActive := false
            latestStory := "FAILED ABOVE-LEVEL ATTEMPT | " + activeLevelName
            latestStoryDetail := str.tostring(closesRequiredInput) + " consecutive confirmed 5-minute closes below the balance band."
        else
            sequenceDirection := 0
            beyondCloseCount := 0
            insideCloseCount := 0
            balancedCloseCount := 1
            latestStory := "BALANCED 1/" + str.tostring(closesRequiredInput) + " | " + activeLevelName
            latestStoryDetail := "Close returned inside the configured balance band; directional streak reset."
            progressLabelText := "1/" + str.tostring(closesRequiredInput)
            progressLabelPrice := activeLevel
            progressLabelColor := neutralColorInput
    else if sequenceDirection == -1
        if fiveMinuteClose < activeLevel - balanceDistance
            beyondCloseCount += 1
            insideCloseCount := 0
            balancedCloseCount := 0
```

```
        latestStory := "BELOW " + str.tostring(math.min(beyondCloseCount,
closesRequiredInput)) + "/" + str.tostring(closesRequiredInput) + " | " +
activeLevelName

        latestStoryDetail := "Close " +
str.tostring(math.min(beyondCloseCount, closesRequiredInput)) + " of " +
str.tostring(closesRequiredInput) + " outside the lower balance band."

        progressLabelText := str.tostring(math.min(beyondCloseCount,
closesRequiredInput)) + "/" + str.tostring(closesRequiredInput)

        progressLabelPrice := activeLevel

        progressLabelColor := bearColorInput

        if beyondCloseCount >= closesRequiredInput

          acceptanceCompletedNow := true

          sequenceActive := false

          latestStory := "ACCEPTED BELOW | " + activeLevelName

          latestStoryDetail := str.tostring(closesRequiredInput) + " consecutive
confirmed 5-minute closes below the balance band."

      else if fiveMinuteClose > activeLevel + balanceDistance

        insideCloseCount += 1

        beyondCloseCount := 0

        balancedCloseCount := 0

        latestStory := "BACK ABOVE BAND " +
str.tostring(math.min(insideCloseCount, closesRequiredInput)) + "/" +
str.tostring(closesRequiredInput) + " | " + activeLevelName

        latestStoryDetail := "Close moved beyond the upper balance-band
edge after a below-level attempt."
```

```
            progressLabelText := str.tostring(math.min(insideCloseCount,
closesRequiredInput)) + "/" + str.tostring(closesRequiredInput)
            progressLabelPrice := activeLevel
            progressLabelColor := neutralColorInput
            if insideCloseCount >= closesRequiredInput
              failedAuctionCompletedNow := true
              sequenceActive := false
              latestStory := "FAILED BELOW-LEVEL ATTEMPT | " + activeLevelName
              latestStoryDetail := str.tostring(closesRequiredInput) + " consecutive
confirmed 5-minute closes above the balance band."
          else
            sequenceDirection := 0
            beyondCloseCount := 0
            insideCloseCount := 0
            balancedCloseCount := 1
            latestStory := "BALANCED 1/" + str.tostring(closesRequiredInput) + " | "
+ activeLevelName
            latestStoryDetail := "Close returned inside the configured balance
band; directional streak reset."
            progressLabelText := "1/" + str.tostring(closesRequiredInput)
            progressLabelPrice := activeLevel
            progressLabelColor := neutralColorInput
        else
          if fiveMinuteClose > activeLevel + balanceDistance
```

```
            sequenceDirection := 1
            beyondCloseCount := 1
            balancedCloseCount := 0
            insideCloseCount := 0
            latestStory := "ABOVE 1/" + str.tostring(closesRequiredInput) + " | " +
activeLevelName
            latestStoryDetail := "First close outside the upper balance band;
acceptance count started."
            progressLabelText := "1/" + str.tostring(closesRequiredInput)
            progressLabelPrice := activeLevel
            progressLabelColor := bullColorInput
        else if fiveMinuteClose < activeLevel - balanceDistance
            sequenceDirection := -1
            beyondCloseCount := 1
            balancedCloseCount := 0
            insideCloseCount := 0
            latestStory := "BELOW 1/" + str.tostring(closesRequiredInput) + " | " +
activeLevelName
            latestStoryDetail := "First close outside the lower balance band;
acceptance count started."
            progressLabelText := "1/" + str.tostring(closesRequiredInput)
            progressLabelPrice := activeLevel
            progressLabelColor := bearColorInput
        else
```

```
        balancedCloseCount += 1

        latestStory := "BALANCED " +
str.tostring(math.min(balancedCloseCount, closesRequiredInput)) + "/" +
str.tostring(closesRequiredInput) + " | " + activeLevelName

        latestStoryDetail := "Consecutive 5-minute closes are sitting inside
the configured balance band."

        progressLabelText := str.tostring(math.min(balancedCloseCount,
closesRequiredInput)) + "/" + str.tostring(closesRequiredInput)

        progressLabelPrice := activeLevel

        progressLabelColor := neutralColorInput

        if balancedCloseCount >= closesRequiredInput

          balancedCompletedNow := true

          sequenceActive := false

          latestStory := "BALANCED AT | " + activeLevelName

          latestStoryDetail := str.tostring(closesRequiredInput) + " consecutive
confirmed 5-minute closes within the balance band."


// --- Progress Display ---

f_dashboardPosition(string positionText) =>

  positionText == "Top Left" ? position.top_left : positionText == "Bottom Right"
? position.bottom_right : positionText == "Bottom Left" ? position.bottom_left :
position.top_right


var table storyTable =
table.new(f_dashboardPosition(dashboardPositionInput), 2, 6, bgcolor =
```

color.new(#161616, 0), frame_color = #2E2E2E, frame_width = 1, border_color = #2E2E2E, border_width = 1)

if showProgressLabelsInput and progressLabelText != "" and not na(fiveMinuteClosedTime)

    float candleLabelPrice = sequenceDirection == 1 ? fiveMinuteLow : fiveMinuteHigh

    string candleLabelStyle = sequenceDirection == 1 ? label.style_label_up : label.style_label_down

    label.new(fiveMinuteClosedTime, candleLabelPrice, progressLabelText, xloc = xloc.bar_time, yloc = yloc.price, color = color.new(progressLabelColor, 5), style = candleLabelStyle, textcolor = color.white, size = size.tiny)

if not na(fiveMinuteClosedTime) and (acceptanceCompletedNow or failedAuctionCompletedNow or balancedCompletedNow)

    string outcomeText = acceptanceCompletedNow ? sequenceDirection == 1 ? "ACCEPTED ABOVE" : "ACCEPTED BELOW" : failedAuctionCompletedNow ? sequenceDirection == 1 ? "FAILED ABOVE" : "FAILED BELOW" : "BALANCED"

    color outcomeColor = acceptanceCompletedNow ? (sequenceDirection == 1 ? bullColorInput : bearColorInput) : failedAuctionCompletedNow ? (sequenceDirection == 1 ? bearColorInput : bullColorInput) : neutralColorInput

    float outcomePrice = sequenceDirection == 1 ? fiveMinuteLow : fiveMinuteHigh

    label.new(fiveMinuteClosedTime + 5 * 60 * 1000, outcomePrice, outcomeText, xloc = xloc.bar_time, yloc = yloc.price, color = color.new(outcomeColor, 5), style = label.style_label_left, textcolor = color.white, size = size.small)

```
if barstate.islast
    if showDashboardInput
        string timeframeStatus = chartSupportsFiveMinute ? "5m CLOSES" : "USE CHART TF <= 5M"
        string levelStatus = not na(activeLevel) ? activeLevelName + " @ " + str.tostring(activeLevel, format.mintick) : "WAITING FOR TOUCH"
        string profileStatus = array.size(completedRthProfiles) >= COMPOSITE_PROFILE_DAYS ? "5D COMPOSITE READY" : "5D PROFILES " + str.tostring(array.size(completedRthProfiles)) + "/" + str.tostring(COMPOSITE_PROFILE_DAYS)
        color storyColor = str.contains(latestStory, "ACCEPTED ABOVE") or str.contains(latestStory, "ABOVE ") or str.contains(latestStory, "FAILED BELOW") ? bullColorInput : str.contains(latestStory, "ACCEPTED BELOW") or str.contains(latestStory, "BELOW ") or str.contains(latestStory, "FAILED ABOVE") ? bearColorInput : neutralColorInput
        table.cell(storyTable, 0, 0, "THREE 5s", text_color = #DBDBDB, bgcolor = #161616, text_size = size.small)
        table.cell(storyTable, 1, 0, timeframeStatus, text_color = chartSupportsFiveMinute ? neutralColorInput : bearColorInput, bgcolor = #161616, text_size = size.small)
        table.cell(storyTable, 0, 1, "Active reference", text_color = #DBDBDB, bgcolor = #161616)
        table.cell(storyTable, 1, 1, levelStatus, text_color = not na(activeLevel) ? #f2c744 : #808080, bgcolor = #161616)
        table.cell(storyTable, 0, 2, "5-minute story", text_color = #DBDBDB, bgcolor = #161616)
```

```
    table.cell(storyTable, 1, 2, latestStory, text_color = storyColor, bgcolor = #161616)

    table.cell(storyTable, 0, 3, "Latest close", text_color = #808080, bgcolor = #161616, text_size = size.tiny)

    table.cell(storyTable, 1, 3, latestStoryDetail, text_color = #808080, bgcolor = #161616, text_size = size.tiny)

    table.cell(storyTable, 0, 4, "Level sources", text_color = #DBDBDB, bgcolor = #161616)

    table.cell(storyTable, 1, 4, "NY prior RTH + Globex | VRS prior RTH + 5D", text_color = #DBDBDB, bgcolor = #161616, text_size = size.tiny)

    table.cell(storyTable, 0, 5, "Profile history", text_color = #808080, bgcolor = #161616, text_size = size.tiny)

    table.cell(storyTable, 1, 5, profileStatus, text_color = array.size(completedRthProfiles) >= COMPOSITE_PROFILE_DAYS ? bullColorInput : #808080, bgcolor = #161616, text_size = size.tiny)

  else

    table.clear(storyTable, 0, 0, 1, 5)


// --- Alerts ---

alertcondition(levelSequenceStartedNow, "Three 5s: reference test started", "A confirmed 5-minute candle touched a tracked reference on {{ticker}}.")

alertcondition(acceptanceCompletedNow, "Three 5s: closes accepted beyond level", "The configured number of consecutive confirmed 5-minute closes held beyond a tracked reference on {{ticker}}.")

alertcondition(failedAuctionCompletedNow, "Three 5s: failed attempt back inside", "The configured number of consecutive confirmed 5-minute closes returned to or through a tracked reference on {{ticker}}.")
```

alertcondition(balancedCompletedNow, "Three 5s: balanced at reference", "The configured number of consecutive confirmed 5-minute closes remained in the balance zone at a tracked reference on {{ticker}}.")

// --- Reference-level core complete ---