//@version=5

indicator("Amoozesh-boors.com Pivot & Moving ", " Amoozesh-boors.com Pivot & Moving ", 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

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( #FFD700, "" , inline="MA #1")

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

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

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)

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

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)

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

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)

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

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"

kind = input.string(title="Type", defval="Traditional", options=[TRADITIONAL, FIBONACCI, WOODIE, CLASSIC, DEMARK, CAMARILLA])

pivot\_time\_frame = input.string(title="Pivots Timeframe", defval=MONTHLY, options=[AUTO, DAILY, WEEKLY, MONTHLY, QUARTERLY, YEARLY, BIYEARLY, TRIYEARLY, QUINQUENNIALLY, DECENNIALLY])

look\_back = input.int(title="Number of Pivots Back", defval=1, minval=1, maxval=5000)

is\_daily\_based = input.bool(title="Use Daily-based Values", 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="labels")

position\_labels = input.string("Left", "Labels Position", options=["Left", "Right"], group="labels")

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

var DEF\_COLOR =#ED9121

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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