using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Threading;

using PowerLanguage.Function;

using ATCenterProxy.interop;

using PowerLanguage.Indicator;

using PowerLanguage.VolumeProfile;

namespace PowerLanguage.Strategy

{

[IOGMode(IOGMode.Enabled)]

public class \_RVPIBT : SignalObject

{

[Input]

public DirectionType Direction { get; set; }

[Input]

public OrderType Order { get; set; }

[Input]

public string TickEntryEndTime { get; set; }

[Input]

public bool TickEntry { get; set; }

[Input]

public bool TickExit { get; set; }

[Input]

public int TickTouchEntry { get; set; }

[Input]

public bool IBEntry { get; set; }

[Input]

public bool IBExit { get; set; }

[Input]

public int AtrLength { get; set; }

[Input]

public bool AtrEntry { get; set; }

[Input]

public string VolumeProfileEntryTime { get; set; }

[Input]

public bool VolumeProfileEntry { get; set; }

[Input]

public bool VolumeProfileExit { get; set; }

[Input]

public bool CloseOnEOD { get; set; }

[Input]

public bool UseExpertCommentary { get; set; }

[Input]

public string TradingStartTime { get; set; }

[Input]

public string TradingEndTime { get; set; }

[Input]

public string TradeEntryEndTime { get; set; }

public enum DirectionType

{

LongShort,

Long,

Short

}

public enum OrderType

{

Limit,

MarketThisBar

}

private class VolumeProfileValues

{

public double POC { get; set; }

public double VAL { get; set; }

public double VAH { get; set; }

public double VAR { get; set; }

}

private Dictionary<EOrderAction, IOrderMarket> \_marketOrder;

private Dictionary<EOrderAction, IOrderPriced> \_stopOrder;

private Dictionary<EOrderAction, IOrderPriced> \_limitOrder;

private bool \_isBuyEnabled, \_isSellEnabled;

private double \_tickSize;

private int \_dailyDataStream;

private IInstrument \_djtic, \_daily;

private TimeSpan \_tradingStartTime, \_tradingEndTime, \_volumeProfileEntryTime, \_tradeEntryEndTime, \_tickEntryEndTime;

private IDataLoaderResult \_dataLoaderResult;

private bool \_isSubscribed;

private int \_tickOverboughtCounter, \_tickOversoldCounter;

private VariableSeries<bool> \_isTradingTime;

private double? \_initialBalanceHigh, \_initialBalanceLow;

private double? \_atr, m\_atr;

public double? \_pt, \_sl, \_ptf, \_slf;

private VariableObject<Int32> m\_counter;

private VariableObject<Double> m\_yestopen;

private VariableObject<Double> m\_todaysopen;

private VariableObject<Double> m\_yesthigh;

private VariableObject<Double> m\_todayshigh;

private VariableObject<Double> m\_yestlow;

private VariableObject<Double> m\_todayslow;

private VariableObject<Double> m\_yestclose;

private VolumeProfileValues[] \_volumeProfile;

public \_RVPIBT(object ctx) : base(ctx)

{

Direction = DirectionType.LongShort;

Order = OrderType.Limit;

TickTouchEntry = 2;

AtrLength = 10;

VolumeProfileEntryTime = "9:55";

TradingStartTime = "09:30";

TradingEndTime = "16:00";

TradeEntryEndTime = "15:40";

TickEntryEndTime = "14:00";

}

protected override void Create()

{

\_marketOrder = new Dictionary<EOrderAction, IOrderMarket>(4);

\_stopOrder = new Dictionary<EOrderAction, IOrderPriced>(4);

\_limitOrder = new Dictionary<EOrderAction, IOrderPriced>(4);

m\_counter = new VariableObject<Int32>(this);

m\_yestopen = new VariableObject<Double>(this);

m\_todaysopen = new VariableObject<Double>(this);

m\_yesthigh = new VariableObject<Double>(this);

m\_todayshigh = new VariableObject<Double>(this);

m\_yestlow = new VariableObject<Double>(this);

m\_todayslow = new VariableObject<Double>(this);

m\_yestclose = new VariableObject<Double>(this);

foreach (var orderAction in new[] { EOrderAction.Buy, EOrderAction.Sell, EOrderAction.BuyToCover, EOrderAction.SellShort })

{

\_marketOrder.Add(orderAction, OrderCreator.MarketThisBar(new SOrderParameters(Contracts.Default, orderAction)));

\_stopOrder.Add(orderAction, OrderCreator.Stop(new SOrderParameters(Contracts.Default, orderAction)));

\_limitOrder.Add(orderAction, OrderCreator.Limit(new SOrderParameters(Contracts.Default, orderAction)));

}

\_isTradingTime = new VariableSeries<bool>(this);

}

protected override void StartCalc()

{

\_isBuyEnabled = true;

\_isSellEnabled = true;

if (Direction == DirectionType.Long)

\_isSellEnabled = false;

if (Direction == DirectionType.Short)

\_isBuyEnabled = false;

\_tickSize = Bars.Info.MinMove / Bars.Info.PriceScale;

\_volumeProfileEntryTime = TimeSpan.Parse(VolumeProfileEntryTime);

\_tradingStartTime = TimeSpan.Parse(TradingStartTime);

\_tradingEndTime = TimeSpan.Parse(TradingEndTime);

\_tradeEntryEndTime = TimeSpan.Parse(TradeEntryEndTime);

\_volumeProfile = new VolumeProfileValues[2];

\_tickEntryEndTime = TimeSpan.Parse(TickEntryEndTime);

for (var dataStream = 1; dataStream <= MaxDataStream; dataStream++)

{

var bars = BarsOfData(dataStream);

if (bars == null || bars == Bars)

continue;

var request = bars.Request;

if (request.Symbol == "DJTIC")

{

\_djtic = bars;

}

else if (request.Symbol == Bars.Request.Symbol)

{

var resolution = request.Resolution;

if (resolution.Type == EResolution.Day && resolution.Size == 1)

{

\_daily = bars;

\_dailyDataStream = dataStream;

}

}

}

if (\_daily == null)

throw new Exception("Daily data missing");

}

protected override void CalcBar()

{

if (Bars.CurrentBar < 1)

return;

OnTick();

if (Bars.Status == EBarState.Close)

{

OnBar();

}

{

if (Bars.Time[0].Date != Bars.Time[1].Date)

{

m\_counter.Value = (m\_counter.Value + 1);

m\_yestopen.Value = m\_todaysopen.Value;

m\_yesthigh.Value = m\_todayshigh.Value;

m\_yestlow.Value = m\_todayslow.Value;

m\_yestclose.Value = Bars.Close[1];

m\_todaysopen.Value = Bars.Open[0];

m\_todayshigh.Value = Bars.High[0];

m\_todayslow.Value = Bars.Low[0];

}

else{

if (PublicFunctions.DoubleGreater(Bars.High[0], m\_todayshigh.Value)){

m\_todayshigh.Value = Bars.High[0];

}

if (PublicFunctions.DoubleLess(Bars.Low[0], m\_todayslow.Value)){

m\_todayslow.Value = Bars.Low[0];

}

}

}

}

private string ToString(double? val)

{

return val.HasValue ? val.Value.ToString() : "N/A";

}

private VolumeProfileValues GetVolumeProfileValues(int barIndex)

{

var volumeProfile = VolumeProfile.ItemForBar(barIndex);

if (volumeProfile == null)

return \_volumeProfile[0];

var POCPrice = volumeProfile.POCForBar(barIndex);

var VAHPrice = volumeProfile.HighVAForBar(barIndex);

var VALPrice = volumeProfile.LowVAForBar(barIndex);

if (POCPrice == null)

Output.WriteLine("GetVolumeProfileValues: Volume Profile NoData, barIndex= {0}, DateTime= {1}", barIndex, Bars.Time.Value);

var volumeProfileValues = new VolumeProfileValues

{

POC = (POCPrice == null) ? 0 : POCPrice.Dbl,

VAH = (VAHPrice == null) ? 0 : VAHPrice.Dbl,

VAL = (VALPrice == null) ? 0 : VALPrice.Dbl,

};

volumeProfileValues.VAR = volumeProfileValues.VAH - volumeProfileValues.VAL;

return volumeProfileValues;

}

private void OnTick()

{

if (\_djtic == null)

return;

var tickHigh = \_djtic.HighValue;

var tickLow = \_djtic.LowValue;

var tick = \_djtic.CloseValue;

var close = Bars.CloseValue;

\_volumeProfile[0] = GetVolumeProfileValues(Bars.FullSymbolData.Current - 1);

m\_atr = this.AverageTrueRange(AtrLength);

if (TickExit)

{

var marketPosition = StrategyInfo.MarketPosition;

if (marketPosition != 0)

{

if (marketPosition > 0 && tickHigh >= 200 + (m\_atr \* 4.5 \* TickTouchEntry))

\_marketOrder[EOrderAction.Sell].Send("Tick LX");

if (marketPosition < 0 && tickLow <= -200 - (m\_atr \* 3.5 \* TickTouchEntry))

\_marketOrder[EOrderAction.BuyToCover].Send("Tick SX");

}

}

if (TickEntry)

{

if (tick >= 360 + (m\_atr \* 9 \* TickTouchEntry))

\_tickOverboughtCounter++;

if (tick <= -280 - (m\_atr \* 7 \* TickTouchEntry))

\_tickOversoldCounter++;

if (\_isBuyEnabled && \_tickOversoldCounter >= TickTouchEntry && close < \_volumeProfile[0].POC - (m\_atr \* 1.5) && Bars.TimeValue.TimeOfDay < \_tickEntryEndTime)

{

\_tickOversoldCounter = 0;

\_marketOrder[EOrderAction.Buy].Send("Tick LE");

}

if (\_isSellEnabled && \_tickOverboughtCounter >= TickTouchEntry && close > \_volumeProfile[0].POC + (m\_atr \* 1.5) && Bars.TimeValue.TimeOfDay < \_tickEntryEndTime)

{

\_tickOverboughtCounter = 0;

\_marketOrder[EOrderAction.SellShort].Send("Tick SE");

}

}

}

private void OnBar()

{

\_isTradingTime.Value = \_tradingStartTime <= Bars.TimeValue.TimeOfDay && Bars.TimeValue.TimeOfDay < \_tradingEndTime;

if (IBEntry || IBExit || VolumeProfileEntry || VolumeProfileExit || TickEntry || TickExit)

{

\_volumeProfile[0] = GetVolumeProfileValues(Bars.FullSymbolData.Current - 1);

}

CheckForExit();

CheckForEntry();

if (Bars.LastBarInSession)

{

OnEndOfDay();

}

UpdateExpertCommentary();

}

private void OnEndOfDay()

{

if (AtrEntry)

{

\_atr = this.AverageTrueRange(AtrLength, 0, \_dailyDataStream);

}

if (IBEntry)

{

\_initialBalanceHigh = null;

\_initialBalanceLow = null;

}

if (VolumeProfileEntry || VolumeProfileExit)

{

\_volumeProfile[1] = \_volumeProfile[0];

}

if (TickEntry || TickExit)

{

\_volumeProfile[1] = \_volumeProfile[0];

}

}

private void UpdateExpertCommentary()

{

try

{

if (!UseExpertCommentary)

return;

if (TickEntry)

{

ExpertCommentary.WriteLine("Tick Overbought Touches = {0}", \_tickOverboughtCounter);

ExpertCommentary.WriteLine("Tick Oversold Touches = {0}", \_tickOversoldCounter);

ExpertCommentary.WriteLine("Tick High = {0}", \_djtic.HighValue);

ExpertCommentary.WriteLine("Tick Low = {0}", \_djtic.LowValue);

}

if (IBEntry)

{

ExpertCommentary.WriteLine("IB High = {0}", ToString(\_initialBalanceHigh));

ExpertCommentary.WriteLine("IB Low = {0}", ToString(\_initialBalanceLow));

ExpertCommentary.WriteLine("IB Range = {0}", ToString(\_initialBalanceHigh - \_initialBalanceLow));

ExpertCommentary.WriteLine("IB High Range = {0}", ToString(\_initialBalanceHigh - \_volumeProfile[0].POC));

ExpertCommentary.WriteLine("IB Low Range = {0}", ToString(\_volumeProfile[0].POC - \_initialBalanceLow));

}

if (AtrEntry)

{

ExpertCommentary.WriteLine("Daily ATR = {0}", ToString(\_atr));

}

if (VolumeProfileEntry || VolumeProfileExit)

{

if (\_volumeProfile != null && \_volumeProfile[1] != null)

{

ExpertCommentary.WriteLine("Today's VAH = {0}", ToString(\_volumeProfile[0].VAH));

ExpertCommentary.WriteLine("Today's POC = {0}", ToString(\_volumeProfile[0].POC));

ExpertCommentary.WriteLine("Today's VAL = {0}", ToString(\_volumeProfile[0].VAL));

ExpertCommentary.WriteLine("POC Change = {0}", ToString(\_volumeProfile[0].POC - \_volumeProfile[1].POC));

ExpertCommentary.WriteLine("Yesterday's VAH = {0}", ToString(\_volumeProfile[1].VAH));

ExpertCommentary.WriteLine("Yesterday's POC = {0}", ToString(\_volumeProfile[1].POC));

ExpertCommentary.WriteLine("Yesterday's VAL = {0}", ToString(\_volumeProfile[1].VAL));

ExpertCommentary.WriteLine("Today's VAH Range = {0}", ToString(\_volumeProfile[0].VAH - \_volumeProfile[0].POC));

ExpertCommentary.WriteLine("Today's VAL Range = {0}", ToString(\_volumeProfile[0].POC - \_volumeProfile[0].VAL));

ExpertCommentary.WriteLine("Today's VA Range = {0}", ToString(\_volumeProfile[0].VAR));

ExpertCommentary.WriteLine("Open Gap = {0}", ToString(m\_todaysopen.Value - m\_yestclose.Value));

ExpertCommentary.WriteLine("Today's Open = {0}", ToString(m\_todaysopen.Value));

ExpertCommentary.WriteLine("Today's High = {0}", ToString(m\_todayshigh.Value));

ExpertCommentary.WriteLine("Today's Low = {0}", ToString(m\_todayslow.Value));

ExpertCommentary.WriteLine("Yesterday's Close = {0}", ToString(m\_yestclose.Value));

ExpertCommentary.WriteLine("Bar Close = {0}", ToString(Bars.CloseValue));

ExpertCommentary.WriteLine("3m ATR = {0}", ToString(m\_atr));

ExpertCommentary.WriteLine("Daily ATR = {0}", ToString(\_atr));

ExpertCommentary.WriteLine("Profit Target = {0}", ToString(\_pt));

ExpertCommentary.WriteLine("Stop Loss = {0}", ToString(\_sl));

ExpertCommentary.WriteLine("Profit Target Max = {0}", ToString(\_ptf));

ExpertCommentary.WriteLine("Stop Loss Max = {0}", ToString(\_slf));

}

}

}

catch (Exception ex)

{

Output.WriteLine("UpdateExpertCommentary Exception: {0}", ex.Message);

}

}

private void CheckForEntry()

{

var time = Bars.Time;

var open = Bars.OpenValue;

var close = Bars.CloseValue;

var openGap = m\_todaysopen.Value - m\_yestclose.Value;

// Average True Range Entry for Future Open Gap Fill Trade

if (\_isTradingTime[0] && AtrEntry && \_atr.HasValue && time[0].TimeOfDay < \_tradeEntryEndTime)

{

if (\_isBuyEnabled && close <= m\_yestclose.Value - \_atr \* (\_atr/100) && close < \_volumeProfile[0].POC - \_volumeProfile[0].VAR \* \_atr / 10)

\_marketOrder[EOrderAction.Buy].Send("ATR LE");

if (\_isSellEnabled && close >= m\_yestclose.Value + \_atr \* (\_atr/100) && close > \_volumeProfile[0].POC + \_volumeProfile[0].VAR \* \_atr / 10)

\_marketOrder[EOrderAction.SellShort].Send("ATR SE");

}

// Initial Balance Entry

var ibTradeTime = Bars.Time[0].Date.Add(\_tradingStartTime).AddHours(1);

var isIbEntryTime = Bars.Time[1] < ibTradeTime && Bars.Time[0] >= ibTradeTime;

if (IBEntry && isIbEntryTime && \_volumeProfile[0] != null)

{

var initialStart = time[0].Date.Add(\_tradingStartTime);

\_initialBalanceHigh = double.MinValue;

\_initialBalanceLow = double.MaxValue;

for (var i = 1; i < Bars.CurrentBar; i++)

{

if (time[i] <= initialStart)

break;

\_initialBalanceHigh = Math.Max(Bars.High[i], \_initialBalanceHigh.Value);

\_initialBalanceLow = Math.Min(Bars.Low[i], \_initialBalanceLow.Value);

}

var ibRange = \_initialBalanceHigh - \_initialBalanceLow;

var ibLRange = \_volumeProfile[0].POC - \_initialBalanceLow;

var ibHRange = \_initialBalanceHigh - \_volumeProfile[0].POC;

var ihVAR = \_volumeProfile[0].VAH - \_volumeProfile[0].POC;

var ilVAR = \_volumeProfile[0].POC -\_volumeProfile[0].VAL;

var iHE = \_initialBalanceHigh - \_ptf;

var iLE = \_initialBalanceLow + \_ptf;

var TPOC = \_volumeProfile[0].POC;

var TVAH = \_volumeProfile[0].VAH;

var TVAL = \_volumeProfile[0].VAL;

var YPOC = \_volumeProfile[1].POC;

var YVAH = \_volumeProfile[1].VAH;

var YVAL = \_volumeProfile[1].VAL;

var POCCh = \_volumeProfile[0].POC - \_volumeProfile[1].POC;

var TO = m\_todaysopen.Value;

var YC = m\_yestclose.Value;

// Long Entry

if (\_isBuyEnabled && (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC < YPOC && Math.Abs(openGap) > Math.Abs(POCCh) && TVAL > YVAL) || (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC < YPOC && openGap < 0 && TVAL > YVAL && TO > TPOC)|| (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC < YPOC && Math.Abs(openGap) > Math.Abs(POCCh) && TVAL < YVAL) || (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC < YPOC && openGap/TO > 0.015 && openGap < 0 && Math.Abs(openGap) < Math.Abs(POCCh) && TVAL < YVAL && TO < TPOC) || (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC < YPOC && openGap < 0 && Math.Abs(openGap) < Math.Abs(POCCh) && TVAL < YVAL && TO < TPOC && TVAH < YVAL) || (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC < YPOC && openGap < 0 && Math.Abs(openGap) < Math.Abs(POCCh) && TVAL < YVAL && TO > TPOC && TVAH < YVAL & Math.Abs(openGap/TO) > 0.05))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC > YPOC && TO > TPOC))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR < ilVAR && ibHRange < ibLRange && TO > YC && TPOC > YPOC && TO < TPOC) || (ihVAR < ilVAR && ibHRange < ibLRange && TO > YC && TPOC > YPOC && Math.Abs(openGap) > Math.Abs(POCCh) && TVAL > YVAL))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR < ilVAR && ibHRange > ibLRange && TO < YC && TPOC > YPOC))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR < ilVAR && ibHRange > ibLRange && TO > YC && TPOC < YPOC))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR > ilVAR && ibHRange < ibLRange && TO > YC && TPOC < YPOC))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR > ilVAR && ibHRange < ibLRange && TO > YC && TPOC > YPOC))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR > ilVAR && ibHRange > ibLRange && TO < YC && TPOC < YPOC && Math.Abs(openGap) < Math.Abs(POCCh)&& TVAL < YVAL && TO > TPOC && TVAH < YVAL && Math.Abs(openGap) > 60) || (ihVAR > ilVAR && ibHRange > ibLRange && TO < YC && TPOC < YPOC && Math.Abs(openGap) < Math.Abs(POCCh) && TVAL > YVAL && TO > TPOC) || (ihVAR > ilVAR && ibHRange > ibLRange && TO < YC && TPOC < YPOC && Math.Abs(openGap) < Math.Abs(POCCh) && TVAL < YVAL && TO < TPOC && openGap > -1))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR > ilVAR && ibHRange > ibLRange && TO < YC && TPOC > YPOC && TVAL < YVAL && Math.Abs(openGap) < Math.Abs(POCCh) && TO > TPOC))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR > ilVAR && ibHRange > ibLRange && TO > YC && TPOC < YPOC)) //&& Math.Abs(openGap) < Math.Abs(POCCh))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

if (\_isBuyEnabled && (ihVAR > ilVAR && ibHRange > ibLRange && TO > YC && TPOC > YPOC && Math.Abs(openGap) < Math.Abs(POCCh) && TVAL > YVAL && TO < TPOC) || (ihVAR > ilVAR && ibHRange > ibLRange && TO > YC && TPOC > YPOC && Math.Abs(openGap) > Math.Abs(POCCh) && TVAL > YVAL && TO < TPOC) || (ihVAR > ilVAR && ibHRange > ibLRange && TO > YC && TPOC > YPOC && Math.Abs(openGap) < Math.Abs(POCCh) && openGap/TO < 0.003) || (ihVAR > ilVAR && ibHRange > ibLRange && TO > YC && TPOC > YPOC && Math.Abs(openGap) > Math.Abs(POCCh) && TVAL > YVAL && TO > TPOC && openGap/TO < 0.018))

\_marketOrder[EOrderAction.Buy].Send("IB LE");

// Short Entry

if (\_isSellEnabled && (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC < YPOC && TVAL < YVAL))

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

if (\_isSellEnabled && (ihVAR < ilVAR && ibHRange < ibLRange && TO < YC && TPOC > YPOC && TO < TPOC))

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

if (\_isSellEnabled && (ihVAR < ilVAR && ibHRange < ibLRange && TO > YC && TPOC < YPOC))

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

if (\_isSellEnabled && (ihVAR < ilVAR && ibHRange < ibLRange && TO > YC && TPOC > YPOC && TO > TPOC))

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

if (\_isSellEnabled && (ihVAR < ilVAR && ibHRange > ibLRange && TO < YC && TPOC < YPOC) && (Math.Abs(openGap) <= Math.Abs(POCCh) && TVAL < YVAL))

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

if (\_isSellEnabled && (ihVAR > ilVAR && ibHRange < ibLRange && TO < YC && TPOC < YPOC))

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

if (\_isSellEnabled && (ihVAR > ilVAR && ibHRange > ibLRange && TO < YC && TPOC < YPOC && TVAL < YVAL && TVAH < YVAL) || (ihVAR > ilVAR && ibHRange > ibLRange && TO < YC && TPOC < YPOC && TVAL < YVAL && TVAH > YVAL))

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

if (\_isSellEnabled && ihVAR > ilVAR && ibHRange > ibLRange && TO < YC && TPOC > YPOC && TVAL > YVAL && TO > TPOC)

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

if (\_isSellEnabled && (ihVAR > ilVAR && ibHRange > ibLRange && TO > YC && TPOC > YPOC && (Math.Abs(openGap) < Math.Abs(POCCh) && TVAL > YVAL && TO > TPOC) ||(ihVAR > ilVAR && ibHRange > ibLRange && TO > YC && TPOC > YPOC && openGap > POCCh && TVAL > YVAL && TO > TPOC)))

\_marketOrder[EOrderAction.SellShort].Send("IB SE");

}

// Volume Profile Entry

if (VolumeProfileEntry && \_volumeProfile != null && \_volumeProfile[0] != null && time[0].TimeOfDay < \_tradeEntryEndTime)

{

if (time[0].TimeOfDay < \_volumeProfileEntryTime)

return;

\_atr = this.AverageTrueRange(AtrLength, 0, \_dailyDataStream);

var hRange = \_volumeProfile[0].VAH - \_volumeProfile[0].POC;

var lRange = \_volumeProfile[0].POC -\_volumeProfile[0].VAL;

var TPOC = \_volumeProfile[0].POC;

var TVAH = \_volumeProfile[0].VAH;

var TVAL = \_volumeProfile[0].VAL;

var YPOC = \_volumeProfile[1].POC;

var YVAH = \_volumeProfile[1].VAH;

var YVAL = \_volumeProfile[1].VAL;

var POCCh = \_volumeProfile[0].POC - \_volumeProfile[1].POC;

var TO = m\_todaysopen.Value;

var YC = m\_yestclose.Value;

var YH = m\_yesthigh.Value;

var YL = m\_yestlow.Value;

// Long Conditions

if (\_isBuyEnabled && close < \_volumeProfile[0].VAL - (m\_atr) && openGap < \_volumeProfile[0].VAR && \_volumeProfile[0].POC > \_volumeProfile[1].POC && time[0].TimeOfDay > \_volumeProfileEntryTime)

\_marketOrder[EOrderAction.Buy].Send("VP LE");

if (\_isBuyEnabled && close > \_volumeProfile[1].VAL && close < \_volumeProfile[0].POC -(m\_atr \*2) && hRange < lRange && m\_todaysopen.Value < \_volumeProfile[0].POC && time[0].TimeOfDay > \_volumeProfileEntryTime && close < \_volumeProfile[0].VAH )

\_marketOrder[EOrderAction.Buy].Send("VP LE");

if (\_isBuyEnabled && close > \_volumeProfile[1].VAH && close < \_volumeProfile[0].VAL - (m\_atr) && m\_todaysopen.Value < \_volumeProfile[0].POC && time[0].TimeOfDay > \_volumeProfileEntryTime && close < \_volumeProfile[0].VAH )

\_marketOrder[EOrderAction.Buy].Send("VP LE");

if (\_isBuyEnabled && close > YVAL && TO > YPOC && close < TPOC)

\_marketOrder[EOrderAction.Buy].Send("VP LE");

// Short Conditions

if (\_isSellEnabled && close > \_volumeProfile[0].VAH + (m\_atr) && \_volumeProfile[0].POC > \_volumeProfile[1].POC && close < \_volumeProfile[1].VAL && openGap < \_volumeProfile[0].VAR && time[0].TimeOfDay > \_volumeProfileEntryTime)

\_marketOrder[EOrderAction.SellShort].Send("VP SE");

if (\_isSellEnabled && close > \_volumeProfile[0].POC + (m\_atr \* 2.5) && \_volumeProfile[0].POC > \_volumeProfile[1].POC && hRange > lRange && m\_todaysopen.Value > \_volumeProfile[0].POC && openGap < \_volumeProfile[0].VAR && time[0].TimeOfDay > \_volumeProfileEntryTime)

\_marketOrder[EOrderAction.SellShort].Send("VP SE");

if (\_isSellEnabled && close > \_volumeProfile[0].POC + \_ptf && m\_todayshigh.Value < m\_yestlow.Value && time[0].TimeOfDay > \_volumeProfileEntryTime)

\_marketOrder[EOrderAction.SellShort].Send("VP SE");

//if (\_isSellEnabled && close < YH && TO < YPOC && close > TVAH)

// \_marketOrder[EOrderAction.SellShort].Send("VP SE");

}

}

private void CheckForExit()

{

\_pt = (2.5 + m\_atr) \* 4;

\_sl = (2.5 + m\_atr) \* 4;

\_ptf = 30;

\_slf = 30;

var marketPosition = StrategyInfo.MarketPosition;

if (marketPosition == 0)

return;

if (CloseOnEOD)

{

if (Bars.Time[0].TimeOfDay >= \_tradingEndTime)

{

if (marketPosition > 0)

\_marketOrder[EOrderAction.Sell].Send("EOD LX");

if (marketPosition < 0)

\_marketOrder[EOrderAction.BuyToCover].Send("EOD SX");

}

}

if (\_pt.HasValue && \_ptf.HasValue)

GenerateProfitTargetPt(Math.Min(\_pt.Value, \_ptf.Value));

if (\_sl.HasValue && \_slf.HasValue)

GenerateStopLossPt(Math.Min(\_sl.Value, \_slf.Value));

double ibStopLoss, vpStopLoss;

if (IBExit && \_initialBalanceHigh.HasValue && \_initialBalanceLow.HasValue)

ibStopLoss = marketPosition > 0 ? \_initialBalanceHigh.Value : \_initialBalanceLow.Value;

else

ibStopLoss = marketPosition > 0 ? double.MaxValue : double.MinValue;

if (VolumeProfileExit && \_volumeProfile[0] != null && m\_atr.HasValue)

vpStopLoss = marketPosition > 0 ? \_volumeProfile[0].POC + (2 + m\_atr.Value) : \_volumeProfile[0].POC - (2 + m\_atr.Value);

else

vpStopLoss = marketPosition > 0 ? double.MaxValue : double.MinValue;

var stopLoss = marketPosition > 0 ? Math.Min(ibStopLoss, vpStopLoss) : Math.Max(ibStopLoss, vpStopLoss);

if (stopLoss > double.MinValue && stopLoss < double.MaxValue)

GenerateStopLoss(stopLoss);

}

protected override void StopCalc()

{

base.StopCalc();

DataLoader.EndLoadData(\_dataLoaderResult);

}

}

}