//**@version=**5

indicator("Pivot Points Standard11", "manual lines", overlay=true, max\_lines\_count=500, max\_labels\_count=500)

AUTO = "Auto"

DAILY = "188m(1/2)"

WEEKLY = "125m(1/3)"

MONTHLY = "94m(1/4)"

QUARTERLY = "Quarterly"

YEARLY = "Yearly"

year13="13 Year"

year21="21 Year"

year34="34 Year"

year55="55 Year"

year89="89 Year"

year144="144 Year"

year987="987 Year"

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=AUTO, options=[AUTO, DAILY, WEEKLY, MONTHLY, QUARTERLY, YEARLY, year13,year21,year34,year55,year89,year144,year987])

look\_back = input.int(title="Number of Pivots Back", defval=15, 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")

show\_prices = input.bool(title="Show Prices", 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 = #FB8C00

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

var p = array.new\_float()

level = input.int(title="Levels", defval=7,step=1)

startn = input.int(title="Start N", defval=-2,step=1)

endn = input.int(title="End N", defval=2,step=1)

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

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

var pivotrange = 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])

var lines = array.new\_line()

var labels = array.new\_label()

draw\_label(i,yval, txt, txt\_color) =>//draw\_label(i, array.get(p, i), "P", p\_color)

    if (show\_labels or show\_prices) and not na(yval)

        display\_text = (show\_labels ? txt : "") + (show\_prices ? str.format(" ({0})", math.round\_to\_mintick(yval)) : "")

        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=yval, text=display\_text, textcolor=txt\_color, style=label\_style, color=#00000000, xloc=xloc.bar\_time))

// fibonacci(i,yval,col,linestyle)=>

//     array.push(lines, line.new(array.get(arr\_time, i), yval , array.get(arr\_time, i + 1), yval, color=col, xloc=xloc.bar\_time, width=1,style=linestyle))

addline(i,yval,col,linestyle)=>

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

draw\_line(i, pivot, col) =>//draw\_line(i, p, p\_color)

    if array.size(arr\_time) > 1

        prange=array.get(pivotrange,i)

        pivotval=array.get(pivot, i)

        valueArr0=array.new\_float()

        newArr=array.new\_float()

        for c=startn to endn by 1   //to find and store values

            val=pivotval+(prange\*c)

            array.push(valueArr0,val)

            draw\_label(i,val,"Q"+str.tostring(c),p\_color)

        for itr=0 to (array.size(valueArr0)-1) by 1 // to display valueArr0 values

            addline(i,array.get(valueArr0,itr),color.white,line.style\_solid)

        //level=7

        if(level!=0)

            for l=1 to level by 1

                linecol=color.white

                if l==1

                    linecol:=color.new(color.white,30)

                else if l==2

                    linecol:=color.new(color.white,50)

                else if l==3

                    linecol:=color.new(#4dd0e1,50)

                else if l==4

                    linecol:=color.new(#ec407a,45)

                else if l==5

                    linecol:=color.new(#9c27b0,60)

                else if l==6

                    linecol:=color.new(#c5df94,80)

                else if l==7

                    linecol:=color.new(#f77c80,80)

                for c=0 to (array.size(valueArr0)-2) by 1

                    val=(array.get(valueArr0,c)+array.get(valueArr0,c+1))\*0.5

                    array.push(newArr,val)

                    addline(i,val,linecol,line.style\_solid)

                valueArr0:=array.concat(valueArr0,newArr)

                array.sort(valueArr0,order.ascending)

                array.clear(newArr)

calc\_pivot() =>

    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(pivotrange,pivot\_range)

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 := "188"

    else if pivot\_time\_frame == WEEKLY

        resolution := "125"

    else if pivot\_time\_frame == MONTHLY

        resolution := "94"

    else if pivot\_time\_frame == QUARTERLY

        resolution := "6M"

    else if pivot\_time\_frame == YEARLY or pivot\_time\_frame == year13 or pivot\_time\_frame == year21 or pivot\_time\_frame == year34 or pivot\_time\_frame == year55 or pivot\_time\_frame == year89 or pivot\_time\_frame == year144 or pivot\_time\_frame == year987

        resolution := "12M"

    resolution

resolution = get\_pivot\_resolution()

SIMPLE\_DIVISOR = -1

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

    year13 => 13

    year21 => 21

    year34 => 34

    year55 => 55

    year89 => 89

    year144 => 144

    year987 => 987

    => 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)

    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)