//**@version=5**

indicator("Amoozesh-Boors.com MA & PI  ", "Amoozesh-Boors.com MA & PI  ", overlay=true, max\_lines\_count=500, max\_labels\_count=500)

ma(source, length, type) =>

    type == "SMA" ? ta.sma(source, length) :

     type == "EMA" ? ta.ema(source, length) :

     type == "SMMA (RMA)" ? ta.rma(source, length) :

     type == "WMA" ? ta.wma(source, length) :

     type == "VWMA" ? ta.vwma(source, length) :

     na

//

movingtime = input.string("Current", "Movingtime" , options=["Current","M","W","D","4h","1h"], inline="movingtime" )

show\_movings = input(true,"Show Movings", inline = "movingtime")

calcma(moving, mtime) =>

    request.security(syminfo.tickerid, mtime , moving)

//

show\_ma1   = input(true   , "MA №1", inline="MA #1")

ma1\_type   = input.string("EMA"  , ""     , inline="MA #1", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"])

ma1\_source = input(close  , ""     , inline="MA #1")

ma1\_length = input.int(20     , ""     , inline="MA #1", minval=1)

ma1\_color  = input(     #d79c07, ""     , inline="MA #1")

ma1 = ma(ma1\_source, ma1\_length, ma1\_type)

ma101 = ma1

show\_ma2   = input(true   , "MA №2", inline="MA #2")

ma2\_type   = input.string("EMA"  , ""     , inline="MA #2", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"])

ma2\_source = input(close  , ""     , inline="MA #2")

ma2\_length = input.int(50     , ""     , inline="MA #2", minval=1)

ma2\_color  = input( #0000FF, ""     , inline="MA #2")

ma2 = ma(ma2\_source, ma2\_length, ma2\_type)

ma201 = ma2

show\_ma3   = input(true   , "MA №3", inline="MA #3")

ma3\_type   = input.string("EMA"  , ""     , inline="MA #3", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"])

ma3\_source = input(close  , ""     , inline="MA #3")

ma3\_length = input.int(100    , ""     , inline="MA #3", minval=1)

ma3\_color  = input( #458B00, ""     , inline="MA #3")

ma3 = ma(ma3\_source, ma3\_length, ma3\_type)

ma301=ma3

show\_ma0   = input(false   , "MA №0", inline="MA #0")

ma0\_type   = input.string("EMA"  , ""     , inline="MA #0", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"])

ma0\_source = input(close  , ""     , inline="MA #0")

ma0\_length = input.int(150     , ""     , inline="MA #0", minval=1)

ma0\_color  = input(     #000000, ""     , inline="MA #0")

ma0 = ma(ma0\_source, ma0\_length, ma0\_type)

ma001=ma0

show\_ma4   = input(true   , "MA №4", inline="MA #4")

ma4\_type   = input.string("SMA"  , ""     , inline="MA #4", options=["SMA", "EMA", "SMMA (RMA)", "WMA", "VWMA"])

ma4\_source = input(close  , ""     , inline="MA #4")

ma4\_length = input.int(200    , ""     , inline="MA #4", minval=1)

ma4\_color  = input( #CD1076, ""     , inline="MA #4")

ma4 = ma(ma4\_source, ma4\_length, ma4\_type)

ma401=ma4

if movingtime == "M"

    ma101 := calcma(ma1,'M')

    ma201 := calcma(ma2,'M')

    ma301 := calcma(ma3,'M')

    ma001 := calcma(ma0,'M')

    ma401 := calcma(ma4,'M')

else if movingtime == "W"

    ma101:= calcma(ma1,'W')

    ma201:= calcma(ma2,'W')

    ma301:= calcma(ma3,'W')

    ma001:= calcma(ma0,'W')

    ma401:= calcma(ma4,'W')

else if movingtime =="D"

    ma101:=calcma(ma1,'D')

    ma201:=calcma(ma2,'D')

    ma301:=calcma(ma3,'D')

    ma001:=calcma(ma0,'D')

    ma401:=calcma(ma4,'D')

else if movingtime == "4h"

    ma101:= calcma(ma1, '240')

    ma201:= calcma(ma2, '240')

    ma301:= calcma(ma3, '240')

    ma001:= calcma(ma0, '240')

    ma401:= calcma(ma4, '240')

else if movingtime == "1h"

    ma101:=calcma(ma1,'60')

    ma201:=calcma(ma2,'60')

    ma301:=calcma(ma3,'60')

    ma001:=calcma(ma0,'60')

    ma401:=calcma(ma4,'60')

//

plot(show\_ma1 and show\_movings ? ma101 : na, color = ma1\_color, title="MA №1")

plot(show\_ma2 and show\_movings ? ma201 : na, color = ma2\_color, title="MA №2")

plot(show\_ma3 and show\_movings ? ma301 : na, color = ma3\_color, title="MA №3")

plot(show\_ma0 and show\_movings ? ma001 : na, color = ma0\_color, title="MA №0")

plot(show\_ma4 and show\_movings ? ma401 : na, color = ma4\_color, title="MA №4")

// start copy to end

//Pvt#1

AUTO = "Auto"

DAILY = "Daily"

WEEKLY = "Weekly"

MONTHLY = "Monthly"

QUARTERLY = "Quarterly"

YEARLY = "Yearly"

BIYEARLY = "Biyearly"

TRIYEARLY = "Triyearly"

QUINQUENNIALLY = "Quinquennially"

DECENNIALLY = "Decennially"

TRADITIONAL = "Traditional"

FIBONACCI = "Fibonacci"

WOODIE = "Woodie"

CLASSIC = "Classic"

DEMARK = "DM"

CAMARILLA = "Camarilla"

show\_pivot1 = input.bool(title="Show Pvt#1",defval=true,group="Pvt#1")

kind = input.string(title="Type", defval="Traditional", options=[TRADITIONAL, FIBONACCI, WOODIE, CLASSIC, DEMARK, CAMARILLA],inline = "P1",group="Pvt#1")

pivot\_time\_frame = input.string(title="Time", defval=MONTHLY, options=[AUTO, DAILY, WEEKLY, MONTHLY, QUARTERLY, YEARLY, BIYEARLY, TRIYEARLY, QUINQUENNIALLY, DECENNIALLY],group="Pvt#1", inline = "P1")

look\_back = input.int(title="Pvt#", defval=1, minval=1, maxval=5000,inline="P1",group="Pvt#1")

is\_daily\_based = input.bool(title="Use Daily-based Values",group="Pvt#1", defval=true, tooltip="When this option is unchecked, Pivot Points will use intraday data while calculating on intraday charts. If Extended Hours are displayed on the chart, they will be taken into account during the pivot level calculation. If intraday OHLC values are different from daily-based values (normal for stocks), the pivot levels will also differ.")

show\_labels = input.bool(title="Show Labels", defval=true ,group="Pvt#1",inline="Lable #1")

position\_labels = input.string("Left", "Labels Position", options=["Left", "Right"],group="Pvt#1",inline="Lable #1")

line\_width = input.int(title="Line Width", defval=1, minval=1, maxval=100,group="Pvt#1",inline="Lable #1")

//Pvt#2

AUTO02 = "Auto"

DAILY02 = "Daily"

WEEKLY02 = "Weekly"

MONTHLY02 = "Monthly"

QUARTERLY02 = "Quarterly"

YEARLY02 = "Yearly"

BIYEARLY02 = "Biyearly"

TRIYEARLY02 = "Triyearly"

QUINQUENNIALLY02 = "Quinquennially"

DECENNIALLY02 = "Decennially"

TRADITIONAL02 = "Traditional"

FIBONACCI02 = "Fibonacci"

WOODIE02 = "Woodie"

CLASSIC02 = "Classic"

DEMARK02 = "DM"

CAMARILLA02 = "Camarilla"

show\_pivot2 = input.bool(title="Show Pvt#2",defval=true,group="Pvt#2")

kind02 = input.string(title="Type", defval="Traditional", options=[TRADITIONAL02, FIBONACCI02, WOODIE02, CLASSIC02, DEMARK02, CAMARILLA02],inline = "P2",group="Pvt#2")

pivot\_time\_frame02 = input.string(title="Time", defval=WEEKLY02, options=[AUTO02, DAILY02, WEEKLY02, MONTHLY02, QUARTERLY02, YEARLY02, BIYEARLY02, TRIYEARLY02, QUINQUENNIALLY02, DECENNIALLY02],inline = "P2",group="Pvt#2")

look\_back02 = input.int(title="Pvt#", defval=1, minval=1, maxval=5000,inline = "P2",group="Pvt#2")

is\_daily\_based02 = input.bool(title="Use Daily-based Values",group="Pvt#2", defval=true, tooltip="When this option is unchecked, Pivot Points will use intraday data while calculating on intraday charts. If Extended Hours are displayed on the chart, they will be taken into account during the pivot level calculation. If intraday OHLC values are different from daily-based values (normal for stocks), the pivot levels will also differ.")

show\_labels02 = input.bool(title="Show Labels", defval=true, group="Pvt#2",inline="Lable#2")

position\_labels02 = input.string("Left", "Labels Position", options=["Left", "Right"], group="Pvt#2",inline="Lable#2")

line\_width02 = input.int(title="Line Width", defval=1, minval=1, maxval=100, group="Pvt#2",inline="Lable#2")

var DEF\_COLOR =#c46c00

var arr\_time = array.new\_int()

var p = array.new\_float()

p\_color = input.color(DEF\_COLOR, "P‏  ‏  ‏", inline="P", group="levels")

p\_show = input.bool(true, "", inline="P", group="levels")

var r1 = array.new\_float()

var s1 = array.new\_float()

s1\_color = input.color(DEF\_COLOR, "S1", inline="S1/R1" , group="levels")

s1\_show = input.bool(true, "", inline="S1/R1", group="levels")

r1\_color = input.color(DEF\_COLOR, "‏  ‏  ‏  ‏  ‏  ‏  ‏  ‏R1", inline="S1/R1", group="levels")

r1\_show = input.bool(true, "", inline="S1/R1", group="levels")

var r2 = array.new\_float()

var s2 = array.new\_float()

s2\_color = input.color(DEF\_COLOR, "S2", inline="S2/R2", group="levels")

s2\_show = input.bool(true, "", inline="S2/R2", group="levels")

r2\_color = input.color(DEF\_COLOR, "‏  ‏  ‏  ‏  ‏  ‏  ‏  ‏R2", inline="S2/R2", group="levels")

r2\_show = input.bool(true, "", inline="S2/R2", group="levels")

var r3 = array.new\_float()

var s3 = array.new\_float()

s3\_color = input.color(DEF\_COLOR, "S3", inline="S3/R3", group="levels")

s3\_show = input.bool(true, "", inline="S3/R3", group="levels")

r3\_color = input.color(DEF\_COLOR, "‏  ‏  ‏  ‏  ‏  ‏  ‏  ‏R3", inline="S3/R3", group="levels")

r3\_show = input.bool(true, "", inline="S3/R3", group="levels")

var r4 = array.new\_float()

var s4 = array.new\_float()

var r5 = array.new\_float()

var s5 = array.new\_float()

pivotX\_open = float(na)

pivotX\_open := nz(pivotX\_open[1], open)

pivotX\_high = float(na)

pivotX\_high := nz(pivotX\_high[1], high)

pivotX\_low = float(na)

pivotX\_low := nz(pivotX\_low[1], low)

pivotX\_prev\_open = float(na)

pivotX\_prev\_open := nz(pivotX\_prev\_open[1])

pivotX\_prev\_high = float(na)

pivotX\_prev\_high := nz(pivotX\_prev\_high[1])

pivotX\_prev\_low = float(na)

pivotX\_prev\_low := nz(pivotX\_prev\_low[1])

pivotX\_prev\_close = float(na)

pivotX\_prev\_close := nz(pivotX\_prev\_close[1])

get\_pivot\_resolution() =>

    resolution = "M"

**if** pivot\_time\_frame == AUTO

        if timeframe.isintraday

            resolution := timeframe.multiplier <= 15 ? "D" : "W"

        else if timeframe.isweekly or timeframe.ismonthly

            resolution := "12M"

**else** if pivot\_time\_frame == DAILY

        resolution := "D"

**else** if pivot\_time\_frame == WEEKLY

        resolution := "W"

**else** if pivot\_time\_frame == MONTHLY

        resolution := "M"

**else** if pivot\_time\_frame == QUARTERLY

        resolution := "3M"

**else** if pivot\_time\_frame == YEARLY or pivot\_time\_frame == BIYEARLY or pivot\_time\_frame == TRIYEARLY or pivot\_time\_frame == QUINQUENNIALLY or pivot\_time\_frame == DECENNIALLY

        resolution := "12M"

    resolution

var lines = array.new\_line()

var labels = array.new\_label()

draw\_line(i, pivot, col) =>

    if array.size(arr\_time) > 1

        array.push(lines, line.new(array.get(arr\_time, i), array.get(pivot, i), array.get(arr\_time, i + 1), array.get(pivot, i), color=col, xloc=xloc.bar\_time, width=line\_width))

draw\_label(i, y, txt, txt\_color) =>

    if (show\_labels  and not na(y))

        display\_text = (show\_labels ? txt : "")

        label\_style = position\_labels == "Left" ? label.style\_label\_right : label.style\_label\_left

        x = position\_labels == "Left" ? array.get(arr\_time, i) : array.get(arr\_time, i + 1)

        array.push(labels, label.new(x = x, y=y, text=display\_text, textcolor=txt\_color, style=label\_style, color=#00000000, xloc=xloc.bar\_time))

traditional() =>

    pivotX\_Median = (pivotX\_prev\_high + pivotX\_prev\_low + pivotX\_prev\_close) / 3

    array.push(p, pivotX\_Median)

    array.push(r1, pivotX\_Median \* 2 - pivotX\_prev\_low)

    array.push(s1, pivotX\_Median \* 2 - pivotX\_prev\_high)

    array.push(r2, pivotX\_Median + 1 \* (pivotX\_prev\_high - pivotX\_prev\_low))

    array.push(s2, pivotX\_Median - 1 \* (pivotX\_prev\_high - pivotX\_prev\_low))

    array.push(r3, pivotX\_Median \* 2 + (pivotX\_prev\_high - 2 \* pivotX\_prev\_low))

    array.push(s3, pivotX\_Median \* 2 - (2 \* pivotX\_prev\_high - pivotX\_prev\_low))

fibonacci() =>

    pivotX\_Median = (pivotX\_prev\_high + pivotX\_prev\_low + pivotX\_prev\_close) / 3

    pivot\_range = pivotX\_prev\_high - pivotX\_prev\_low

    array.push(p, pivotX\_Median)

    array.push(r1, pivotX\_Median + 0.382 \* pivot\_range)

    array.push(s1, pivotX\_Median - 0.382 \* pivot\_range)

    array.push(r2, pivotX\_Median + 0.618 \* pivot\_range)

    array.push(s2, pivotX\_Median - 0.618 \* pivot\_range)

    array.push(r3, pivotX\_Median + 1 \* pivot\_range)

    array.push(s3, pivotX\_Median - 1 \* pivot\_range)

woodie() =>

    pivotX\_Woodie\_Median = (pivotX\_prev\_high + pivotX\_prev\_low + pivotX\_open \* 2)/4

    pivot\_range = pivotX\_prev\_high - pivotX\_prev\_low

    array.push(p, pivotX\_Woodie\_Median)

    array.push(r1, pivotX\_Woodie\_Median \* 2 - pivotX\_prev\_low)

    array.push(s1, pivotX\_Woodie\_Median \* 2 - pivotX\_prev\_high)

    array.push(r2, pivotX\_Woodie\_Median + 1 \* pivot\_range)

    array.push(s2, pivotX\_Woodie\_Median - 1 \* pivot\_range)

    pivot\_point\_r3 = pivotX\_prev\_high + 2 \* (pivotX\_Woodie\_Median - pivotX\_prev\_low)

    pivot\_point\_s3 = pivotX\_prev\_low - 2 \* (pivotX\_prev\_high - pivotX\_Woodie\_Median)

    array.push(r3, pivot\_point\_r3)

    array.push(s3, pivot\_point\_s3)

    array.push(r4, pivot\_point\_r3 + pivot\_range)

    array.push(s4, pivot\_point\_s3 - pivot\_range)

classic() =>

    pivotX\_Median = (pivotX\_prev\_high + pivotX\_prev\_low + pivotX\_prev\_close)/3

    pivot\_range = pivotX\_prev\_high - pivotX\_prev\_low

    array.push(p, pivotX\_Median)

    array.push(r1, pivotX\_Median \* 2 - pivotX\_prev\_low)

    array.push(s1, pivotX\_Median \* 2 - pivotX\_prev\_high)

    array.push(r2, pivotX\_Median + 1 \* pivot\_range)

    array.push(s2, pivotX\_Median - 1 \* pivot\_range)

    array.push(r3, pivotX\_Median + 2 \* pivot\_range)

    array.push(s3, pivotX\_Median - 2 \* pivot\_range)

    array.push(r4, pivotX\_Median + 3 \* pivot\_range)

    array.push(s4, pivotX\_Median - 3 \* pivot\_range)

demark() =>

    pivotX\_Demark\_X = pivotX\_prev\_high + pivotX\_prev\_low \* 2 + pivotX\_prev\_close

**if** pivotX\_prev\_close == pivotX\_prev\_open

        pivotX\_Demark\_X := pivotX\_prev\_high + pivotX\_prev\_low + pivotX\_prev\_close \* 2

    if pivotX\_prev\_close > pivotX\_prev\_open

        pivotX\_Demark\_X := pivotX\_prev\_high \* 2 + pivotX\_prev\_low + pivotX\_prev\_close

    array.push(p, pivotX\_Demark\_X / 4)

    array.push(r1, pivotX\_Demark\_X / 2 - pivotX\_prev\_low)

    array.push(s1, pivotX\_Demark\_X / 2 - pivotX\_prev\_high)

camarilla() =>

    pivotX\_Median = (pivotX\_prev\_high + pivotX\_prev\_low + pivotX\_prev\_close) / 3

    pivot\_range = pivotX\_prev\_high - pivotX\_prev\_low

    array.push(p, pivotX\_Median)

    array.push(r1, pivotX\_prev\_close + pivot\_range \* 1.1 / 12.0)

    array.push(s1, pivotX\_prev\_close - pivot\_range \* 1.1 / 12.0)

    array.push(r2, pivotX\_prev\_close + pivot\_range \* 1.1 / 6.0)

    array.push(s2, pivotX\_prev\_close - pivot\_range \* 1.1 / 6.0)

    array.push(r3, pivotX\_prev\_close + pivot\_range \* 1.1 / 4.0)

    array.push(s3, pivotX\_prev\_close - pivot\_range \* 1.1 / 4.0)

    array.push(r4, pivotX\_prev\_close + pivot\_range \* 1.1 / 2.0)

    array.push(s4, pivotX\_prev\_close - pivot\_range \* 1.1 / 2.0)

    r5\_val = pivotX\_prev\_high / pivotX\_prev\_low \* pivotX\_prev\_close

    array.push(r5, r5\_val)

    array.push(s5, 2 \* pivotX\_prev\_close - r5\_val)

calc\_pivot() =>

**if** kind == TRADITIONAL

        traditional()

**else** if kind == FIBONACCI

        fibonacci()

**else** if kind == WOODIE

        woodie()

**else** if kind == CLASSIC

        classic()

**else** if kind == DEMARK

        demark()

**else** if kind == CAMARILLA

        camarilla()

resolution = get\_pivot\_resolution()

SIMPLE\_DIVISOR = -1

custom\_years\_divisor = switch pivot\_time\_frame

    BIYEARLY => 2

    TRIYEARLY => 3

    QUINQUENNIALLY => 5

    DECENNIALLY => 10

    => SIMPLE\_DIVISOR

calc\_high(prev, curr) =>

    if na(prev) or na(curr)

        nz(prev, nz(curr, na))

    else

        math.max(prev, curr)

calc\_low(prev, curr) =>

    if not na(prev) and not na(curr)

        math.min(prev, curr)

    else

        nz(prev, nz(curr, na))

calc\_OHLC\_for\_pivot(custom\_years\_divisor) =>

**if** custom\_years\_divisor == SIMPLE\_DIVISOR

        [open, high, low, close, open[1], high[1], low[1], close[1], time[1], time\_close]

    else

**var** prev\_sec\_open = float(na)

**var** prev\_sec\_high = float(na)

**var** prev\_sec\_low = float(na)

**var** prev\_sec\_close = float(na)

**var** prev\_sec\_time = int(na)

**var** curr\_sec\_open = float(na)

**var** curr\_sec\_high = float(na)

**var** curr\_sec\_low = float(na)

**var** curr\_sec\_close = float(na)

        if year(time\_close) % custom\_years\_divisor == 0

            curr\_sec\_open := open

            curr\_sec\_high := high

            curr\_sec\_low := low

            curr\_sec\_close := close

            prev\_sec\_high := high[1]

            prev\_sec\_low := low[1]

            prev\_sec\_close := close[1]

            prev\_sec\_time := time[1]

**for** i = 2 to custom\_years\_divisor

                prev\_sec\_open :=  nz(open[i], prev\_sec\_open)

                prev\_sec\_high := calc\_high(prev\_sec\_high, high[i])

                prev\_sec\_low := calc\_low(prev\_sec\_low, low[i])

                prev\_sec\_time := nz(time[i], prev\_sec\_time)

        [curr\_sec\_open, curr\_sec\_high, curr\_sec\_low, curr\_sec\_close, prev\_sec\_open, prev\_sec\_high, prev\_sec\_low, prev\_sec\_close, prev\_sec\_time, time\_close]

[sec\_open, sec\_high, sec\_low, sec\_close, prev\_sec\_open, prev\_sec\_high, prev\_sec\_low, prev\_sec\_close, prev\_sec\_time, sec\_time] = request.security(syminfo.tickerid, resolution, calc\_OHLC\_for\_pivot(custom\_years\_divisor), lookahead = barmerge.lookahead\_on)

sec\_open\_gaps\_on = request.security(syminfo.tickerid, resolution, open, gaps = barmerge.gaps\_on, lookahead = barmerge.lookahead\_on)

is\_change\_years = custom\_years\_divisor > 0 and ta.change(time(resolution)) and year(time\_close) % custom\_years\_divisor == 0

var is\_change = false

var uses\_current\_bar = timeframe.isintraday and kind == WOODIE

var change\_time = int(na)

is\_time\_change = (ta.change(time(resolution)) and custom\_years\_divisor == SIMPLE\_DIVISOR) or is\_change\_years

if is\_time\_change

    change\_time := time

var start\_time = time

var was\_last\_premarket = false

var start\_calculate\_in\_premarket = false

is\_last\_premarket = barstate.islast and session.ispremarket and time\_close > sec\_time and not was\_last\_premarket

if is\_last\_premarket

    was\_last\_premarket := true

    start\_calculate\_in\_premarket := true

if session.ismarket

    was\_last\_premarket := false

without\_time\_change = barstate.islast and array.size(arr\_time) == 0

is\_can\_calc\_pivot = (not uses\_current\_bar and is\_time\_change and session.ismarket) or (ta.change(sec\_open) and not start\_calculate\_in\_premarket) or is\_last\_premarket or (uses\_current\_bar and not na(sec\_open\_gaps\_on)) or without\_time\_change

enough\_bars\_for\_calculate = prev\_sec\_time >= start\_time or is\_daily\_based

if is\_can\_calc\_pivot and enough\_bars\_for\_calculate

    if array.size(arr\_time) == 0 and is\_daily\_based

        pivotX\_prev\_open := prev\_sec\_open[1]

        pivotX\_prev\_high := prev\_sec\_high[1]

        pivotX\_prev\_low := prev\_sec\_low[1]

        pivotX\_prev\_close := prev\_sec\_close[1]

        pivotX\_open := sec\_open[1]

        pivotX\_high := sec\_high[1]

        pivotX\_low := sec\_low[1]

        array.push(arr\_time, start\_time)

        calc\_pivot()

**if** is\_daily\_based

**if** is\_last\_premarket

            pivotX\_prev\_open := sec\_open

            pivotX\_prev\_high := sec\_high

            pivotX\_prev\_low := sec\_low

            pivotX\_prev\_close := sec\_close

            pivotX\_open := open

            pivotX\_high := high

            pivotX\_low := low

        else

            pivotX\_prev\_open := prev\_sec\_open

            pivotX\_prev\_high := prev\_sec\_high

            pivotX\_prev\_low := prev\_sec\_low

            pivotX\_prev\_close := prev\_sec\_close

            pivotX\_open := sec\_open

            pivotX\_high := sec\_high

            pivotX\_low := sec\_low

    else

        pivotX\_prev\_high := pivotX\_high

        pivotX\_prev\_low := pivotX\_low

        pivotX\_prev\_open := pivotX\_open

        pivotX\_prev\_close := close[1]

        pivotX\_open := open

        pivotX\_high := high

        pivotX\_low := low

    if barstate.islast and not is\_change and array.size(arr\_time) > 0 and not without\_time\_change

        array.set(arr\_time, array.size(arr\_time) - 1, change\_time)

**else** if without\_time\_change

        array.push(arr\_time, start\_time)

    else

        array.push(arr\_time, nz(change\_time, time))

    calc\_pivot()

    if array.size(arr\_time) > look\_back

        if array.size(arr\_time) > 0

            array.shift(arr\_time)

        if array.size(p) > 0 and p\_show

            array.shift(p)

        if array.size(r1) > 0 and r1\_show

            array.shift(r1)

        if array.size(s1) > 0 and s1\_show

            array.shift(s1)

        if array.size(r2) > 0 and r2\_show

            array.shift(r2)

        if array.size(s2) > 0 and s2\_show

            array.shift(s2)

        if array.size(r3) > 0 and r3\_show

            array.shift(r3)

        if array.size(s3) > 0 and s3\_show

            array.shift(s3)

    is\_change := true

else if not is\_daily\_based

    pivotX\_high := math.max(pivotX\_high, high)

    pivotX\_low := math.min(pivotX\_low, low)

if barstate.islast and not is\_daily\_based and array.size(arr\_time) == 0

    runtime.error("Not enough intraday data to calculate Pivot Points. Lower the Pivots Timeframe or turn on the 'Use Daily-based Values' option in the indicator settings.")

if barstate.islast and array.size(arr\_time) > 0 and is\_change

    is\_change := false

    if custom\_years\_divisor > 0

        last\_pivot\_time = array.get(arr\_time, array.size(arr\_time) - 1)

        pivot\_timeframe = str.tostring(12 \* custom\_years\_divisor) + "M"

        estimate\_pivot\_time = last\_pivot\_time + timeframe.in\_seconds(pivot\_timeframe) \* 1000

        array.push(arr\_time, estimate\_pivot\_time)

    else

        array.push(arr\_time, time\_close(resolution))

**for** i = 0 to array.size(lines) - 1

        if array.size(lines) > 0

            line.delete(array.shift(lines))

        if array.size(labels) > 0

            label.delete(array.shift(labels))

**for** i = 0 to array.size(arr\_time) - 2

        if array.size(p) > 0 and p\_show and show\_pivot1

            draw\_line(i, p, p\_color)

            draw\_label(i, array.get(p, i), "P", p\_color)

        if array.size(r1) > 0 and r1\_show and show\_pivot1

            draw\_line(i, r1, r1\_color)

            draw\_label(i, array.get(r1, i), "R1", r1\_color)

        if array.size(s1) > 0 and s1\_show and show\_pivot1

            draw\_line(i, s1, s1\_color)

            draw\_label(i, array.get(s1, i), "S1", s1\_color)

        if array.size(r2) > 0 and r2\_show and show\_pivot1

            draw\_line(i, r2, r2\_color)

            draw\_label(i, array.get(r2, i), "R2", r2\_color)

        if array.size(s2) > 0 and s2\_show and show\_pivot1

            draw\_line(i, s2, s2\_color)

            draw\_label(i, array.get(s2, i), "S2", s2\_color)

        if array.size(r3) > 0 and r3\_show and show\_pivot1

            draw\_line(i, r3, r3\_color)

            draw\_label(i, array.get(r3, i), "R3", r3\_color)

        if array.size(s3) > 0 and s3\_show and show\_pivot1

            draw\_line(i, s3, s3\_color)

            draw\_label(i, array.get(s3, i), "S3", s3\_color)

// Past from COPY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

var DEF\_COLOR02 =#c46c00

var arr\_time02 = array.new\_int()

var p02 = array.new\_float()

p\_color02 = input.color(DEF\_COLOR02, "P‏  ‏  ‏", inline="P", group="levels")

p\_show02 = input.bool(true, "", inline="P", group="levels")

var r102 = array.new\_float()

var s102 = array.new\_float()

s1\_color02 = input.color(DEF\_COLOR02, "S1", inline="S1/R1" , group="levels")

s1\_show02 = input.bool(true, "", inline="S1/R1", group="levels")

r1\_color02 = input.color(DEF\_COLOR02, "‏  ‏  ‏  ‏  ‏  ‏  ‏  ‏R1", inline="S1/R1", group="levels")

r1\_show02 = input.bool(true, "", inline="S1/R1", group="levels")

var r202 = array.new\_float()

var s202 = array.new\_float()

s2\_color02 = input.color(DEF\_COLOR02, "S2", inline="S2/R2", group="levels")

s2\_show02 = input.bool(true, "", inline="S2/R2", group="levels")

r2\_color02 = input.color(DEF\_COLOR02, "‏  ‏  ‏  ‏  ‏  ‏  ‏  ‏R2", inline="S2/R2", group="levels")

r2\_show02 = input.bool(true, "", inline="S2/R2", group="levels")

var r302 = array.new\_float()

var s302 = array.new\_float()

s3\_color02 = input.color(DEF\_COLOR02, "S3", inline="S3/R3", group="levels")

s3\_show02 = input.bool(true, "", inline="S3/R3", group="levels")

r3\_color02 = input.color(DEF\_COLOR02, "‏  ‏  ‏  ‏  ‏  ‏  ‏  ‏R3", inline="S3/R3", group="levels")

r3\_show02 = input.bool(true, "", inline="S3/R3", group="levels")

var r402 = array.new\_float()

var s402 = array.new\_float()

var r502 = array.new\_float()

var s502 = array.new\_float()

pivotX\_open02 = float(na)

pivotX\_open02 := nz(pivotX\_open02[1], open)

pivotX\_high02 = float(na)

pivotX\_high02 := nz(pivotX\_high02[1], high)

pivotX\_low02 = float(na)

pivotX\_low02 := nz(pivotX\_low02[1], low)

pivotX\_prev\_open02 = float(na)

pivotX\_prev\_open02 := nz(pivotX\_prev\_open02[1])

pivotX\_prev\_high02 = float(na)

pivotX\_prev\_high02 := nz(pivotX\_prev\_high02[1])

pivotX\_prev\_low02 = float(na)

pivotX\_prev\_low02 := nz(pivotX\_prev\_low02[1])

pivotX\_prev\_close02 = float(na)

pivotX\_prev\_close02 := nz(pivotX\_prev\_close02[1])

get\_pivot\_resolution02() =>

    resolution02 = "M"

**if** pivot\_time\_frame02 == AUTO02

        if timeframe.isintraday

            resolution02 := timeframe.multiplier <= 15 ? "D" : "W"

        else if timeframe.isweekly or timeframe.ismonthly

            resolution02 := "12M"

**else** if pivot\_time\_frame02 == DAILY02

        resolution02 := "D"

**else** if pivot\_time\_frame02 == WEEKLY02

        resolution02 := "W"

**else** if pivot\_time\_frame02 == MONTHLY02

        resolution02 := "M"

**else** if pivot\_time\_frame02 == QUARTERLY02

        resolution02 := "3M"

**else** if pivot\_time\_frame02 == YEARLY02 or pivot\_time\_frame02 == BIYEARLY02 or pivot\_time\_frame02 == TRIYEARLY02 or pivot\_time\_frame02 == QUINQUENNIALLY02 or pivot\_time\_frame02 == DECENNIALLY02

        resolution02 := "12M"

    resolution02

var lines02 = array.new\_line()

var labels02 = array.new\_label()

draw\_line02(i, pivot, col) =>

    if array.size(arr\_time02) > 1

        array.push(lines02, line.new(array.get(arr\_time02, i), array.get(pivot, i), array.get(arr\_time02, i + 1), array.get(pivot, i), color=col, xloc=xloc.bar\_time, width=line\_width02))

draw\_label02(i, y, txt, txt\_color) =>

    if (show\_labels02  and not na(y))

        display\_text02 = (show\_labels02 ? txt : "")

        label\_style02 = position\_labels == "Left" ? label.style\_label\_right : label.style\_label\_left

        x02 = position\_labels == "Left" ? array.get(arr\_time02, i) : array.get(arr\_time02, i + 1)

        array.push(labels02, label.new(x = x02, y=y, text=display\_text02, textcolor=txt\_color, style=label\_style02, color=#00000000, xloc=xloc.bar\_time))

traditional02() =>

    pivotX\_Median02 = (pivotX\_prev\_high02 + pivotX\_prev\_low02 + pivotX\_prev\_close02) / 3

    array.push(p02, pivotX\_Median02)

    array.push(r102, pivotX\_Median02 \* 2 - pivotX\_prev\_low02)

    array.push(s102, pivotX\_Median02 \* 2 - pivotX\_prev\_high02)

    array.push(r202, pivotX\_Median02 + 1 \* (pivotX\_prev\_high02 - pivotX\_prev\_low02))

    array.push(s202, pivotX\_Median02 - 1 \* (pivotX\_prev\_high02 - pivotX\_prev\_low02))

    array.push(r302, pivotX\_Median02 \* 2 + (pivotX\_prev\_high02 - 2 \* pivotX\_prev\_low02))

    array.push(s302, pivotX\_Median02 \* 2 - (2 \* pivotX\_prev\_high02 - pivotX\_prev\_low02))

fibonacci02() =>

    pivotX\_Median02 = (pivotX\_prev\_high02 + pivotX\_prev\_low02 + pivotX\_prev\_close02) / 3

    pivot\_range02 = pivotX\_prev\_high02 - pivotX\_prev\_low02

    array.push(p02, pivotX\_Median02)

    array.push(r102, pivotX\_Median02 + 0.382 \* pivot\_range02)

    array.push(s102, pivotX\_Median02 - 0.382 \* pivot\_range02)

    array.push(r202, pivotX\_Median02 + 0.618 \* pivot\_range02)

    array.push(s202, pivotX\_Median02 - 0.618 \* pivot\_range02)

    array.push(r302, pivotX\_Median02 + 1 \* pivot\_range02)

    array.push(s302, pivotX\_Median02 - 1 \* pivot\_range02)

woodie02() =>

    pivotX\_Woodie\_Median02 = (pivotX\_prev\_high02 + pivotX\_prev\_low02 + pivotX\_open02 \* 2)/4

    pivot\_range02 = pivotX\_prev\_high02 - pivotX\_prev\_low02

    array.push(p02, pivotX\_Woodie\_Median02)

    array.push(r102, pivotX\_Woodie\_Median02 \* 2 - pivotX\_prev\_low02)

    array.push(s102, pivotX\_Woodie\_Median02 \* 2 - pivotX\_prev\_high02)

    array.push(r202, pivotX\_Woodie\_Median02 + 1 \* pivot\_range02)

    array.push(s202, pivotX\_Woodie\_Median02 - 1 \* pivot\_range02)

    pivot\_point\_r302 = pivotX\_prev\_high02 + 2 \* (pivotX\_Woodie\_Median02 - pivotX\_prev\_low02)

    pivot\_point\_s302 = pivotX\_prev\_low02 - 2 \* (pivotX\_prev\_high02 - pivotX\_Woodie\_Median02)

    array.push(r302, pivot\_point\_r302)

    array.push(s302, pivot\_point\_s302)

    array.push(r402, pivot\_point\_r302 + pivot\_range02)

    array.push(s402, pivot\_point\_s302 - pivot\_range02)

classic02() =>

    pivotX\_Median02 = (pivotX\_prev\_high02 + pivotX\_prev\_low02 + pivotX\_prev\_close02)/3

    pivot\_range02 = pivotX\_prev\_high02 - pivotX\_prev\_low02

    array.push(p02, pivotX\_Median02)

    array.push(r102, pivotX\_Median02 \* 2 - pivotX\_prev\_low02)

    array.push(s102, pivotX\_Median02 \* 2 - pivotX\_prev\_high02)

    array.push(r202, pivotX\_Median02 + 1 \* pivot\_range02)

    array.push(s202, pivotX\_Median02 - 1 \* pivot\_range02)

    array.push(r302, pivotX\_Median02 + 2 \* pivot\_range02)

    array.push(s302, pivotX\_Median02 - 2 \* pivot\_range02)

    array.push(r402, pivotX\_Median02 + 3 \* pivot\_range02)

    array.push(s402, pivotX\_Median02 - 3 \* pivot\_range02)

demark02() =>

    pivotX\_Demark\_X02 = pivotX\_prev\_high02 + pivotX\_prev\_low02 \* 2 + pivotX\_prev\_close02

**if** pivotX\_prev\_close02 == pivotX\_prev\_open02

        pivotX\_Demark\_X02 := pivotX\_prev\_high02 + pivotX\_prev\_low02 + pivotX\_prev\_close02 \* 2

    if pivotX\_prev\_close02 > pivotX\_prev\_open02

        pivotX\_Demark\_X02 := pivotX\_prev\_high02 \* 2 + pivotX\_prev\_low02 + pivotX\_prev\_close02

    array.push(p02, pivotX\_Demark\_X02 / 4)

    array.push(r102, pivotX\_Demark\_X02 / 2 - pivotX\_prev\_low02)

    array.push(s102, pivotX\_Demark\_X02 / 2 - pivotX\_prev\_high02)

camarilla02() =>

    pivotX\_Median02 = (pivotX\_prev\_high02 + pivotX\_prev\_low02 + pivotX\_prev\_close02) / 3

    pivot\_range02 = pivotX\_prev\_high02 - pivotX\_prev\_low02

    array.push(p02, pivotX\_Median02)

    array.push(r102, pivotX\_prev\_close02 + pivot\_range02 \* 1.1 / 12.0)

    array.push(s102, pivotX\_prev\_close02 - pivot\_range02 \* 1.1 / 12.0)

    array.push(r202, pivotX\_prev\_close02 + pivot\_range02 \* 1.1 / 6.0)

    array.push(s202, pivotX\_prev\_close02 - pivot\_range02 \* 1.1 / 6.0)

    array.push(r302, pivotX\_prev\_close02 + pivot\_range02 \* 1.1 / 4.0)

    array.push(s302, pivotX\_prev\_close02 - pivot\_range02 \* 1.1 / 4.0)

    array.push(r402, pivotX\_prev\_close02 + pivot\_range02 \* 1.1 / 2.0)

    array.push(s402, pivotX\_prev\_close02 - pivot\_range02 \* 1.1 / 2.0)

    r5\_val02 = pivotX\_prev\_high02 / pivotX\_prev\_low02 \* pivotX\_prev\_close02

    array.push(r502, r5\_val02)

    array.push(s502, 2 \* pivotX\_prev\_close02 - r5\_val02)

calc\_pivot02() =>

**if** kind02 == TRADITIONAL02

        traditional02()

**else** if kind02 == FIBONACCI02

        fibonacci02()

**else** if kind02 == WOODIE02

        woodie02()

**else** if kind02 == CLASSIC02

        classic02()

**else** if kind02 == DEMARK02

        demark02()

**else** if kind02 == CAMARILLA02

        camarilla02()

resolution02 = get\_pivot\_resolution02()

SIMPLE\_DIVISOR02 = -1

custom\_years\_divisor02 = switch pivot\_time\_frame02

    BIYEARLY02 => 2

    TRIYEARLY02 => 3

    QUINQUENNIALLY02 => 5

    DECENNIALLY02 => 10

    => SIMPLE\_DIVISOR02

calc\_high02(prev, curr) =>

    if na(prev) or na(curr)

        nz(prev, nz(curr, na))

    else

        math.max(prev, curr)

calc\_low02(prev, curr) =>

    if not na(prev) and not na(curr)

        math.min(prev, curr)

    else

        nz(prev, nz(curr, na))

calc\_OHLC\_for\_pivot02(custom\_years\_divisor02) =>

**if** custom\_years\_divisor02 == SIMPLE\_DIVISOR02

        [open, high, low, close, open[1], high[1], low[1], close[1], time[1], time\_close]

    else

**var** prev\_sec\_open02 = float(na)

**var** prev\_sec\_high02 = float(na)

**var** prev\_sec\_low02 = float(na)

**var** prev\_sec\_close02 = float(na)

**var** prev\_sec\_time02 = int(na)

**var** curr\_sec\_open02 = float(na)

**var** curr\_sec\_high02 = float(na)

**var** curr\_sec\_low02 = float(na)

**var** curr\_sec\_close02 = float(na)

        if year(time\_close) % custom\_years\_divisor02 == 0

            curr\_sec\_open02 := open

            curr\_sec\_high02 := high

            curr\_sec\_low02 := low

            curr\_sec\_close02 := close

            prev\_sec\_high02 := high[1]

            prev\_sec\_low02 := low[1]

            prev\_sec\_close02 := close[1]

            prev\_sec\_time02 := time[1]

**for** i = 2 to custom\_years\_divisor02

                prev\_sec\_open02 :=  nz(open[i], prev\_sec\_open02)

                prev\_sec\_high02 := calc\_high02(prev\_sec\_high02, high[i])

                prev\_sec\_low02 := calc\_low02(prev\_sec\_low02, low[i])

                prev\_sec\_time02 := nz(time[i], prev\_sec\_time02)

        [curr\_sec\_open02, curr\_sec\_high02, curr\_sec\_low02, curr\_sec\_close02, prev\_sec\_open02, prev\_sec\_high02, prev\_sec\_low02, prev\_sec\_close02, prev\_sec\_time02, time\_close]

[sec\_open02, sec\_high02, sec\_low02, sec\_close02, prev\_sec\_open02, prev\_sec\_high02, prev\_sec\_low02, prev\_sec\_close02, prev\_sec\_time02, sec\_time02] = request.security(syminfo.tickerid, resolution02, calc\_OHLC\_for\_pivot02(custom\_years\_divisor02), lookahead = barmerge.lookahead\_on)

sec\_open\_gaps\_on02 = request.security(syminfo.tickerid, resolution02, open, gaps = barmerge.gaps\_on, lookahead = barmerge.lookahead\_on)

is\_change\_years02 = custom\_years\_divisor02 > 0 and ta.change(time(resolution02)) and year(time\_close) % custom\_years\_divisor02 == 0

var is\_change02 = false

var uses\_current\_bar02 = timeframe.isintraday and kind02 == WOODIE02

var change\_time02 = int(na)

is\_time\_change02 = (ta.change(time(resolution02)) and custom\_years\_divisor02 == SIMPLE\_DIVISOR02) or is\_change\_years02

if is\_time\_change02

    change\_time02 := time

var start\_time02 = time

var was\_last\_premarket02 = false

var start\_calculate\_in\_premarket02 = false

is\_last\_premarket02 = barstate.islast and session.ispremarket and time\_close > sec\_time02 and not was\_last\_premarket02

if is\_last\_premarket02

    was\_last\_premarket02 := true

    start\_calculate\_in\_premarket02 := true

if session.ismarket

    was\_last\_premarket02 := false

without\_time\_change02 = barstate.islast and array.size(arr\_time02) == 0

is\_can\_calc\_pivot02 = (not uses\_current\_bar02 and is\_time\_change02 and session.ismarket) or (ta.change(sec\_open02) and not start\_calculate\_in\_premarket02) or is\_last\_premarket02 or (uses\_current\_bar02 and not na(sec\_open\_gaps\_on02)) or without\_time\_change02

enough\_bars\_for\_calculate02 = prev\_sec\_time02 >= start\_time02 or is\_daily\_based02

if is\_can\_calc\_pivot02 and enough\_bars\_for\_calculate02

    if array.size(arr\_time02) == 0 and is\_daily\_based02

        pivotX\_prev\_open02 := prev\_sec\_open02[1]

        pivotX\_prev\_high02 := prev\_sec\_high02[1]

        pivotX\_prev\_low02 := prev\_sec\_low02[1]

        pivotX\_prev\_close02 := prev\_sec\_close02[1]

        pivotX\_open02 := sec\_open02[1]

        pivotX\_high02 := sec\_high02[1]

        pivotX\_low02 := sec\_low02[1]

        array.push(arr\_time02, start\_time02)

        calc\_pivot02()

**if** is\_daily\_based02

**if** is\_last\_premarket02

            pivotX\_prev\_open02 := sec\_open02

            pivotX\_prev\_high02 := sec\_high02

            pivotX\_prev\_low02 := sec\_low02

            pivotX\_prev\_close02 := sec\_close02

            pivotX\_open02 := open

            pivotX\_high02 := high

            pivotX\_low02 := low

        else

            pivotX\_prev\_open02 := prev\_sec\_open02

            pivotX\_prev\_high02 := prev\_sec\_high02

            pivotX\_prev\_low02 := prev\_sec\_low02

            pivotX\_prev\_close02 := prev\_sec\_close02

            pivotX\_open02 := sec\_open02

            pivotX\_high02 := sec\_high02

            pivotX\_low02 := sec\_low02

    else

        pivotX\_prev\_high02 := pivotX\_high02

        pivotX\_prev\_low02 := pivotX\_low02

        pivotX\_prev\_open02 := pivotX\_open02

        pivotX\_prev\_close02 := close[1]

        pivotX\_open02 := open

        pivotX\_high02 := high

        pivotX\_low02 := low

    if barstate.islast and not is\_change02 and array.size(arr\_time02) > 0 and not without\_time\_change02

        array.set(arr\_time02, array.size(arr\_time02) - 1, change\_time02)

**else** if without\_time\_change02

        array.push(arr\_time02, start\_time02)

    else

        array.push(arr\_time02, nz(change\_time02, time))

    calc\_pivot02()

    if array.size(arr\_time02) > look\_back02

        if array.size(arr\_time02) > 0

            array.shift(arr\_time02)

        if array.size(p02) > 0 and p\_show02

            array.shift(p02)

        if array.size(r102) > 0 and r1\_show02

            array.shift(r102)

        if array.size(s102) > 0 and s1\_show02

            array.shift(s102)

        if array.size(r202) > 0 and r2\_show02

            array.shift(r202)

        if array.size(s202) > 0 and s2\_show02

            array.shift(s202)

        if array.size(r302) > 0 and r3\_show02

            array.shift(r302)

        if array.size(s302) > 0 and s3\_show02

            array.shift(s302)

    is\_change02 := true

else if not is\_daily\_based02

    pivotX\_high02 := math.max(pivotX\_high02, high)

    pivotX\_low02 := math.min(pivotX\_low02, low)

if barstate.islast and not is\_daily\_based02 and array.size(arr\_time02) == 0

    runtime.error("Not enough intraday data to calculate Pivot Points. Lower the Pivots Timeframe or turn on the 'Use Daily-based Values' option in the indicator settings.")

if barstate.islast and array.size(arr\_time02) > 0 and is\_change02

    is\_change02 := false

    if custom\_years\_divisor02 > 0

        last\_pivot\_time02 = array.get(arr\_time02, array.size(arr\_time02) - 1)

        pivot\_timeframe02 = str.tostring(12 \* custom\_years\_divisor02) + "M"

        estimate\_pivot\_time02 = last\_pivot\_time02 + timeframe.in\_seconds(pivot\_timeframe02) \* 1000

        array.push(arr\_time02, estimate\_pivot\_time02)

    else

        array.push(arr\_time02, time\_close(resolution02))

**for** i = 0 to array.size(lines02) - 1

        if array.size(lines02) > 0

            line.delete(array.shift(lines02))

        if array.size(labels02) > 0

            label.delete(array.shift(labels02))

**for** i = 0 to array.size(arr\_time02) - 2

        if array.size(p02) > 0 and p\_show02 and show\_pivot2

            draw\_line02(i, p02, p\_color02)

            draw\_label02(i, array.get(p02, i), "P", p\_color02)

        if array.size(r102) > 0 and r1\_show02 and show\_pivot2

            draw\_line02(i, r102, r1\_color02)

            draw\_label02(i, array.get(r102, i), "R1", r1\_color02)

        if array.size(s102) > 0 and s1\_show02 and show\_pivot2

            draw\_line02(i, s102, s1\_color02)

            draw\_label02(i, array.get(s102, i), "S1", s1\_color02)

        if array.size(r202) > 0 and r2\_show02 and show\_pivot2

            draw\_line02(i, r202, r2\_color02)

            draw\_label02(i, array.get(r202, i), "R2", r2\_color02)

        if array.size(s202) > 0 and s2\_show02 and show\_pivot2

            draw\_line02(i, s202, s2\_color02)

            draw\_label02(i, array.get(s202, i), "S2", s2\_color02)

        if array.size(r302) > 0 and r3\_show02 and show\_pivot2

            draw\_line02(i, r302, r3\_color02)

            draw\_label02(i, array.get(r302, i), "R3", r3\_color02)

        if array.size(s302) > 0 and s3\_show02 and show\_pivot2

            draw\_line02(i, s302, s3\_color02)

            draw\_label02(i, array.get(s302, i), "S3", s3\_color02)

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

// The original indicator is created by ashkanpower and modified by FX365\_Thailand

// Original : Pip Value

// https://www.tradingview.com/script/OdV2qkiW-Pip-Value/

// Revision history

// v60.0 First release

// v61.0 Added logic for gold and silver

// v63.0 Added logic for BTC and ETH

//**@version=5**

//indicator("Pip Value Calculator", overlay=true)

// compact = input.bool(title="compact mode",defval=false)

acc\_currency = input.string(defval="USD", options=["USD","JPY", "EUR", "GBP", "AUD", "THB"], title="Account Currency")

pips = input.int(title="Pips",defval=10, minval=1, tooltip="[Gold]1USD=10pips,[Silver]1USD=100Pips,[BTC/ETH]1USD=10pips")

lots = input.float(title="Lots",defval=1, minval=0.01, tooltip="[Forex]1 lot is 100,000 unit of currency,[Gold]1lot=100ounce,[Silver]1lot=5000ounce,[BTC]1lot=1BTC,[ETH]1lot=1ETH")

// loss = input.float(title="Loss Amount in Account Currency",  defval=10000, minval=1)

// ratio = input.float(title="R&R(1:N), N = ? ", defval=2, minval=1, maxval=100)

fontcolor = input.color(defval=color.black, title="Font Color", inline="color" )

bgcolor = input.color(defval=color.white, title="Background Color", inline="color" )

cur = syminfo.currency

base = syminfo.basecurrency

usdjpy\_rate = request.security("USDJPY", timeframe.period, close)

usdeur\_rate = request.security("USDEUR", timeframe.period, close)

usdgbp\_rate = request.security("USDGBP", timeframe.period, close)

usdaud\_rate = request.security("USDAUD", timeframe.period, close)

usdthb\_rate = request.security("USDTHB", timeframe.period, close)

// baseusd =  (syminfo.ticker == 'XAGUSD' or syminfo.ticker == 'SILVER') ? 1 :  nz(request.security(base + "USD", timeframe.period, close, ignore\_invalid\_symbol=true), 1)

baseusd = nz(request.security(base + "USD", timeframe.period, close, ignore\_invalid\_symbol=true), 1)

jpyFixer = base == "JPY" or cur == "JPY" ? 100 : 1

profit\_forex = acc\_currency == "USD" ? math.round(pips \* (10 \* lots) / close \* baseusd \* jpyFixer , 2)

 : acc\_currency == "JPY" ? math.round(pips \* (10 \* lots) / close \* baseusd \* jpyFixer \* usdjpy\_rate, 2)

 : acc\_currency == "EUR" ? math.round(pips \* (10 \* lots) / close \* baseusd \* jpyFixer \* usdeur\_rate, 2)

 : acc\_currency == "GBP" ? math.round(pips \* (10 \* lots) / close \* baseusd \* jpyFixer \* usdgbp\_rate, 2)

 : acc\_currency == "AUD" ? math.round(pips \* (10 \* lots) / close \* baseusd \* jpyFixer \* usdaud\_rate, 2)

 : acc\_currency == "THB" ? math.round(pips \* (10 \* lots) / close \* baseusd \* jpyFixer \* usdthb\_rate, 2)

 : na

gold\_silver\_profit = (syminfo.ticker == 'XAGUSD' or syminfo.ticker == 'SILVER') ? pips \* 50 \* lots : (syminfo.ticker == 'XAUUSD' or syminfo.ticker == 'GOLD') ? pips \* 10 \* lots : na

profit\_commo = acc\_currency == "USD" ? math.round(gold\_silver\_profit,2)

 : acc\_currency == "JPY" ? math.round(gold\_silver\_profit \* usdjpy\_rate,2)

 : acc\_currency == "EUR" ? math.round(gold\_silver\_profit \* usdeur\_rate,2)

 : acc\_currency == "GBP" ? math.round(gold\_silver\_profit \* usdgbp\_rate,2)

 : acc\_currency == "AUD" ? math.round(gold\_silver\_profit \* usdaud\_rate,2)

 : acc\_currency == "THB" ? math.round(gold\_silver\_profit \* usdthb\_rate,2)

 : na

crypto\_profit = (syminfo.ticker == 'BTCUSD' ) ? pips \* 10 \* lots /100 :  (syminfo.ticker == 'ETHUSD') ? pips \* lots /10 : na

profit\_crypto = acc\_currency == "USD" ? math.round(crypto\_profit,2)

 : acc\_currency == "JPY" ? math.round(crypto\_profit \* usdjpy\_rate,2)

 : acc\_currency == "EUR" ? math.round(crypto\_profit \* usdeur\_rate,2)

 : acc\_currency == "GBP" ? math.round(crypto\_profit \* usdgbp\_rate,2)

 : acc\_currency == "AUD" ? math.round(crypto\_profit \* usdaud\_rate,2)

 : acc\_currency == "THB" ? math.round(crypto\_profit \* usdthb\_rate,2)

 : na

//Get profit for plotting

profit = (syminfo.ticker == 'XAGUSD' or syminfo.ticker == 'SILVER' or syminfo.ticker == 'XAUUSD' or syminfo.ticker == 'GOLD') ? profit\_commo : (syminfo.ticker == 'BTCUSD' or syminfo.ticker == 'ETHUSD') ? profit\_crypto : profit\_forex

theLoss = 0.0

foundPips = 0

// for p = 1 to 1000

//     prof =  acc\_currency == "USD" ?  p \* (10 \* lots) / close \* baseusd \* jpyFixer

//      : acc\_currency == "JPY" ?  p \* (10 \* lots) / close \* baseusd \* jpyFixer / usdjpy\_rate

//      : na

//     if prof > theLoss and prof <= loss

//         theLoss := prof

//         foundPips := p

var lab = label.new(0 ,0, "")

var **table** riskTable = table.new(position.bottom\_right , 1, 3, bgcolor = bgcolor, frame\_width = 4, border\_width = 0)

if barstate.islast

    str1 = ""

    str2 = ""

    str1 := acc\_currency == "USD" ? str.tostring(profit) + "$ per " + str.tostring(pips) + " pips"

      : acc\_currency == "JPY" ? str.tostring(profit) + "JPY per " + str.tostring(pips) + " pips"

      : acc\_currency == "EUR" ? str.tostring(profit) + "EUR per " + str.tostring(pips) + " pips"

      : acc\_currency == "GBP" ? str.tostring(profit) + "GBP per " + str.tostring(pips) + " pips"

      : acc\_currency == "AUD" ? str.tostring(profit) + "AUD per " + str.tostring(pips) + " pips"

      : acc\_currency == "THB" ? str.tostring(profit) + "THB per " + str.tostring(pips) + " pips"

      : na

    // str2 := "Use " + str.tostring(foundPips) + ":" + str.tostring(foundPips \* ratio) + " pips for " + str.tostring(math.round(theLoss, 2)) + "$"

    table.cell(riskTable, 0, 0, str.tostring(lots) + " lot(s)", text\_color = fontcolor, text\_halign = text.align\_left,bgcolor = bgcolor)

    table.cell(riskTable, 0, 1, str1, text\_color = fontcolor, text\_halign = text.align\_left,bgcolor = bgcolor)

    // table.cell(riskTable, 0, 2, str2, text\_color = color.black, text\_halign = text.align\_left)

// if barstate.islast

//     label.delete(lab)

//     if compact == false

//         lab := label.new(bar\_index + 30, low, , textcolor=color.black, style=label.style\_label\_center)

//     else

//         lab := label.new(bar\_index + 30, low, str.tostring(profit) + "$   " + str.tostring(pips) + "p   " + str.tostring(lots / 100) + " l\n\n " + str.tostring(foundPips) + ":" + str.tostring(foundPips \* ratio) + "   " + str.tostring(math.round(theLoss, 2)) + "$  R:R", textcolor=color.black, style=label.style\_label\_center)

//     label.set\_color(lab, color.white)