

```
#include "sierrachart.h"
#include "scstudyfunctions.h"
```

```
/*=====*/
SCSFExport scsf_MarketStructure(SCStudyInterfaceRef sc)
{
    SCFloatArrayRef Array_High = sc.High;
    SCFloatArrayRef ArrayLow = sc.Low;
    SCFloatArrayRef Array_Open = sc.Open;
    SCFloatArrayRef Array_Close = sc.Close;

    SCSubgraphRef Subgraph_MSH = sc.Subgraph[0];
    SCSubgraphRef Subgraph_MSL = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Labels = sc.Subgraph[2];
    SCSubgraphRef Subgraph_BarTypes = sc.Subgraph[3];

    SCInputRef Input_DrawLabels = sc.Input[0];
    SCInputRef Input_LabelsOffset = sc.Input[1];
    SCInputRef Input_DisplayPriceValues = sc.Input[2];

    // Set configuration variables
    if (sc.SetDefaults)
    {

        sc.GraphName="Market Structure MSL/MSH";

        sc.GraphRegion = 0; // Use the main price graph region

        Subgraph_MSH.Name = "MSH";
        Subgraph_MSH.PrimaryColor = RGB(255,0,255);
        Subgraph_MSH.DrawStyle = DRAWSTYLE_COLOR_BAR;
        Subgraph_MSH.LineWidth = 2;
        Subgraph_MSH.DrawZeros = false;

        Subgraph_MSL.Name = "MSL";
        Subgraph_MSL.PrimaryColor = RGB(255, 128, 0);
        Subgraph_MSL.DrawStyle = DRAWSTYLE_COLOR_BAR;
        Subgraph_MSL.LineWidth = 2;
        Subgraph_MSL.DrawZeros = false;

        Subgraph_Labels.Name = "Labels";
        Subgraph_Labels.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;
        Subgraph_Labels.PrimaryColor = RGB(255, 0,255);
        Subgraph_Labels.SecondaryColor = RGB(255, 128, 0);
        Subgraph_Labels.LineWidth = 10;
        Subgraph_Labels.SecondaryColorUsed = true;
        Subgraph_Labels.DrawZeros = false;

        Input_DrawLabels.Name = "Draw Labels";
        Input_DrawLabels.SetYesNo(true);

        Input_LabelsOffset.Name = "Text Labels Percentage Offset";
        Input_LabelsOffset.SetFloat(5.0f);

        Input_DisplayPriceValues.Name = "Display Price Values";
        Input_DisplayPriceValues.SetYesNo(0);

        sc.AutoLoop          = 1;

        return;
    }
}
```

```

// define constants
enum MarketStructureValues
{
    NONE    = 0,
    HH      = 1,
    LH      = 2,
    LL      = 3,
    HL      = 4
};

enum BarTypeValues
{
    BT_DOWN,
    BT_UP,
    BT_NEUTRAL
};

const int UniqueID = 1546846579;

if(Array_Close[sc.Index] < Array_Open[sc.Index])
    Subgraph_BarTypes[sc.Index] = BT_DOWN;
else
    if(Array_Close[sc.Index] > Array_Open[sc.Index])
        Subgraph_BarTypes[sc.Index] = BT_UP;
    else
        Subgraph_BarTypes[sc.Index] = BT_NEUTRAL;

// we need at least 3 bars
if(sc.Index < 2)
    return;

sc.DataStartIndex = 2;

// Start calculations here
int CurrentIndex = sc.Index - 1;

if(Subgraph_BarTypes[CurrentIndex-2] == BT_DOWN &&
    Subgraph_BarTypes[CurrentIndex-1] == BT_DOWN &&
    Subgraph_BarTypes[CurrentIndex] == BT_UP)
{
    // these three bars can make MSL
    if(ArrayLow[CurrentIndex-2] > ArrayLow[CurrentIndex-1] &&
        ArrayLow[CurrentIndex] > ArrayLow[CurrentIndex-1])
    {
        // this is MSL
        Subgraph_MSL[CurrentIndex-1] = 1;
    }
}

if(Subgraph_BarTypes[CurrentIndex-2] == BT_UP &&
    Subgraph_BarTypes[CurrentIndex-1] == BT_UP &&
    Subgraph_BarTypes[CurrentIndex] == BT_DOWN)
{
    // these three bars can make MSH
    if(Array_High[CurrentIndex-2] < Array_High[CurrentIndex-1] &&
        Array_High[CurrentIndex] < Array_High[CurrentIndex-1])
    {
        // this is MSH
        Subgraph_MSH[CurrentIndex-1] = 1;
    }
}

// Fill Labels array

```

```

if(Subgraph_MSH[CurrentIndex-1] == 1)
{
    // look back through Labels array to find previous label HH or LH
    int Index=CurrentIndex-1;
    for(; Index >= 0; Index--)
    {
        if(Subgraph_Labels[Index] == HH || Subgraph_Labels[Index] == LH)
            break;
    }

    // now Index store index of the element
    if(Index < 0) // this is a first label, assume HH
        Subgraph_Labels[CurrentIndex-1] = HH;
    else
    {
        // compare High values
        if(Array_High[Index] == Array_High[CurrentIndex-1])
            Subgraph_Labels[CurrentIndex-1] = Subgraph_Labels[Index];
        else
            if(Array_High[Index] < Array_High[CurrentIndex-1])
                Subgraph_Labels[CurrentIndex-1] = HH;
            else
                Subgraph_Labels[CurrentIndex-1] = LH;
    }
}

```

```

if(Subgraph_MSL[CurrentIndex-1] == 1)
{
    // look back through Labels array to find previous label LL or HL
    int Index=CurrentIndex-1;
    for(; Index >= 0; Index--)
    {
        if(Subgraph_Labels[Index] == LL || Subgraph_Labels[Index] == HL)
            break;
    }

    // now Index store index of the element
    if(Index < 0) // this is a first label, assume LL
        Subgraph_Labels[CurrentIndex-1] = LL;
    else
    {
        // compare Low values
        if(ArrayLow[Index] == ArrayLow[CurrentIndex-1])
            Subgraph_Labels[CurrentIndex-1] = Subgraph_Labels[Index];
        else
            if(ArrayLow[Index] > ArrayLow[CurrentIndex-1])
                Subgraph_Labels[CurrentIndex-1] = LL;
            else
                Subgraph_Labels[CurrentIndex-1] = HL;
    }
}

```

```

// check if we need draw labels
if(!Input_DrawLabels.GetYesNo())
    return;

```

```

// if pattern is not set do not add the empty label
if(Subgraph_Labels[sc.Index-2] == 0)
    return;

```

// Since we are using UTAM_ADD_ALWAYS, we must not attempt to draw labels when we are on the last bar, even if the labels do not appear on the last bar. Otherwise, we get the same label added again and again as the bar is updated.

```

if(sc.GetBarHasClosedStatus(sc.Index) == BHCS_BAR_HAS_NOT_CLOSED)
    return;

s_UseTool Tool;
Tool.Clear(); // reset tool structure for our next use
Tool.ChartNumber = sc.ChartNumber;
Tool.DrawingType = DRAWING_TEXT;
Tool.Region = sc.GraphRegion;
Tool.FontFace = sc.ChartTextFont();
Tool.FontBold = true;

Tool.ReverseTextColor = 0;
Tool.FontSize = Subgraph_Labels.LineWidth;

Tool.LineNumber = UniqueID;
Tool.BeginIndex = sc.Index-2;

SCString Label;
SCString Value;

float Offset = Input_LabelsOffset.GetFloat()*0.01f*(sc.High[sc.Index - 2]-sc.Low[sc.Index - 2]);

switch(static_cast<int>(Subgraph_Labels[sc.Index-2]))
{
case HH:
    {
        if(!Input_DisplayPriceValues.GetYesNo())
        {
            Label = "HH";
        }
        else
        {
            Value = sc.FormatGraphValue(sc.High[sc.Index-2], sc.BaseGraphValueFormat);
            Label.Format("HH %s", Value.GetChars());
        }
        Tool.Text = Label;
        Tool.Color = Subgraph_Labels.PrimaryColor;

        Tool.BeginValue = sc.High[sc.Index-2] + Offset;
        Tool.TextAlignment = DT_CENTER | DT_BOTTOM;
        break;
    }

case LH:
    {
        if(!Input_DisplayPriceValues.GetYesNo())
        {
            Label = "LH";
        }
        else
        {
            Value = sc.FormatGraphValue(sc.High[sc.Index-2], sc.BaseGraphValueFormat);
            Label.Format("LH %s", Value.GetChars());
        }
        Tool.Text = Label;
        Tool.Color = Subgraph_Labels.PrimaryColor;

        Tool.BeginValue = sc.High[sc.Index-2] + Offset;
        Tool.TextAlignment = DT_CENTER | DT_BOTTOM;
        break;
    }

case LL:
    {

```

```

        if(!Input_DisplayPriceValues.GetYesNo())
        {
            Label = "LL";
        }
        else
        {
            Value = sc.FormatGraphValue(sc.Low[sc.Index-2], sc.BaseGraphValueFormat);
            Label.Format("LL %s", Value.GetChars());
        }
        Tool.Text      = Label;
        Tool.Color      = Subgraph_Labels.SecondaryColor;

        Tool.BeginValue = sc.Low[sc.Index-2] - Offset;
        Tool.TextAlignment = DT_CENTER | DT_TOP;
        break;
    }

case HL:
{
    if(!Input_DisplayPriceValues.GetYesNo())
    {
        Label = "HL";
    }
    else
    {
        Value = sc.FormatGraphValue(sc.Low[sc.Index-2], sc.BaseGraphValueFormat);
        Label.Format("HL %s", Value.GetChars());
    }
    Tool.Text      = Label;
    Tool.Color      = Subgraph_Labels.SecondaryColor;

    Tool.BeginValue = sc.Low[sc.Index-2] - Offset;
    Tool.TextAlignment = DT_CENTER | DT_TOP;
    break;
}

default:
    return;
}

Tool.AddMethod = UTAM_ADD_ALWAYS;

sc.UseTool(Tool);
}

/*=====*/
SCSFExport scsf_WoodiesPanel(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_CCIDiff = sc.Subgraph[0];
    SCSubgraphRef Subgraph_ProjectedBuy = sc.Subgraph[1];
    SCSubgraphRef Subgraph_ProjectedSell = sc.Subgraph[2];
    SCSubgraphRef Subgraph_CCIPred = sc.Subgraph[3];
    SCSubgraphRef Subgraph_BackColors = sc.Subgraph[4];
    SCSubgraphRef Subgraph_CCIDifferenceHigh = sc.Subgraph[5];
    SCSubgraphRef Subgraph_CCIDifferenceLow = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HighPC = sc.Subgraph[7];
    SCSubgraphRef Subgraph_LastPC = sc.Subgraph[8];
    SCSubgraphRef Subgraph_LowPC = sc.Subgraph[9];
    SCSubgraphRef Subgraph_EmaAngle = sc.Subgraph[10];

    SCInputRef Input_PanelRangeTopVal = sc.Input[0];
    SCInputRef Input_PanelRangeBottomVal = sc.Input[1];
    SCInputRef Input_ShowLabels = sc.Input[2];
    SCInputRef Input_ColumnsNumber = sc.Input[3];

```

```

SCInputRef Input_CCISStudyRef = sc.Input[4];
SCInputRef Input_EntryOffset = sc.Input[5];
SCInputRef Input_TradingStopTime = sc.Input[7];
SCInputRef Input_CCIPredictorStudyRef = sc.Input[8];
SCInputRef Input_ShowPreviousClose = sc.Input[9];
SCInputRef Input_SideWinderStudyRef = sc.Input[11];
SCInputRef Input_LeftEdgeTextOffset = sc.Input[12];

```

```

if (sc.SetDefaults)

```

```

{

```

```

    // Set the configuration and defaults

```

```

    sc.GraphName = "Woodies Panel";

```

```

    sc.AutoLoop = 0;

```

```

    sc.GraphRegion = 1;

```

```

    sc.CalculationPrecedence = LOW_PREC_LEVEL;

```

```

    Subgraph_CCIDiff.Name = "CCIDiff";

```

```

    Subgraph_CCIDiff.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;

```

```

    Subgraph_CCIDiff.PrimaryColor = RGB(255,255,255);

```

```

    Subgraph_CCIDiff.LineWidth = 10; //font size

```

```

    Subgraph_CCIDiff.DrawZeros = false;

```

```

    Subgraph_ProjectedBuy.Name = "Projected Buy";

```

```

    Subgraph_ProjectedBuy.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;

```

```

    Subgraph_ProjectedBuy.PrimaryColor = RGB(184,251,197);

```

```

    Subgraph_ProjectedBuy.LineWidth = 10; //font size

```

```

    Subgraph_ProjectedBuy.DrawZeros = false;

```

```

    Subgraph_ProjectedSell.Name = "Projected Sell";

```

```

    Subgraph_ProjectedSell.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;

```

```

    Subgraph_ProjectedSell.PrimaryColor = RGB(239,204,246);

```

```

    Subgraph_ProjectedSell.LineWidth = 10; //font size

```

```

    Subgraph_ProjectedSell.DrawZeros = false;

```

```

    Subgraph_CCIPred.Name = "CCI Pred. H/L";

```

```

    Subgraph_CCIPred.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;

```

```

    Subgraph_CCIPred.PrimaryColor = RGB(0,0,0);

```

```

    Subgraph_CCIPred.LineWidth = 10; //font size

```

```

    Subgraph_CCIPred.DrawZeros = false;

```

```

    Subgraph_BackColors.Name = "Background Colors";

```

```

    Subgraph_BackColors.DrawStyle = DRAWSTYLE_IGNORE;

```

```

    Subgraph_BackColors.PrimaryColor = RGB(64,128,128);

```

```

    Subgraph_BackColors.SecondaryColor = RGB(45,89,89);

```

```

    Subgraph_BackColors.SecondaryColorUsed = 1;

```

```

    Subgraph_BackColors.DrawZeros = false;

```

```

    Subgraph_CCIDifferenceHigh.Name = "CCIDiff H";

```

```

    Subgraph_CCIDifferenceHigh.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;

```

```

    Subgraph_CCIDifferenceHigh.PrimaryColor = RGB(0,255,0);

```

```

    Subgraph_CCIDifferenceHigh.LineWidth = 10; //font size

```

```

    Subgraph_CCIDifferenceHigh.DrawZeros = false;

```

```

    Subgraph_CCIDifferenceLow.Name = "CCIDiff L";

```

```

    Subgraph_CCIDifferenceLow.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;

```

```

    Subgraph_CCIDifferenceLow.PrimaryColor = RGB(237,80,255);

```

```

    Subgraph_CCIDifferenceLow.LineWidth = 10; //font size

```

```

    Subgraph_CCIDifferenceLow.DrawZeros = false;

```

```

    Subgraph_HighPC.Name = "High Prev/Cur";

```

```

    Subgraph_HighPC.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;

```

```

    Subgraph_HighPC.PrimaryColor = RGB(0,0,0);

```

```

    Subgraph_HighPC.SecondaryColor = RGB(0,255,0);

```

```

Subgraph_HighPC.SecondaryColorUsed = 1;
Subgraph_HighPC.LineWidth = 10; //font size
Subgraph_HighPC.DrawZeros = false;

Subgraph_LastPC.Name = "Last Prev/Cur";
Subgraph_LastPC.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;
Subgraph_LastPC.PrimaryColor = RGB(0,0,0);
Subgraph_LastPC.LineWidth = 14; //font size
Subgraph_LastPC.DrawZeros = false;

Subgraph_LowPC.Name = "Low Prev/Cur";
Subgraph_LowPC.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;
Subgraph_LowPC.PrimaryColor = RGB(0,0,0);
Subgraph_LowPC.SecondaryColor = RGB(255,0,0);
Subgraph_LowPC.SecondaryColorUsed = 1;
Subgraph_LowPC.LineWidth = 10; //font size
Subgraph_LowPC.DrawZeros = false;

Subgraph_EmaAngle.Name = "EMA Angle";
Subgraph_EmaAngle.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;
Subgraph_EmaAngle.PrimaryColor = RGB(0,0,0);
Subgraph_EmaAngle.LineWidth = 10; //font size
Subgraph_EmaAngle.DrawZeros = false;

Input_PanelRangeTopVal.Name = "Panel Range Top Value";
Input_PanelRangeTopVal.SetInt(250);

Input_PanelRangeBottomVal.Name = "Panel Range Bottom Value";
Input_PanelRangeBottomVal.SetInt(-250);

Input_ShowLabels.Name = "Show Descriptive Labels";
Input_ShowLabels.SetYesNo(true);

Input_ColumnsNumber.Name = "Number of Fill Space Columns";
Input_ColumnsNumber.SetInt(10);

Input_CCISudyRef.Name = "CCI Study Reference";
Input_CCISudyRef.SetStudyID(1);

Input_EntryOffset.Name = "Projected Entry Offset in Ticks";
Input_EntryOffset.SetInt(1);

Input_TradingStopTime.Name = "Trading Stop Time";
Input_TradingStopTime.SetTime(HMS_TIME(15, 30, 0));

Input_CCIPredictorStudyRef.Name = "CCI Predictor Study Reference";
Input_CCIPredictorStudyRef.SetStudyID(1);

Input_ShowPreviousClose.Name = "Show Previous Close";
Input_ShowPreviousClose.SetYesNo(false);

Input_SideWinderStudyRef.Name = "SideWinder Study Reference";
Input_SideWinderStudyRef.SetStudyID(1);

Input_LeftEdgeTextOffset.Name = "Left Edge Text Offset";
Input_LeftEdgeTextOffset.SetInt(1);
Input_LeftEdgeTextOffset.SetIntLimits(0,4);

sc.UpdateAlways = 1;

return;
}

if (sc.LastCallToFunction)

```

```

{
    sc.PreserveFillSpace = 0;
    sc.NumFillSpaceBars = 0;
    sc.UseGlobalChartColors = true;
    sc.ScaleBorderColor = RGB(128,128,128);

    return;
}

{
    sc.PreserveFillSpace = 1;
    sc.NumFillSpaceBars = Input_ColumnsNumber.GetInt();
}

int CurrentVisibleIndex = sc.IndexOfLastVisibleBar;

if (Subgraph_BackColors.PrimaryColor == RGB(255,0,0))
{
    Subgraph_BackColors.PrimaryColor = RGB(64,128,128);
    Subgraph_BackColors.SecondaryColor = RGB(45,89,89);
}

if (sc.GraphRegion >= 2)
    sc.GraphRegion = 1;

float RangeIncrement = 0.0f;

//Determine our value increment for the panel data lines based on what lines are visible.
int NumberOfLines = 0;
for(int c= 0; c<16; c++)
{
    if(sc.Subgraph[c].DrawStyle != DRAWSTYLE_IGNORE && sc.Subgraph[c].DrawStyle != DRAWSTYLE_HIDDEN)
        NumberOfLines++;
}

RangeIncrement = (Input_PanelRangeTopVal.GetInt() - Input_PanelRangeBottomVal.GetInt())/(NumberOfLines+1.0f);

float YValue = Input_PanelRangeTopVal.GetInt()-RangeIncrement;

int TextOffset = -(2+Input_LeftEdgeTextOffset.GetInt());

s_UseTool Tool;

if(Subgraph_CCIDifferenceHigh.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_CCIDifferenceHigh.DrawStyle !=
DRAWSTYLE_HIDDEN)
{

    SCFloatArray CCIProjHigh;
    sc.GetStudyArrayUsingID(Input_CCIPredictorStudyRef.GetStudyID(),0,CCIProjHigh);

    float CCIDifferenceHighValue = 0;
    SCFloatArray CCI;
    sc.GetStudyArrayUsingID(Input_CCIStudyRef.GetStudyID(),0,CCI);
    CCIDifferenceHighValue = CCIProjHigh[CurrentVisibleIndex] - CCI[CurrentVisibleIndex-1];

    if (CurrentVisibleIndex != sc.ArraySize-1)

```



```
CCIDifferenceHighValue = CCI[CurrentVisibleIndex] - CCI[CurrentVisibleIndex-1];
```

```
//Worksheet Output
```

```
Subgraph_CCIDifferenceHigh[sc.UpdateStartIndex] = 0;
```

```
Subgraph_CCIDifferenceHigh[sc.ArraySize - 1] = CCIDifferenceHighValue;
```

```
Tool.Clear(); // reset tool structure for our next use
```

```
Tool.ChartNumber = sc.ChartNumber;
```

```
Tool.DrawingType = DRAWING_TEXT;
```

```
Tool.LineNumber = 5035;
```

```
Tool.BeginDateTime = TextOffset;
```

```
Tool.BeginValue = YValue;
```

```
Tool.Color = Subgraph_CCIDifferenceHigh.PrimaryColor;
```

```
Tool.Region = sc.GraphRegion;
```

```
SCString Label;
```

```
if (Input_ShowLabels.GetYesNo())
```

```
{
```

```
    Label = Subgraph_CCIDifferenceHigh.Name;
```

```
}
```

```
Tool.Text.Format(" %.2f %s", CCIDifferenceHighValue, Label.GetChars());
```

```
Tool.FontSize = Subgraph_CCIDifferenceHigh.LineWidth;
```

```
Tool.FontBold = true;
```

```
//Tool.FontFace= "Courier";
```

```
Tool.AddMethod = UTAM_ADD_OR_ADJUST;
```

```
Tool.ReverseTextColor = 0;
```

```
sc.UseTool(Tool);
```

```
YValue -= RangeIncrement;
```

```
}
```

```
if(Subgraph_CCIDiff.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_CCIDiff.DrawStyle !=  
DRAWSTYLE_HIDDEN)
```

```
{
```

```
    float CCIDiffVal = 0;
```

```
    SCFloatArray CCI;
```

```
    sc.GetStudyArrayUsingID(Input_CCISTudyRef.GetStudyID(),0,CCI);
```

```
    CCIDiffVal = CCI[CurrentVisibleIndex] - CCI[CurrentVisibleIndex-1];
```

```
    Subgraph_CCIDiff[sc.UpdateStartIndex] = 0;
```

```
    Subgraph_CCIDiff[sc.ArraySize - 1] = CCIDiffVal;
```

```
Tool.Clear(); // reset tool structure for our next use
```

```
Tool.ChartNumber = sc.ChartNumber;
```

```
Tool.DrawingType = DRAWING_TEXT;
```

```
Tool.LineNumber = 5030;
```

```
Tool.BeginDateTime = TextOffset;
```

```
Tool.BeginValue = YValue;
```

```
Tool.Color = Subgraph_CCIDiff.PrimaryColor;
```

```
Tool.Region = sc.GraphRegion;
```

```
SCString Label;
```

```
if (Input_ShowLabels.GetYesNo())
```

```
{
```

```
    Label = "CCIDiff";
```

```
}
```

```
Tool.Text.Format(" %.2f %s", CCIDiffVal , Label.GetChars());
```

```
Tool.FontSize = Subgraph_CCIDiff.LineWidth;
```

```

Tool.FontBold = true;
Tool.AddMethod = UTAM_ADD_OR_ADJUST;
Tool.ReverseTextColor = 0;

sc.UseTool(Tool);
YValue -= RangeIncrement;
}

if(Subgraph_CCIDifferenceLow.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_CCIDifferenceLow.DrawStyle !=
DRAWSTYLE_HIDDEN)
{
    SCFloatArray CCIProjLow;
    sc.GetStudyArrayUsingID(Input_CCIPredictorStudyRef.GetStudyID(),1,CCIProjLow);

    float CCIDifferenceLowValue = 0;
    SCFloatArray CCI;
    sc.GetStudyArrayUsingID(Input_CCISubjectRef.GetStudyID(),0,CCI);
    CCIDifferenceLowValue = CCIProjLow[CurrentVisibleIndex] - CCI[CurrentVisibleIndex-1];

    if (CurrentVisibleIndex != sc.ArraySize-1)
        CCIDifferenceLowValue = CCI[CurrentVisibleIndex] - CCI[CurrentVisibleIndex-1];

    Subgraph_CCIDifferenceLow[sc.UpdateStartIndex] = 0;
    Subgraph_CCIDifferenceLow[sc.ArraySize - 1] = CCIDifferenceLowValue;

    Tool.Clear(); // reset tool structure for our next use
    Tool.ChartNumber = sc.ChartNumber;
    Tool.DrawingType = DRAWING_TEXT;
    Tool.LineNumber = 5036;
    Tool.BeginDateTime = TextOffset;
    Tool.BeginValue = YValue;
    Tool.Color = Subgraph_CCIDifferenceLow.PrimaryColor;
    Tool.Region = sc.GraphRegion;

    SCString Label;

    if (Input_ShowLabels.GetYesNo())
    {
        Label = Subgraph_CCIDifferenceLow.Name;
    }

    Tool.Text.Format(" %.2f %s", CCIDifferenceLowValue, Label.GetChars());
    Tool.FontSize = Subgraph_CCIDifferenceLow.LineWidth;
    Tool.FontBold = true;
    Tool.AddMethod = UTAM_ADD_OR_ADJUST;
    Tool.ReverseTextColor = 0;

    sc.UseTool(Tool);
    YValue -= RangeIncrement;
}

if(Subgraph_HighPC.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_HighPC.DrawStyle !=
DRAWSTYLE_HIDDEN)
{
    Tool.Clear(); // reset tool structure for our next use
    Tool.ChartNumber = sc.ChartNumber;
    Tool.DrawingType = DRAWING_TEXT;
    Tool.LineNumber = 5037;
    Tool.BeginDateTime = TextOffset;
    Tool.BeginValue = YValue;
    if (sc.High[CurrentVisibleIndex] > sc.High[CurrentVisibleIndex-1])
    {
        Tool.Color = Subgraph_HighPC.SecondaryColor;
    }
}

```

```

    Tool.ReverseTextColor = 0;
}
else
{
    Tool.Color = Subgraph_HighPC.PrimaryColor;
    Tool.ReverseTextColor = 0;
}

Tool.Region = sc.GraphRegion;

SCString Label;

if (Input_ShowLabels.GetYesNo())
{
    Label = Subgraph_HighPC.Name;
}

SCString CurrentHigh = sc.FormatGraphValue(sc.High[CurrentVisibleIndex], sc.BaseGraphValueFormat);
SCString PrevHigh = sc.FormatGraphValue(sc.High[CurrentVisibleIndex-1], sc.BaseGraphValueFormat);
Tool.Text.Format(" %s %s %s", PrevHigh.GetChars(), CurrentHigh.GetChars(), Label.GetChars());
Tool.FontSize = Subgraph_HighPC.LineWidth;
Tool.FontBold = true;
Tool.AddMethod = UTAM_ADD_OR_ADJUST;

sc.UseTool(Tool);
YValue -= RangeIncrement;
//Worksheet Output
Subgraph_HighPC[sc.UpdateStartIndex] = 0;
Subgraph_HighPC[sc.UpdateStartIndex-1] = 0;
Subgraph_HighPC[sc.ArraySize - 1] = sc.High[CurrentVisibleIndex];
Subgraph_HighPC[sc.ArraySize - 2] = sc.High[CurrentVisibleIndex-1];
}

if(Subgraph_LastPC.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_LastPC.DrawStyle !=
DRAWSTYLE_HIDDEN)
{
    Tool.Clear(); // reset tool structure for our next use
    Tool.ChartNumber = sc.ChartNumber;
    Tool.DrawingType = DRAWING_TEXT;
    Tool.LineNumber = 5038;
    Tool.BeginDateTime = TextOffset;
    Tool.BeginValue = YValue;

    {
        Tool.Color = Subgraph_LastPC.PrimaryColor;
        Tool.ReverseTextColor = 0;
    }

    Tool.Region = sc.GraphRegion;

    SCString Label;

    if (Input_ShowLabels.GetYesNo())
    {
        if(Input_ShowPreviousClose.GetYesNo())
            Label = Subgraph_LastPC.Name;
        else
            Label = "Last Price";
    }

    SCString CurLast = sc.FormatGraphValue(sc.Close[CurrentVisibleIndex], sc.BaseGraphValueFormat);
    SCString PrevLast = sc.FormatGraphValue(sc.Close[CurrentVisibleIndex-1], sc.BaseGraphValueFormat);

```

```

//Worksheet Output
Subgraph_LastPC[sc.UpdateStartIndex] = 0;
Subgraph_LastPC[sc.UpdateStartIndex-1] = 0;
Subgraph_LastPC[sc.ArraySize - 1] = sc.Close[CurrentVisibleIndex];
Subgraph_LastPC[sc.ArraySize - 2] = sc.Close[CurrentVisibleIndex-1];

if (Input_ShowPreviousClose.GetYesNo())
{
    Tool.Text.Format(" %s %s %s", PrevLast.GetChars(), CurLast.GetChars(), Label.GetChars());
}
else
{
    Tool.Text.Format(" %s %s", CurLast.GetChars(), Label.GetChars());
}

Tool.FontSize = Subgraph_LastPC.LineWidth;
Tool.FontBold = true;
Tool.AddMethod = UTAM_ADD_OR_ADJUST;

sc.UseTool(Tool);
YValue -= RangeIncrement;
}

if(Subgraph_LowPC.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_LowPC.DrawStyle !=
DRAWSTYLE_HIDDEN)
{
    Tool.Clear(); // reset tool structure for our next use
    Tool.ChartNumber = sc.ChartNumber;
    Tool.DrawingType = DRAWING_TEXT;
    Tool.LineNumber = 5039;
    Tool.BeginDateTime = TextOffset;
    Tool.BeginValue = YValue;
    if (sc.Low[CurrentVisibleIndex] < sc.Low[CurrentVisibleIndex-1])
    {
        Tool.Color = Subgraph_LowPC.SecondaryColor;
        Tool.ReverseTextColor = 0;
    }
    else
    {
        Tool.Color = Subgraph_LowPC.PrimaryColor;
        Tool.ReverseTextColor = 0;
    }

    Tool.Region = sc.GraphRegion;

    SCString Label;

    if (Input_ShowLabels.GetYesNo())
    {
        Label = Subgraph_LowPC.Name;
    }

    SCString CurLow = sc.FormatGraphValue(sc.Low[CurrentVisibleIndex], sc.BaseGraphValueFormat);
    SCString PrevLow = sc.FormatGraphValue(sc.Low[CurrentVisibleIndex-1], sc.BaseGraphValueFormat);

//Worksheet Output
Subgraph_LowPC[sc.UpdateStartIndex] = 0;
Subgraph_LowPC[sc.UpdateStartIndex-1] = 0;
Subgraph_LowPC[sc.ArraySize - 1] = sc.Low[CurrentVisibleIndex];
Subgraph_LowPC[sc.ArraySize - 2] = sc.Low[CurrentVisibleIndex-1];

Tool.Text.Format(" %s %s %s",PrevLow.GetChars(),CurLow.GetChars(),Label.GetChars());

```

```

Tool.FontSize = Subgraph_LowPC.LineWidth;
Tool.FontBold = true;
Tool.AddMethod = UTAM_ADD_OR_ADJUST;

sc.UseTool(Tool);
YValue -= RangeIncrement;
}

n_ACSIL::s_BarPeriod BarPeriod;
sc.GetBarPeriodParameters(BarPeriod);

if(Subgraph_ProjectedBuy.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_ProjectedBuy.DrawStyle !=
DRAWSTYLE_HIDDEN)
{
    float ProjHigh = 0;

    ProjHigh = sc.Low[CurrentVisibleIndex] + BarPeriod.IntradayChartBarPeriodParameter1 +
(Input_EntryOffset.GetInt() * sc.TickSize);

    //Worksheet Output
    Subgraph_ProjectedBuy[sc.UpdateStartIndex] = 0;
    Subgraph_ProjectedBuy[sc.ArraySize - 1] = ProjHigh;

    Tool.Clear(); // reset tool structure for our next use
    Tool.ChartNumber = sc.ChartNumber;
    Tool.DrawingType = DRAWING_TEXT;
    Tool.LineNumber = 5031;
    Tool.BeginDateTime = TextOffset;
    Tool.BeginValue = YValue;
    Tool.Color = Subgraph_ProjectedBuy.PrimaryColor;
    Tool.Region = sc.GraphRegion;

    SCString Label;

    if (Input_ShowLabels.GetYesNo())
    {
        Label = "ProjHigh";
    }

    SCString StrValue = sc.FormatGraphValue(ProjHigh, sc.BaseGraphValueFormat);
    Tool.Text.Format(" %s %s", StrValue.GetChars(), Label.GetChars());
    Tool.FontSize = Subgraph_ProjectedBuy.LineWidth;
    Tool.FontBold = true;
    Tool.AddMethod = UTAM_ADD_OR_ADJUST;
    Tool.ReverseTextColor = 0;

    sc.UseTool(Tool);
    YValue -= RangeIncrement;
}

if(Subgraph_ProjectedSell.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_ProjectedSell.DrawStyle !=
DRAWSTYLE_HIDDEN)
{
    float ProjLow = 0;

    ProjLow = sc.High[CurrentVisibleIndex] - (BarPeriod.IntradayChartBarPeriodParameter1 +
(Input_EntryOffset.GetInt() * sc.TickSize));

    //Spreadsheet Output
    Subgraph_ProjectedSell[sc.UpdateStartIndex] = 0;
    Subgraph_ProjectedSell[sc.ArraySize - 1] = ProjLow;

```

```

Tool.Clear(); // reset tool structure for our next use
Tool.ChartNumber = sc.ChartNumber;
Tool.DrawingType = DRAWING_TEXT;
Tool.LineNumber = 5032;
Tool.BeginDateTime = TextOffset;
Tool.BeginValue = YValue;
Tool.Color = Subgraph_ProjectedSell.PrimaryColor;
Tool.Region = sc.GraphRegion;

SCString Label;

if (Input_ShowLabels.GetYesNo())
{
    Label = "ProjLow";
}

SCString StrValue = sc.FormatGraphValue(ProjLow, sc.BaseGraphValueFormat);
Tool.Text.Format("%s %s", StrValue.GetChars(), Label.GetChars());
Tool.FontSize = Subgraph_ProjectedSell.LineWidth;
Tool.FontBold = true;
Tool.AddMethod = UTAM_ADD_OR_ADJUST;
Tool.ReverseTextColor = 0;

sc.UseTool(Tool);
YValue -= RangeIncrement;
}

if(Subgraph_CCIPred.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_CCIPred.DrawStyle !=
DRAWSTYLE_HIDDEN)
{

    SCFloatArray CCIProjHigh;
    SCFloatArray CCIProjLow;

    sc.GetStudyArrayUsingID(Input_CCIPredictorStudyRef.GetStudyID(),0,CCIProjHigh);
    sc.GetStudyArrayUsingID(Input_CCIPredictorStudyRef.GetStudyID(),1,CCIProjLow);


Tool.Clear(); // reset tool structure for our next use
Tool.ChartNumber = sc.ChartNumber;
Tool.DrawingType = DRAWING_TEXT;
Tool.LineNumber = 5033;
Tool.BeginDateTime = TextOffset;
Tool.BeginValue = YValue;
Tool.Color = Subgraph_CCIPred.PrimaryColor;
Tool.Region = sc.GraphRegion;

SCString Label;

if (Input_ShowLabels.GetYesNo())
{
    Label = Subgraph_CCIPred.Name;
}

float ProjHigh = CCIProjHigh[sc.ArraySize-1];
float ProjLow = CCIProjLow[sc.ArraySize-1];

if (CurrentVisibleIndex != sc.ArraySize-1)
{
    SCFloatArray CCI;
    sc.GetStudyArrayUsingID(Input_CCIStudyRef.GetStudyID(),0,CCI);
    ProjLow = ProjHigh = CCI[CurrentVisibleIndex];
}

```

```

Tool.Text.Format(" %.1f %.1f %s",ProjHigh,ProjLow,Label.GetChars());
Tool.FontSize = Subgraph_CCIPred.LineWidth;
Tool.FontBold = true;
Tool.AddMethod = UTAM_ADD_OR_ADJUST;
Tool.ReverseTextColor = 0;

sc.UseTool(Tool);
YValue -= RangeIncrement;

//Worksheet Output
Subgraph_CCIPred[sc.UpdateStartIndex] = 0;
Subgraph_CCIPred[sc.UpdateStartIndex-1] = 0;
Subgraph_CCIPred[sc.ArraySize - 1] = ProjHigh;
Subgraph_CCIPred[sc.ArraySize - 2] = ProjLow;
}

if(Subgraph_LowPC.DrawStyle != DRAWSTYLE_IGNORE && Subgraph_LowPC.DrawStyle !=
DRAWSTYLE_HIDDEN)
{
    SCFloatArray EMAAngle;
    sc.GetStudyArrayUsingID(Input_SideWinderStudyRef.GetStudyID(),7,EMAAngle);

    Tool.Clear(); // reset tool structure for our next use
    Tool.ChartNumber = sc.ChartNumber;
    Tool.DrawingType = DRAWING_TEXT;
    Tool.LineNumber = 5040;
    Tool.BeginDateTime = TextOffset;
    Tool.BeginValue = YValue;
    Tool.Color = Subgraph_LowPC.PrimaryColor;
    Tool.Region = sc.GraphRegion;

    SCString Label;

    if (Input_ShowLabels.GetYesNo())
    {
        Label = Subgraph_LowPC.Name;
    }

    Tool.Text.Format(" %.1fu00B0 %s", EMAAngle[CurrentVisibleIndex], Label.GetChars());
    Tool.FontSize = Subgraph_LowPC.LineWidth;
    Tool.FontBold = true;
    Tool.AddMethod = UTAM_ADD_OR_ADJUST;
    Tool.ReverseTextColor = 0;

    sc.UseTool(Tool);
    YValue -= RangeIncrement;

    //Worksheet Output
    Subgraph_LowPC[sc.UpdateStartIndex] = 0;
    Subgraph_LowPC[sc.ArraySize - 1] = EMAAngle[CurrentVisibleIndex];

}

}

/*=====*/
void AccumulateZigZagVolume(SCStudyInterfaceRef sc, SCSubgraphRef AccumulatedVolume, SCFloatArrayRef
ZigZagPeakType, int VolumeTypeToAccumulate, SCFloatArrayRef ZigZagReset)
{
    if (VolumeTypeToAccumulate == 0)
        return;

```

```

//Accumulate the volume across the last two zigzags

//Determine starting point
int Count = 0;//To count the number of zigzag lines
int StartingIndex = sc.UpdateStartIndex+1;
while (StartingIndex > 0 && Count < 2)
{
    StartingIndex--;
    if (ZigZagPeakType[StartingIndex] != 0.0f)
        Count ++;
}

double Volume = 0;
int Direction = 0;

for (int Index = StartingIndex+1; Index < sc.ArraySize; ++Index)
{
    if(ZigZagPeakType[Index-1] != 0)
    {
        Volume = 0;
        Direction = static_cast<int>(ZigZagPeakType[Index-1]);
    }

    if (ZigZagReset[Index-1] != 0) // First index, pick up initial volume
    {
        if (VolumeTypeToAccumulate == 1)
            Volume = sc.Volume[Index-1];
        else if (VolumeTypeToAccumulate == 2)
            Volume = sc.BidVolume[Index-1];
        else if (VolumeTypeToAccumulate == 3)
            Volume = sc.AskVolume[Index-1];
        else if (VolumeTypeToAccumulate == 4)
            Volume = sc.AskVolume[Index-1] - sc.BidVolume[Index-1];

        AccumulatedVolume[Index-1] = static_cast<float>(Volume);
        Direction = static_cast<int>(ZigZagPeakType[Index-1]);
    }

    if (Direction == 1)
        AccumulatedVolume.DataColor[Index]= AccumulatedVolume.PrimaryColor;
    else if (Direction == -1)
        AccumulatedVolume.DataColor[Index]= AccumulatedVolume.SecondaryColor;

    if (VolumeTypeToAccumulate == 1)
        Volume += sc.Volume[Index];
    else if (VolumeTypeToAccumulate == 2)
        Volume += sc.BidVolume[Index];
    else if (VolumeTypeToAccumulate == 3)
        Volume += sc.AskVolume[Index];
    else if (VolumeTypeToAccumulate == 4)
        Volume += sc.AskVolume[Index] - sc.BidVolume[Index];

    AccumulatedVolume[Index] = static_cast<float>(Volume);
}
}

/*=====*/
void CreateZigZagLineLabel(SCString &BarLabelText, SCFloatArrayRef ZigZagPeakType, const int ZigZagIndex,
SCFloatArrayRef ZigZagPeakIndex, SCInputRef ShowRevPrice, SCInputRef UseMultiLineLabels, SCStudyInterfaceRef
sc, SCSubgraphRef ZigZagLine, SCInputRef ShowLength, bool ShowLabelPrefixes, SCInputRef ShowTime, SCInputRef
IncludeSecondsInTimeDisplay, SCInputRef ShowZigZagVolume, SCSubgraphRef AccumulatedVolume, SCInputRef
FormatVolumeWithLargeNumberSuffix, SCInputRef ShowZigZagDuration, SCInputRef ShowZigZagNumBars,
SCInputRef DisplayZigZagVolumeDividedByLengthInTicks, SCFloatArrayRef ZigZagReset, SCInputRef
DisplayHHHLLLLHLabels, SCSubgraphRef TextLabels, double& ZigzagLineLength, int& CurrentZigZagNumBars)
{

```



```

int BackRefIndex = static_cast<int>(ZigZagPeakIndex[ZigZagIndex - 1]);

if (ZigZagPeakType[BackRefIndex] == -1 * ZigZagPeakType[ZigZagIndex])
{
    if (ShowRevPrice.GetYesNo())
    {
        if (BarLabelText.GetLength() != 0)
        {
            if (UseMultiLineLabels.GetYesNo())
                BarLabelText += "\n";
            else
                BarLabelText += " ";
        }

        SCString ReversalPrice;
        ReversalPrice = sc.FormatGraphValue(ZigZagLine[ZigZagIndex], sc.GetValueFormat());
        BarLabelText += ReversalPrice;
    }

    ZigzagLineLength = ZigZagLine[ZigZagIndex] - static_cast<double>(ZigZagLine[BackRefIndex]);
    if (ShowLength.GetIndex() != 0)
    {
        if (BarLabelText.GetLength() != 0)
        {
            if (UseMultiLineLabels.GetYesNo())
                BarLabelText += "\n";
            else
                BarLabelText += " ";
        }

        SCString LengthStr;
        if (ShowLabelPrefixes)
            LengthStr = "L.";

        if (ShowLength.GetIndex() == 1) // Points
            LengthStr += sc.FormatGraphValue(ZigzagLineLength, sc.GetValueFormat());
        else if (ShowLength.GetIndex() == 2) // Ticks
        {
            double Ticks = sc.TickSize == 0 ? 0 : ZigzagLineLength / sc.TickSize;
            LengthStr += sc.FormatGraphValue(Ticks, 0);
            LengthStr += "T";
        }
        else if (ShowLength.GetIndex() == 3) // Percent
        {
            float CurrentBarValue = sc.Close[ZigZagIndex];
            float Percent = static_cast<float>((ZigzagLineLength / CurrentBarValue) * 100);
            LengthStr += sc.FormatGraphValue(Percent, 2);
        }

        BarLabelText += LengthStr;
    }

    if (ShowTime.GetYesNo())
    {
        if (BarLabelText.GetLength() != 0)
        {
            if (UseMultiLineLabels.GetYesNo())
                BarLabelText += "\n";
            else
                BarLabelText += " ";
        }

        SCDateTime DT = sc.BaseDateTimeIn[ZigZagIndex];

```

```

SCString TimeStr;
if (IncludeSecondsInTimeDisplay.GetYesNo())
    TimeStr.Format("%i:%02i:%02i", DT.GetHour(), DT.GetMinute(), DT.GetSecond());
else
    TimeStr.Format("%i:%02i", DT.GetHour(), DT.GetMinute());

BarLabelText += TimeStr;
}

if (ShowZigZagVolume.GetYesNo())
{
    if (BarLabelText.GetLength() != 0)
    {
        if (UseMultiLineLabels.GetYesNo())
            BarLabelText += "\n";
        else
            BarLabelText += " ";
    }

    SCString LengthStr;
    if (ShowLabelPrefixes)
        LengthStr = "V:";

    LengthStr += sc.FormatVolumeValue(static_cast<int64_t>(AccumulatedVolume[ZigZagIndex]), 0,
FormatVolumeWithLargeNumberSuffix.GetYesNo());

    BarLabelText += LengthStr;
}

if (ShowZigZagDuration.GetYesNo())
{
    if (BarLabelText.GetLength() != 0)
    {
        if (UseMultiLineLabels.GetYesNo())
            BarLabelText += "\n";
        else
            BarLabelText += " ";
    }

    double ZigZagTimeSpan = 0;
    if (sc.ChartDataType == INTRADAY_DATA)
    {
        if (ZigZagIndex < sc.ArraySize - 1)
            ZigZagTimeSpan = (sc.BaseDateTimeIn[ZigZagIndex + 1] - sc.BaseDateTimeIn[BackRefIndex +
1]).GetAsDouble();
        else
            ZigZagTimeSpan = (sc.LatestDateTimeForLastBar - sc.BaseDateTimeIn[BackRefIndex +
1]).GetAsDouble();
    }
    else
    {
        ZigZagTimeSpan = (sc.BaseDateTimeIn[ZigZagIndex] - sc.BaseDateTimeIn[BackRefIndex +
1]).GetAsDouble();
    }

    SCString TimeDurationStr;
    if (sc.ChartDataType == INTRADAY_DATA)
    {
        TimeDurationStr = sc.FormatGraphValue(ZigZagTimeSpan, 20);

        if (TimeDurationStr.GetLength() >= 3 && TimeDurationStr[0] == '0' && TimeDurationStr[1] == '0' &&
TimeDurationStr[2] == ':')
            TimeDurationStr = TimeDurationStr.GetSubString(TimeDurationStr.GetLength() - 3, 3);
    }
}

```

```

SCString LengthStr;
if (ShowLabelPrefixes)
    LengthStr = "D:";
if (ZigZagTimeSpan >= 1.0)
{
    SCString DaysStr;
    if (sc.ChartDataType == INTRADAY_DATA)
        DaysStr.Format("%iD ", static_cast<int>(ZigZagTimeSpan));
    else
        DaysStr.Format("%i ", static_cast<int>(ZigZagTimeSpan));

    LengthStr += DaysStr;
}
LengthStr += TimeDurationStr;

BarLabelText += LengthStr;
}

CurrentZigZagNumBars = ZigZagIndex - BackRefIndex;

if (ShowZigZagNumBars.GetYesNo())
{
    if (BarLabelText.GetLength() != 0)
    {
        if (UseMultiLineLabels.GetYesNo())
            BarLabelText += "\n";
        else
            BarLabelText += " ";
    }

    SCString LengthStr;
    if (ShowLabelPrefixes)
        LengthStr = "B:";

    SCString NumStr;
    NumStr.Format("%i ", CurrentZigZagNumBars);
    LengthStr += NumStr;

    BarLabelText += LengthStr;
}

if (DisplayZigZagVolumeDividedByLengthInTicks.GetYesNo())
{
    if (BarLabelText.GetLength() != 0)
    {
        if (UseMultiLineLabels.GetYesNo())
            BarLabelText += "\n";
        else
            BarLabelText += " ";
    }

    if (ShowLabelPrefixes)
        BarLabelText += "V/L:";

    int64_t VolumeDividedByLengthInTicks = static_cast<int64_t>(AccumulatedVolume[ZigZagIndex] /
(ZigzagLineLength / sc.TickSize));

    BarLabelText += sc.FormatVolumeValue(VolumeDividedByLengthInTicks, 0,
FormatVolumeWithLargeNumberSuffix.GetYesNo());

}
}

// now go back one more swing
if (ZigZagReset[BackRefIndex] == 0)

```

```

{
    BackRefIndex = static_cast<int>(ZigZagPeakIndex[BackRefIndex - 1]);

    if (ZigZagPeakType[BackRefIndex] == ZigZagPeakType[ZigZagIndex])
    {
        if (DisplayHHHLLLLLHLabels.GetYesNo())
        {
            if (BarLabelText.GetLength() != 0)
            {
                if (UseMultiLineLabels.GetYesNo())
                    BarLabelText += "\n";
                else
                    BarLabelText += " ";
            }

            if (ZigZagPeakType[BackRefIndex] == 1.0f) //High
            {
                if (ZigZagLine[BackRefIndex] > ZigZagLine[ZigZagIndex]) //LowerHigh
                {
                    BarLabelText += "LH";
                    TextLabels.Data[ZigZagIndex] = 1;
                }
                else //HigherHigh
                {
                    BarLabelText += "HH";
                    TextLabels.Data[ZigZagIndex] = 2;
                }
            }
            else //Low
            {
                if (ZigZagLine[BackRefIndex] < ZigZagLine[ZigZagIndex]) //HigherLow
                {
                    BarLabelText += "HL";
                    TextLabels.Data[ZigZagIndex] = 3;
                }
                else //LowerLow
                {
                    BarLabelText += "LL";
                    TextLabels.Data[ZigZagIndex] = 4;
                }
            }
        }
    }
}

else
{
    if (ZigZagLine[BackRefIndex] > ZigZagLine[ZigZagIndex])
        TextLabels.Data[ZigZagIndex] = 11; // first pivot low
    else
        TextLabels.Data[ZigZagIndex] = 12; // first pivot high
}
}

/*=====*/
SCSFExport scsf_ZigZagStudy(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_ZigZagLine      = sc.Subgraph[0];
    SCSubgraphRef Subgraph_TextLabels      = sc.Subgraph[1];
    SCSubgraphRef Subgraph_ReversalPrice   = sc.Subgraph[2];
    SCSubgraphRef Subgraph_ZigzagLength    = sc.Subgraph[3];
    SCSubgraphRef Subgraph_AccumulatedVolume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_ZigzagNumBars   = sc.Subgraph[5];
    SCSubgraphRef Subgraph_ZigzagMidPoint  = sc.Subgraph[6];
    SCSubgraphRef Subgraph_ExtensionLinesSubgraph = sc.Subgraph[7];
    SCSubgraphRef Subgraph_ZigZagOscillator = sc.Subgraph[8];

```

```

SCSubgraphRef Subgraph_ZigZagLineDuration = sc.Subgraph[9];

SCInputRef Input_VolumeToAccumulate = sc.Input[0];
SCInputRef Input_LabelOffsetType = sc.Input[1];
SCInputRef Input_FormatVolumeWithLargeNumberSuffix = sc.Input[2];
SCInputRef Input_ColorZigZagLine = sc.Input[3];
SCInputRef Input_VersionUpdate = sc.Input[14];
SCInputRef Input_DataHigh = sc.Input[15];
SCInputRef Input_DataLow = sc.Input[16];
SCInputRef Input_CalculationMode = sc.Input[17];
SCInputRef Input_ReversalPercent = sc.Input[18];
SCInputRef Input_ReversalAmount = sc.Input[19];
SCInputRef Input_NumberBarsForReversal = sc.Input[20];
SCInputRef Input_ShowRevPrice = sc.Input[21];
SCInputRef Input_DisplayHHHLLLLLHLabels = sc.Input[22];
SCInputRef Input_LabelsOffset = sc.Input[23];
SCInputRef Input_AdditionalOutputForSpreadsheets = sc.Input[24];
SCInputRef Input_ShowTime = sc.Input[25];
SCInputRef Input_ShowLength = sc.Input[26];
SCInputRef Input_CalcOnBarClose = sc.Input[27];
SCInputRef Input_CalculateZigZagVolume_UpgradeOnly = sc.Input[28];
SCInputRef Input_ShowZigZagVolume = sc.Input[29];
SCInputRef Input_ShowZigZagDuration = sc.Input[30];
SCInputRef Input_UseMultiLineLabels = sc.Input[31];
SCInputRef Input_OmitLabelPrefixes = sc.Input[32];
SCInputRef Input_ShowZigZagNumBars = sc.Input[33];
SCInputRef Input_ResetAtStartOfSession = sc.Input[34];
SCInputRef Input_DisplayZigZagVolumeDividedByLengthInTicks = sc.Input[35];
SCInputRef Input_ExtendEndpointsUntilFutureIntersection = sc.Input[36];
SCInputRef Input_IncludeSecondsInTimeDisplay = sc.Input[37];
SCInputRef Input_DisplayValueLabel = sc.Input[38];
SCInputRef Input_SingleLineTextHorizontalAlignment = sc.Input[39];
SCInputRef Input_MultiLineTextHorizontalAlignment = sc.Input[40];

```

```

if(sc.SetDefaults)
{
    sc.GraphName="Zig Zag";

    //Manual looping
    sc.AutoLoop = 0;

    sc.GraphRegion = 0;

    sc.ValueFormat = VALUEFORMAT_INHERITED;

    Subgraph_ZigZagLine.Name = "Zig Zag";
    Subgraph_ZigZagLine.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_ZigZagLine.LineStyle = LINESTYLE_DOT;
    Subgraph_ZigZagLine.PrimaryColor = RGB(255,0,255);
    Subgraph_ZigZagLine.SecondaryColorUsed = 1;
    Subgraph_ZigZagLine.SecondaryColor = RGB(255,0,0);
    Subgraph_ZigZagLine.LineWidth = 3;
    Subgraph_ZigZagLine.DrawZeros = false;

    Subgraph_TextLabels.Name = "Text Labels";
    Subgraph_TextLabels.SecondaryColorUsed = 1;
    Subgraph_TextLabels.PrimaryColor = RGB(0,255,0);
    Subgraph_TextLabels.SecondaryColor = RGB(255,0,0);
    Subgraph_TextLabels.LineWidth = 10;
    Subgraph_TextLabels.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;

    Subgraph_ReversalPrice.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_ZigzagLength.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_ZigzagNumBars.DrawStyle = DRAWSTYLE_IGNORE;

```

```

Subgraph_AccumulatedVolume.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AccumulatedVolume.SecondaryColorUsed = 1;

Subgraph_ZigzagMidPoint.Name = "Zig Zag Mid-Point";
Subgraph_ZigzagMidPoint.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_ZigzagMidPoint.PrimaryColor = RGB(128,128,255);
Subgraph_ZigzagMidPoint.LineWidth = 1;
Subgraph_ZigzagMidPoint.DrawZeros = false;

Subgraph_ExtensionLinesSubgraph.Name = "Extension Lines";
Subgraph_ExtensionLinesSubgraph.DrawStyle = DRAWSTYLE_LINE;
Subgraph_ExtensionLinesSubgraph.PrimaryColor = RGB(128, 128, 255);
Subgraph_ExtensionLinesSubgraph.SecondaryColorUsed = true;
Subgraph_ExtensionLinesSubgraph.SecondaryColor = RGB(128, 0, 255);
Subgraph_ExtensionLinesSubgraph.LineWidth = 1;
Subgraph_ExtensionLinesSubgraph.DrawZeros = false;

Subgraph_ZigZagOscillator.Name = "Zig Zag Oscillator";
Subgraph_ZigZagOscillator.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_ZigZagOscillator.PrimaryColor = RGB(0, 200, 0);
Subgraph_ZigZagOscillator.SecondaryColor = RGB(200, 0, 0);
Subgraph_ZigZagOscillator.AutoColoring = AUTOCOLOR_POSNEG;
Subgraph_ZigZagOscillator.LineWidth = 3;
Subgraph_ZigZagOscillator.DrawZeros = false;

Input_VersionUpdate.SetInt(4);

unsigned short DisplayOrder = 1;

Input_DataHigh.Name = "Input Data for High";
Input_DataHigh.SetInputDataIndex(SC_HIGH);
Input_DataHigh.DisplayOrder = DisplayOrder++;

Input_DataLow.Name = "Input Data for Low";
Input_DataLow.SetInputDataIndex(SC_LOW);
Input_DataLow.DisplayOrder = DisplayOrder++;

Input_CalculationMode.Name = "Calculation Mode (1,2,3)";
Input_CalculationMode.SetInt(1);
Input_CalculationMode.SetIntLimits(1, 3);
Input_CalculationMode.DisplayOrder = DisplayOrder++;

Input_ReversalPercent.Name = "Reversal % for Calculation Mode 1";
Input_ReversalPercent.SetFloat(.5f);
Input_ReversalPercent.DisplayOrder = DisplayOrder++;

Input_ReversalAmount.Name = "Reversal Amount for Calculation Mode 2,3";
Input_ReversalAmount.SetFloat(0.01f);
Input_ReversalAmount.DisplayOrder = DisplayOrder++;

Input_NumberBarsForReversal.Name = "Number of Bars Required for Reversal (Calculation Mode 2)";
Input_NumberBarsForReversal.SetInt(5);
Input_NumberBarsForReversal.DisplayOrder = DisplayOrder++;

Input_VolumeToAccumulate.Name = "Volume To Accumulate";
Input_VolumeToAccumulate.SetCustomInputStrings("None;Total Volume;Bid Volume;Ask Volume;Ask Bid Volume
Difference");
Input_VolumeToAccumulate.SetCustomInputIndex(0);
Input_VolumeToAccumulate.DisplayOrder = DisplayOrder++;

Input_ResetAtStartOfSession.Name = "Reset ZigZag At Start Of Trading Day";
Input_ResetAtStartOfSession.SetYesNo(0);
Input_ResetAtStartOfSession.DisplayOrder = DisplayOrder++;

Input_CalcOnBarClose.Name = "Calculate New Values On Bar Close";

```

```

Input_CalcOnBarClose.SetYesNo(0);
Input_CalcOnBarClose.DisplayOrder = DisplayOrder++;

Input_AdditionalOutputForSpreadsheets.Name = "Additional Output for Spreadsheets";
Input_AdditionalOutputForSpreadsheets.SetYesNo(false);
Input_AdditionalOutputForSpreadsheets.DisplayOrder = DisplayOrder++;

Input_DisplayHHHLLLLLHLabels.Name = "Display HH,HL,LL,LH Labels";
Input_DisplayHHHLLLLLHLabels.SetYesNo(0);
Input_DisplayHHHLLLLLHLabels.DisplayOrder = DisplayOrder++;

Input_ShowRevPrice.Name = "Display Reversal Price";
Input_ShowRevPrice.SetYesNo(0);
Input_ShowRevPrice.DisplayOrder = DisplayOrder++;

Input_ShowLength.Name = "Display Length of Zig Zag Line";
Input_ShowLength.SetCustomInputStrings("None;Points;Ticks;Percent");
Input_ShowLength.SetCustomInputIndex(0);
Input_ShowLength.DisplayOrder = DisplayOrder++;

Input_ShowZigZagVolume.Name = "Display ZigZag Volume";
Input_ShowZigZagVolume.SetYesNo(0);
Input_ShowZigZagVolume.DisplayOrder = DisplayOrder++;

Input_FormatVolumeWithLargeNumberSuffix.Name = "Format Volume Using Large Number Suffix (K/M)";
Input_FormatVolumeWithLargeNumberSuffix.SetYesNo(0);
Input_FormatVolumeWithLargeNumberSuffix.DisplayOrder = DisplayOrder++;

Input_ShowTime.Name = "Display Time Labels";
Input_ShowTime.SetYesNo(0);
Input_ShowTime.DisplayOrder = DisplayOrder++;

Input_IncludeSecondsInTimeDisplay.Name = "Include Seconds in Time Display";
Input_IncludeSecondsInTimeDisplay.SetYesNo(1);
Input_IncludeSecondsInTimeDisplay.DisplayOrder = DisplayOrder++;

Input_ShowZigZagDuration.Name = "Display ZigZag Time Duration";
Input_ShowZigZagDuration.SetYesNo(0);
Input_ShowZigZagDuration.DisplayOrder = DisplayOrder++;

Input_ShowZigZagNumBars.Name = "Display ZigZag Number Of Bars";
Input_ShowZigZagNumBars.SetYesNo(0);
Input_ShowZigZagNumBars.DisplayOrder = DisplayOrder++;

Input_DisplayZigZagVolumeDividedByLengthInTicks.Name = "Display ZigZag Volume / Length in Ticks";
Input_DisplayZigZagVolumeDividedByLengthInTicks.SetYesNo(0);
Input_DisplayZigZagVolumeDividedByLengthInTicks.DisplayOrder = DisplayOrder++;

Input_UseMultiLineLabels.Name = "Use Multi-Line Labels";
Input_UseMultiLineLabels.SetYesNo(0);
Input_UseMultiLineLabels.DisplayOrder = DisplayOrder++;

Input_OmitLabelPrefixes.Name = "Omit Label Prefixes";
Input_OmitLabelPrefixes.SetYesNo(0);
Input_OmitLabelPrefixes.DisplayOrder = DisplayOrder++;

Input_LabelOffsetType.Name = "Text Label Offset Specified As";
Input_LabelOffsetType.SetCustomInputStrings("Percentage Of Bar;Tick Offset");
Input_LabelOffsetType.SetCustomInputIndex(1);
Input_LabelOffsetType.DisplayOrder = DisplayOrder++;

Input_LabelsOffset.Name = "Text Label Offset";
Input_LabelsOffset.SetFloat(1.0f);
Input_LabelsOffset.DisplayOrder = DisplayOrder++;

```

```

Input_ColorZigZagLine.Name = "Color ZigZag Line Using";
Input_ColorZigZagLine.SetCustomInputStrings("None;Slope;Trend Confirmed Volume;Confirmed Volume");
Input_ColorZigZagLine.SetCustomInputIndex(0);
Input_ColorZigZagLine.DisplayOrder = DisplayOrder++;

Input_ExtendEndPointsUntilFutureIntersection.Name = "Extend End Points Until Future Intersection";
Input_ExtendEndPointsUntilFutureIntersection.SetYesNo(false);
Input_ExtendEndPointsUntilFutureIntersection.DisplayOrder = DisplayOrder++;

Input_DisplayValueLabel.Name = "Display Value Label for Extension Lines";
Input_DisplayValueLabel.SetYesNo(false);
Input_DisplayValueLabel.DisplayOrder = DisplayOrder++;

Input_SingleLineTextHorizontalAlignment.Name = "Single Line Text Horizontal Alignment";
Input_SingleLineTextHorizontalAlignment.SetCustomInputStrings("Left;Center;Right");
Input_SingleLineTextHorizontalAlignment.SetCustomInputIndex(1);
Input_SingleLineTextHorizontalAlignment.DisplayOrder = DisplayOrder++;

Input_MultiLineTextHorizontalAlignment.Name = "Multi-Line Text Horizontal Alignment";
Input_MultiLineTextHorizontalAlignment.SetCustomInputStrings("Left;Center;Right");
Input_MultiLineTextHorizontalAlignment.SetCustomInputIndex(0);
Input_MultiLineTextHorizontalAlignment.DisplayOrder = DisplayOrder++;

return;
}

SCFloatArrayRef ZigZagPeakType = Subgraph_ZigZagLine.Arrays[0]; // ZigZagPeakType : +1=high peak, -1=low
peak, 0=not peak
SCFloatArrayRef ZigZagPeakIndex = Subgraph_ZigZagLine.Arrays[1]; // ZigZagPeakIndex: index of current peak, -
1=no peak yet
SCFloatArrayRef ZigZagReset = Subgraph_ZigZagLine.Arrays[2]; // ZigZagReset: 0=no reset, 1=reset
SCFloatArrayRef ZigZagTrend = Subgraph_ZigZagLine.Arrays[3]; // ZigZagTrend: 0=none, 1=confirmed up, -
1=confirmed down

int& LastCalcIndex = sc.GetPersistentInt(1);
int& PriorLastZZEndpointIndex = sc.GetPersistentInt(2);
int& ZigZagStartIndex = sc.GetPersistentInt(3);
int& LastUsedLabelDrawingNumber = sc.GetPersistentInt(4);
int& LastUsedBarIndexForExtensionLine = sc.GetPersistentInt(5);

if(sc.UpdateStartIndex == 0)
{
    LastCalcIndex = -1;
    LastUsedLabelDrawingNumber = -1;
    LastUsedBarIndexForExtensionLine = -1;
    PriorLastZZEndpointIndex = -1;
    ZigZagStartIndex = 0;
    ZigZagReset[0] = 1.0f;
    Subgraph_TextLabels[0] = 10; // first point

    if (Input_VersionUpdate.GetInt() < 2)
    {
        Input_ResetAtStartOfSession.SetYesNo(0);
        Input_VersionUpdate.SetInt(2);
    }

    if (Input_VersionUpdate.GetInt() < 3)
    {
        Input_LabelOffsetType.SetCustomInputIndex(0);
        Input_FormatVolumeWithLargeNumberSuffix.SetYesNo(0);
        Subgraph_ZigZagLine.SecondaryColorUsed = 1;
        Subgraph_ZigZagLine.SecondaryColor = RGB(255,0,0);
        Input_ColorZigZagLine.SetCustomInputIndex(0);
        Input_VolumeToAccumulate.SetCustomInputIndex(Input_VolumeToAccumulate.GetIndex()+1);
        if (Input_CalculateZigZagVolume_UpgradeOnly.GetYesNo() == 0)

```



```

    Input_VolumeToAccumulate.SetCustomInputIndex(0);

    Input_VersionUpdate.SetInt(3);
}

if (Input_VersionUpdate.GetInt() < 4)
{
    Input_SingleLineTextHorizontalAlignment.SetCustomInputIndex(1);
    Input_MultiLineTextHorizontalAlignment.SetCustomInputIndex(0);
    Input_VersionUpdate.SetInt(4);
}

if (Input_AdditionalOutputForSpreadsheets.GetYesNo())
{
    Subgraph_ReversalPrice.Name = "Reversal Price";
    Subgraph_ZigzagLength.Name = "Zig Zag Line Length";
    Subgraph_ZigzagNumBars.Name = "Zig Zag Num Bars";
}
else
{
    Subgraph_ReversalPrice.Name = "";
    Subgraph_ZigzagLength.Name = "";
    Subgraph_ZigzagNumBars.Name = "";
}

if (Input_ShowZigZagVolume.GetYesNo() && Input_VolumeToAccumulate.GetIndex() == 0)
    Input_VolumeToAccumulate.SetCustomInputIndex(1);

if (Input_VolumeToAccumulate.GetIndex() != 0)
    Subgraph_AccumulatedVolume.Name = "AccumulatedVolume";
else
    Subgraph_AccumulatedVolume.Name = "";
}
else
{
    if (LastUsedLabelDrawingNumber != -1)
    {
        sc.DeleteACSCChartDrawing(sc.ChartNumber, TOOL_DELETE_CHARTDRAWING,
LastUsedLabelDrawingNumber);

        LastUsedLabelDrawingNumber = -1;
    }

    if (LastUsedBarIndexForExtensionLine != -1)
    {
        sc.DeleteLineUntilFutureIntersection(LastUsedBarIndexForExtensionLine, 0);

        LastUsedBarIndexForExtensionLine = -1;
    }
}

}

bool ShowLabelPrefixes = Input_OmitLabelPrefixes.GetYesNo() == 0;

SCDateTime NextSessionStart = 0;
if (sc.UpdateStartIndex > 0)
    NextSessionStart = SCDateTime(sc.GetTradingDayStartDateTimeOfBar(sc.BaseDateTimeIn[sc.UpdateStartIndex -
1])) + SCDateTime::DAYS(1);

sc.EarliestUpdateSubgraphDataArrayIndex = sc.UpdateStartIndex;

// Standard Manual Looping loop
for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; ++Index)
{

```

```

if (Input_ResetAtStartOfSession.GetYesNo() != 0)
{
    SCDateTime IndexDateTime = sc.BaseDateTimeIn[Index];

    if (IndexDateTime >= NextSessionStart)
    {
        ZigZagStartIndex = Index;
        ZigZagReset[Index] = 1.0f;
        Subgraph_TextLabels[Index] = 10; // first point

        NextSessionStart = SCDateTime(sc.GetTradingDayStartDateTimeOfBar(IndexDateTime)) +
SCDateTime::DAYS(1);
    }
}

int CalcIndex = Index;

if (Input_CalcOnBarClose.GetYesNo())
{
    CalcIndex--;
    if (CalcIndex <= LastCalcIndex)
        continue;
    LastCalcIndex = CalcIndex;
}

//Calculate and set the zigzag lines
switch (Input_CalculationMode.GetInt())
{
case 1:
    sc.ZigZag(
        sc.BaseData[Input_DataHigh.GetInputDataIndex()],
        sc.BaseData[Input_DataLow.GetInputDataIndex()],
        Subgraph_ZigZagLine,
        CalcIndex,
        Input_ReversalPercent.GetFloat() * 0.01f,
        ZigZagStartIndex);
    break;
case 2:
    sc.ZigZag2(
        sc.BaseData[Input_DataHigh.GetInputDataIndex()],
        sc.BaseData[Input_DataLow.GetInputDataIndex()],
        Subgraph_ZigZagLine,
        CalcIndex,
        Input_NumberBarsForReversal.GetInt(),
        Input_ReversalAmount.GetFloat(),
        ZigZagStartIndex);
    break;
case 3:
    sc.ZigZag(
        sc.BaseData[Input_DataHigh.GetInputDataIndex()],
        sc.BaseData[Input_DataLow.GetInputDataIndex()],
        Subgraph_ZigZagLine,
        CalcIndex,
        0.0f,
        Input_ReversalAmount.GetFloat(),
        ZigZagStartIndex);
    break;
}
}

AccumulateZigZagVolume(sc, Subgraph_AccumulatedVolume, ZigZagPeakType,
Input_VolumeToAccumulate.GetIndex(), ZigZagReset);

if (PriorLastZZEndpointIndex != -1)
{

```

```

for (int ZeroIndex = PriorLastZZEndpointIndex; ZeroIndex < sc.ArraySize; ZeroIndex++)
{
    Subgraph_TextLabels[ZeroIndex] = 0;
    Subgraph_ReversalPrice[ZeroIndex] = 0;
    Subgraph_ZigzagLength[ZeroIndex] = 0;
    Subgraph_ZigzagNumBars[ZeroIndex] = 0;
}
}

if (PriorLastZZEndpointIndex == -1)
    PriorLastZZEndpointIndex = 0;

// Get the index of the last endpoint
int IndexOfLastEndPoint = 0;
if (Input_CalcOnBarClose.GetYesNo() && sc.ArraySize > 1)
    IndexOfLastEndPoint = static_cast<int>(ZigZagPeakIndex[sc.ArraySize-2]);
else
    IndexOfLastEndPoint = static_cast<int>(ZigZagPeakIndex[sc.ArraySize-1]);

for (int ZigZagIndex = PriorLastZZEndpointIndex; ZigZagIndex < sc.ArraySize; ZigZagIndex++)
{
    if (Subgraph_ZigZagLine[ZigZagIndex] != 0)
    {
        if (Input_ColorZigZagLine.GetIndex() == 0)
        {
            Subgraph_ZigZagLine.DataColor[ZigZagIndex] = Subgraph_ZigZagLine.PrimaryColor;
        }
        else if (Input_ColorZigZagLine.GetIndex() == 1 && ZigZagIndex > 0)
        {
            if (Subgraph_ZigZagLine[ZigZagIndex] > Subgraph_ZigZagLine[ZigZagIndex-1])
                Subgraph_ZigZagLine.DataColor[ZigZagIndex] = Subgraph_ZigZagLine.PrimaryColor;
            else
                Subgraph_ZigZagLine.DataColor[ZigZagIndex] = Subgraph_ZigZagLine.SecondaryColor;
        }
        else if (Input_ColorZigZagLine.GetIndex() == 2 || Input_ColorZigZagLine.GetIndex() == 3)
        {
            if (ZigZagTrend[ZigZagIndex] > 0)
                Subgraph_ZigZagLine.DataColor[ZigZagIndex] = Subgraph_ZigZagLine.PrimaryColor;
            else if (ZigZagTrend[ZigZagIndex] < 0)
                Subgraph_ZigZagLine.DataColor[ZigZagIndex] = Subgraph_ZigZagLine.SecondaryColor;
        }
    }
}

int CurrentPeakIndex = static_cast<int>(ZigZagPeakIndex[ZigZagIndex]);

Subgraph_ZigZagOscillator[ZigZagIndex] = sc.Close[ZigZagIndex] - Subgraph_ZigZagLine[CurrentPeakIndex];

if (ZigZagPeakType[ZigZagIndex] == 0.0f) //not a peak, nothing to be drawn
    continue;

if (ZigZagReset[ZigZagIndex] != 0) //Continue processing if this is a reset index
{
    continue;
}

int PrevEndPointIndex = -1;
if (ZigZagIndex > 0)
    PrevEndPointIndex = static_cast<int>(ZigZagPeakIndex[ZigZagIndex-1]);

if (PrevEndPointIndex >= 0)
{
    float MidPoint = (Subgraph_ZigZagLine[ZigZagIndex] + Subgraph_ZigZagLine[PrevEndPointIndex]) / 2.0f;
    if (Subgraph_ZigzagMidPoint[ZigZagIndex] != MidPoint)
    {

```

```

        for (int UpdateIndex=PrevEndPointIndex+1; UpdateIndex<=ZigZagIndex; UpdateIndex++)
            Subgraph_ZigzagMidPoint[UpdateIndex] = MidPoint;
    }
}

if (Input_ColorZigZagLine.GetIndex() == 2 || Input_ColorZigZagLine.GetIndex() == 3) // determine zig zag trend color
with confirmed volume
{
    if (PrevEndPointIndex >= 0)
    {
        float Trend = ZigZagPeakType[ZigZagIndex]; // default trend is current direction

        if (PrevEndPointIndex > 0 && ZigZagReset[PrevEndPointIndex] == 0) // have more than one swing
        {
            if (((Input_ColorZigZagLine.GetIndex() == 2 && Trend != ZigZagTrend[PrevEndPointIndex]) || // use current
trend and volume
                Input_ColorZigZagLine.GetIndex() == 3) // only use volume
                && Subgraph_AccumulatedVolume[ZigZagIndex] < Subgraph_AccumulatedVolume[PrevEndPointIndex])
            {
                Trend = -Trend;
            }
        }

        if (ZigZagTrend[ZigZagIndex] != Trend)
        {
            for (int UpdateIndex=PrevEndPointIndex+1; UpdateIndex<=ZigZagIndex; UpdateIndex++)
            {
                ZigZagTrend[UpdateIndex] = Trend;

                if (Trend > 0)
                    Subgraph_ZigZagLine.DataColor[UpdateIndex] = Subgraph_ZigZagLine.PrimaryColor;
                else
                    Subgraph_ZigZagLine.DataColor[UpdateIndex] = Subgraph_ZigZagLine.SecondaryColor;
            }
        }
    }
}
}
}

```

//At this point we have a label to draw

PriorLastZZEndpointIndex = ZigZagIndex;

SCString BarLabelText;

int VerticalTextAlignment = ZigZagPeakType[ZigZagIndex] == 1.0 ? DT_BOTTOM : DT_TOP;

double ZigzagLineLength = 0;

int CurrentZigZagNumBars = 0;

CreateZigZagLineLabel(BarLabelText, ZigZagPeakType, ZigZagIndex, ZigZagPeakIndex, Input_ShowRevPrice,
Input_UseMultiLineLabels, sc, Subgraph_ZigZagLine, Input_ShowLength, ShowLabelPrefixes, Input_ShowTime,
Input_IncludeSecondsInTimeDisplay, Input_ShowZigZagVolume, Subgraph_AccumulatedVolume,
Input_FormatVolumeWithLargeNumberSuffix, Input_ShowZigZagDuration, Input_ShowZigZagNumBars,
Input_DisplayZigZagVolumeDividedByLengthInTicks, ZigZagReset, Input_DisplayHHHLLLLLHLabels,
Subgraph_TextLabels, ZigzagLineLength, CurrentZigZagNumBars);

```

if (BarLabelText.GetLength() > 0 && !sc.HideStudy)
{
    s_UseTool Tool;
    Tool.Clear();
    Tool.ChartNumber = sc.ChartNumber;
    Tool.Region = sc.GraphRegion;
    Tool.DrawingType = DRAWING_TEXT;
    Tool.ReverseTextColor = 0;
    Tool.BeginIndex = ZigZagIndex;
}

```

```

float Offset;
if (Input_LabelOffsetType.GetIndex() == 0)
    Offset = Input_LabelsOffset.GetFloat()*0.01f*(sc.High[ZigZagIndex]-sc.Low[ZigZagIndex]);
else
    Offset = Input_LabelsOffset.GetFloat()*sc.TickSize;

if(ZigZagPeakType[ZigZagIndex] == 1.0f)
    Tool.BeginValue = Subgraph_ZigZagLine[ZigZagIndex] + Offset;
else
    Tool.BeginValue = Subgraph_ZigZagLine[ZigZagIndex] - Offset;

if (ZigZagPeakType[ZigZagIndex] == 1.0f)
    Tool.Color = Subgraph_TextLabels.PrimaryColor;
else
    Tool.Color = Subgraph_TextLabels.SecondaryColor;

//Tool.FontFace = sc.GetChartTextFontFaceName();
Tool.FontSize = Subgraph_TextLabels.LineWidth;
Tool.FontBold = true;
Tool.Text = BarLabelText;
Tool.AddMethod = UTAM_ADD_ALWAYS;
Tool.LineNumber = -1;

if (Input_UseMultiLineLabels.GetYesNo())
{
    if (Input_MultiLineTextHorizontalAlignment.GetIndex() == 0)
    {
        Tool.TextAlignment = DT_LEFT | VerticalTextAlignment;
    }
    else if (Input_MultiLineTextHorizontalAlignment.GetIndex() == 1)
    {
        Tool.TextAlignment = DT_CENTER | VerticalTextAlignment;
    }
    else
    {
        Tool.TextAlignment = DT_RIGHT | VerticalTextAlignment;
    }
}
else
{
    if (Input_SingleLineTextHorizontalAlignment.GetIndex() == 0)
    {
        Tool.TextAlignment = DT_LEFT | VerticalTextAlignment;
    }
    else if (Input_SingleLineTextHorizontalAlignment.GetIndex() == 1)
    {
        Tool.TextAlignment = DT_CENTER | VerticalTextAlignment;
    }
    else
    {
        Tool.TextAlignment = DT_RIGHT | VerticalTextAlignment;
    }
}

sc.UseTool(Tool);
LastUsedLabelDrawingNumber = Tool.LineNumber;
}

if (Input_ExtendEndPointsUntilFutureIntersection.GetYesNo())
{
    sc.AddLineUntilFutureIntersection
    (ZigZagIndex
    , 0 // LineIDForBar
    , Subgraph_ZigZagLine[ZigZagIndex]

```

```

        , ZigZagPeakType[ZigZagIndex] == 1.0f ? Subgraph_ExtensionLinesSubgraph.PrimaryColor :
Subgraph_ExtensionLinesSubgraph.SecondaryColor
        , Subgraph_ExtensionLinesSubgraph.LineWidth
        , Subgraph_ExtensionLinesSubgraph.LineStyle
        , Input_DisplayValueLabel.GetBoolean()
        , false
        , ""
    );

    LastUsedBarIndexForExtensionLine = ZigZagIndex;
}

Subgraph_ReversalPrice[ZigZagIndex] = Subgraph_ZigZagLine[ZigZagIndex];
Subgraph_ZigzagLength[ZigZagIndex] = static_cast<float>(ZigzagLineLength);
Subgraph_ZigzagNumBars[ZigZagIndex] = static_cast<float>(CurrentZigZagNumBars);
}

//Determine the beginning of the last line
int Count = 0; //To count the number of zigzag pivots
int StartingIndex = sc.ArraySize;
while (StartingIndex > 0 && Count < 2)
{
    StartingIndex--;
    if (ZigZagPeakType[StartingIndex] != 0.0f)
        Count ++;
}

sc.EarliestUpdateSubgraphDataArrayIndex = StartingIndex;
}

/*=====*/
SCSFExport scsf_ZigZagCumulativeVolumeStudy(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_ZigZagLine = sc.Subgraph[0];
    SCSubgraphRef Subgraph_AccumulatedVolume = sc.Subgraph[4];

    SCFloatArrayRef Array_ZigZagPeakType = Subgraph_ZigZagLine.Arrays[0];
    SCFloatArrayRef Array_ZigZagReset = Subgraph_ZigZagLine.Arrays[2]; // ZigZagReset: 0=no reset, 1=reset

    SCInputRef Input_VolumeToAccumulate = sc.Input[0];
    SCInputRef Input_VersionUpdate = sc.Input[14];
    SCInputRef Input_DataHigh = sc.Input[15];
    SCInputRef Input_DataLow = sc.Input[16];
    SCInputRef Input_CalculationMode = sc.Input[17];
    SCInputRef Input_ReversalPercent = sc.Input[18];
    SCInputRef Input_ReversalAmount = sc.Input[19];
    SCInputRef Input_NumberBarsForReversal = sc.Input[20];
    SCInputRef Input_ResetAtStartOfSession = sc.Input[26];
    SCInputRef Input_CalcOnBarClose = sc.Input[27];

    if(sc.SetDefaults)
    {
        sc.GraphName="Zig Zag Cumulative Volume";

        //Manual looping
        sc.AutoLoop = 0;
        sc.ValueFormat= 0;
        sc.GraphRegion = 1;

        Subgraph_ZigZagLine.Name = "Zig Zag";
        Subgraph_ZigZagLine.DrawStyle = DRAWSTYLE_IGNORE;

        Subgraph_AccumulatedVolume.Name= "AccumulatedVolume";
        Subgraph_AccumulatedVolume.DrawStyle = DRAWSTYLE_BAR;
    }
}

```

```

Subgraph_AccumulatedVolume.SecondaryColorUsed = 1;

Input_VolumeToAccumulate.Name= "Volume To Accumulate";
Input_VolumeToAccumulate.SetCustomInputStrings("Total Volume;Bid Volume;Ask Volume;Ask Bid Volume
Difference");
Input_VolumeToAccumulate.SetCustomInputIndex(0);

Input_VersionUpdate.SetInt(2);

Input_DataHigh.Name = "Input Data for High";
Input_DataHigh.SetInputDataIndex(SC_HIGH);

Input_DataLow.Name = "Input Data for Low";
Input_DataLow.SetInputDataIndex(SC_LOW);

Input_CalculationMode.Name = "Calculation Mode (1,2,3)";
Input_CalculationMode.SetInt(1);
Input_CalculationMode.SetIntLimits(1, 3);

Input_ReversalPercent.Name = "Reversal % for Calculation Mode 1";
Input_ReversalPercent.SetFloat(.5f);

Input_ReversalAmount.Name = "Reversal Amount for Calculation Mode 2,3";
Input_ReversalAmount.SetFloat(0.01f);

Input_NumberBarsForReversal.Name = "Number of Bars Required for Reversal (Calculation Mode 2)";
Input_NumberBarsForReversal.SetInt(5);

Input_ResetAtStartOfSession.Name = "Reset ZigZag At Start Of Trading Day";
Input_ResetAtStartOfSession.SetYesNo(0);

Input_CalcOnBarClose.Name = "Calculate New Values On Bar Close";
Input_CalcOnBarClose.SetYesNo(0);

return;
}

int &LastCalcIndex = sc.GetPersistentInt(1);
int &ZigZagStartIndex = sc.GetPersistentInt(2);

if (sc.UpdateStartIndex == 0)
{
    LastCalcIndex = -1;
    ZigZagStartIndex = 0;
    Array_ZigZagReset[0] = 1.0f;

    if (Input_VersionUpdate.GetInt() < 2)
    {
        Input_ResetAtStartOfSession.SetYesNo(0);
        Input_VersionUpdate.SetInt(2);
    }
}

int VolumeTypeToAccumulate = Input_VolumeToAccumulate.GetIndex() + 1; // adjust for no "none" option

SCDateTime NextSessionStart = 0;
if (sc.UpdateStartIndex > 0)
    NextSessionStart = SCDateTime(sc.GetTradingDayStartDateTimeOfBar(sc.BaseDateTimeIn[sc.UpdateStartIndex -
1])) + SCDateTime::DAYS(1);

sc.EarliestUpdateSubgraphDataArrayIndex = sc.UpdateStartIndex;

// Standard Manual Looping loop
for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
{

```

```

if (Input_ResetAtStartOfSession.GetYesNo() != 0)
{
    SCDateTime IndexDateTime = sc.BaseDateTimeIn[Index];

    if (IndexDateTime >= NextSessionStart)
    {
        ZigZagStartIndex = Index;
        Array_ZigZagReset[Index] = 1.0f;

        NextSessionStart = SCDateTime(sc.GetTradingDayStartDateTimeOfBar(IndexDateTime)) +
SCDateTime::DAYS(1);
    }
}

int CalcIndex = Index;

if (Input_CalcOnBarClose.GetYesNo())
{
    CalcIndex--;
    if (CalcIndex <= LastCalcIndex)
        continue;
    LastCalcIndex = CalcIndex;
}

switch (Input_CalculationMode.GetInt())
{
case 1:
    sc.ZigZag(
        sc.BaseData[Input_DataHigh.GetInputDataIndex()],
        sc.BaseData[Input_DataLow.GetInputDataIndex()],
        Subgraph_ZigZagLine,
        CalcIndex,
        Input_ReversalPercent.GetFloat() * 0.01f,
        ZigZagStartIndex);
    break;
case 2:
    sc.ZigZag2(
        sc.BaseData[Input_DataHigh.GetInputDataIndex()],
        sc.BaseData[Input_DataLow.GetInputDataIndex()],
        Subgraph_ZigZagLine,
        CalcIndex,
        Input_NumberBarsForReversal.GetInt(),
        Input_ReversalAmount.GetFloat(),
        ZigZagStartIndex);
    break;
case 3:
    sc.ZigZag(
        sc.BaseData[Input_DataHigh.GetInputDataIndex()],
        sc.BaseData[Input_DataLow.GetInputDataIndex()],
        Subgraph_ZigZagLine,
        CalcIndex,
        0.0f,
        Input_ReversalAmount.GetFloat(),
        ZigZagStartIndex);
    break;
}
}

AccumulateZigZagVolume(sc, Subgraph_AccumulatedVolume, Array_ZigZagPeakType, VolumeTypeToAccumulate,
Array_ZigZagReset);

//Determine the beginning of the last line
int Count = 0; //To count the number of zigzag pivots
int StartingIndex = sc.ArraySize;

```



```

while (StartingIndex > 0 && Count < 2)
{
    StartingIndex--;
    if (Array_ZigZagPeakType[StartingIndex] != 0.0f)
        Count ++;
}

sc.EarliestUpdateSubgraphDataArrayIndex = StartingIndex;
}

/*=====*/
SCSFExport scsf_VolumeColoredBasedOnVolume2(SCStudyInterfaceRef sc)
{

    SCSubgraphRef Subgraph_Volume = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Volume3rdColor = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Volume4thColor = sc.Subgraph[2];

    if (sc.SetDefaults)
    {
        // Set the configuration and defaults

        sc.GraphName = "Volume - Colored Based on Volume 2";

        sc.StudyDescription = "This study displays the Volume per bar. Each volume bar is colored based upon whether the volume is up or down compared to the previous bar volume. It uses a 3rd color when the Volume is lower than the 2 previous bars.";

        sc.AutoLoop = 1; // true

        Subgraph_Volume.Name = "Volume";
        Subgraph_Volume.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_Volume.SecondaryColorUsed = 1;
        Subgraph_Volume.PrimaryColor = RGB(0,255,0);
        Subgraph_Volume.SecondaryColor = RGB(255,0,0);

        Subgraph_Volume3rdColor.Name = "Volume 3rd Color";
        Subgraph_Volume3rdColor.PrimaryColor = RGB(255,128,64);

        Subgraph_Volume4thColor.Name = "Default Color";
        Subgraph_Volume4thColor.PrimaryColor = RGB(0,128,255);

        return;
    }

    // Section 2 - Do data processing here

    Subgraph_Volume[sc.Index] = sc.Volume[sc.Index];

    if(sc.Volume[sc.Index] < sc.Volume[sc.Index-1]
    && sc.Volume[sc.Index] < sc.Volume[sc.Index-2])
        Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume3rdColor.PrimaryColor;
    else if(sc.Volume[sc.Index-1] > sc.Volume[sc.Index])
        Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume.SecondaryColor;
    else if(sc.Volume[sc.Index-1] < sc.Volume[sc.Index])
        Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume.PrimaryColor;
    else //If volume is equal
        Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume4thColor.PrimaryColor;

}

/*=====*/
SCSFExport scsf_BackgroundColoring(SCStudyInterfaceRef sc)

```

```

{
    SCSubgraphRef Subgraph_Top = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Bottom = sc.Subgraph[1];

    if (sc.SetDefaults)
    {
        // Set the configuration and defaults

        sc.GraphName = "Background Coloring";

        sc.AutoLoop = 1;
        sc.GraphRegion = 0;

        Subgraph_Top.Name = "Top";
        Subgraph_Top.DrawStyle = DRAWSTYLE_FILL_RECTANGLE_TOP;
        Subgraph_Top.PrimaryColor = RGB(0,255,0);
        Subgraph_Top.DrawZeros = false;

        Subgraph_Bottom.Name = "Bottom";
        Subgraph_Bottom.DrawStyle = DRAWSTYLE_FILL_RECTANGLE_BOTTOM;
        Subgraph_Bottom.DrawZeros = RGB(255,0,255);
        Subgraph_Bottom.DrawZeros = false;

        sc.ScaleRangeType = SCALE_INDEPENDENT;

        return;
    }

    // Do data processing

    Subgraph_Top[sc.ArraySize - 1] = 2;
    Subgraph_Bottom[sc.ArraySize - 1] = 1;
}

/*=====*/
SCSFExport scsf_MovingAverageSimple(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Avg = sc.Subgraph[0];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average - Simple";

        sc.GraphRegion = 0;
        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.AutoLoop = 1;

        Subgraph_Avg.Name = "Avg";
        Subgraph_Avg.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg.PrimaryColor = RGB(0, 255, 0);
        Subgraph_Avg.LineWidth = 1;
        Subgraph_Avg.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_Length.Name = "Length";
        Input_Length.SetInt(10);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);
    }
}

```

```

    return;
}

sc.DataStartIndex = Input_Length.GetInt() - 1;

sc.SimpleMovAvg(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_Avg, Input_Length.GetInt());
}

```

```

/*=====*/

```

```

SCSFExport scsf_MovingAverageSimpleSkipZeros(SCStudyInterfaceRef sc)

```

```

{
    SCSubgraphRef Subgraph_Avg = sc.Subgraph[0];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average - Simple Skip Zeros";

        sc.GraphRegion = 0;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_Avg.Name = "Avg";
        Subgraph_Avg.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg.PrimaryColor = RGB(0, 255, 0);
        Subgraph_Avg.LineWidth = 1;
        Subgraph_Avg.DrawZeros = false;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_Length.Name = "Length";
        Input_Length.SetInt(10);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);
    }
}

```

```

    return;
}

```

```

int LengthInput = Input_Length.GetInt();

```

```

sc.DataStartIndex = LengthInput - 1;

```

```

SCFloatArrayRef InputArray = sc.BaseDataIn[Input_Data.GetInputDataIndex()];

```

```

if (LengthInput <= 0)
    return;

```

```

float Sum = 0;

```

```

int Count = 0;

```

```

if (sc.Index < LengthInput - 1)
    LengthInput = sc.Index + 1;

```

```

for(int BarIndex = sc.Index - LengthInput + 1; BarIndex <= sc.Index; BarIndex++)
{
    if (InputArray[BarIndex] == 0.0f)
        continue;
}

```

```

        Sum += InputArray[BarIndex];
        Count ++;
    }

    if (Count > 0)
        Subgraph_Avg[sc.Index] = Sum / Count;
    else
        Subgraph_Avg[sc.Index] = 0;

    return;
}

/*=====*/
SCSFExport scsf_MovingAverageTimePeriod(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Average = sc.Subgraph[0];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_TimePeriodType = sc.Input[1];
    SCInputRef Input_TimePeriodLength = sc.Input[2];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average - Time Period";

        sc.GraphRegion = 0;
        sc.ValueFormat = 2;
        sc.AutoLoop = 0;

        Subgraph_Average.Name = "Avg";
        Subgraph_Average.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Average.PrimaryColor = RGB(0, 255, 0);
        Subgraph_Average.LineWidth = 1;
        Subgraph_Average.DrawZeros = false;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_TimePeriodType.Name = "Time Period Type";
        Input_TimePeriodType.SetTimePeriodType(TIME_PERIOD_LENGTH_UNIT_DAYS);

        Input_TimePeriodLength.Name = "Time Period Length";
        Input_TimePeriodLength.SetInt(1);

        return;
    }

    SCDateTime TimePeriodLengthAsDateTime = sc.TimePeriodSpan(Input_TimePeriodType.GetIndex(),
Input_TimePeriodLength.GetInt());
    int LengthInput = 0;

    int LookbackStartIndex = sc.GetContainingIndexForSCDateTime(sc.ChartNumber,
(sc.BaseDateTimeln[sc.UpdateStartIndex + 1] - TimePeriodLengthAsDateTime).GetAsDouble());

    SCFloatArrayRef Array_InputData = sc.BaseDataIn[Input_Data.GetInputDataIndex()];

    for(int BarIndex = sc.UpdateStartIndex; BarIndex < sc.ArraySize; BarIndex++)
    {
        float Sum = 0;
        int Count = 0;

        for (int LookbackIndex = LookbackStartIndex; LookbackIndex <= BarIndex; LookbackIndex++)
        {
            if (Array_InputData[LookbackIndex] == 0.0f)

```

```

        continue;

        Sum += Array_InputData[LookbackIndex];
        Count++;
    }

    if (Count > 0)
        Subgraph_Average[BarIndex] = Sum / Count;
    else
        Subgraph_Average[BarIndex] = 0;

    if (BarIndex < sc.ArraySize - 1)
    {
        SCDatetime NewLookbackStartDateTime = sc.BaseDateTimeIn[BarIndex + 1] - TimePeriodLengthAsDateTime;

        while (true)
        {
            SCDatetime CurrentAbsoluteTimeDifference = SCDatetime(sc.BaseDateTimeIn[LookbackStartIndex] -
NewLookbackStartDateTime).GetAbsoluteValue();
            SCDatetime NextAbsoluteTimeDifference = SCDatetime(sc.BaseDateTimeIn[LookbackStartIndex + 1] -
NewLookbackStartDateTime).GetAbsoluteValue();

            if (NextAbsoluteTimeDifference > CurrentAbsoluteTimeDifference)
                break;

            LookbackStartIndex++;
        }
    }
}

return;
}

/*=====*/

SCSFExport scsf_AverageDailyRange(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_ADR = sc.Subgraph[0];

    SCInputRef Input_Length = sc.Input[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Average Daily Range";

        sc.GraphRegion = 1;
        sc.ValueFormat = VALUEFORMAT_INHERITED;

        sc.AutoLoop = 1;

        Subgraph_ADR.Name = "ADR";
        Subgraph_ADR.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_ADR.PrimaryColor = RGB(0, 255, 0);
        Subgraph_ADR.LineWidth = 2;
        Subgraph_ADR.DrawZeros = true;

        Input_Length.Name = "Length";
        Input_Length.SetInt(30);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }
}

```

```

}

sc.DataStartIndex = Input_Length.GetInt() - 1;
Subgraph_ADR.Arrays[0][sc.Index] = sc.High[sc.Index] - sc.Low[sc.Index];

sc.SimpleMovAvg(Subgraph_ADR.Arrays[0], Subgraph_ADR, Input_Length.GetInt());
}
/*=====*/
SCSFExport scsf_OBVWithMovAvg(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_OBV = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Avg = sc.Subgraph[1];
    SCInputRef Input_OBVLength = sc.Input[3];
    SCInputRef Input_MovAvgLength = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "On Balance Volume with Mov Avg";

        sc.ValueFormat = 0;
        sc.AutoLoop = 1;

        Subgraph_OBV.Name = "OBV";
        Subgraph_OBV.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_OBV.PrimaryColor = RGB(0,255,0);
        Subgraph_OBV.DrawZeros = true;

        Subgraph_Avg.Name = "Avg";
        Subgraph_Avg.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg.PrimaryColor = RGB(0,0,255);
        Subgraph_Avg.DrawZeros = true;

        //OBVLength.Name = "OBV Length";
        Input_OBVLength.SetInt(10);
        Input_OBVLength.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_MovAvgLength.Name = "Mov Avg Length";
        Input_MovAvgLength.SetInt(10);
        Input_MovAvgLength.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.DataStartIndex = Input_MovAvgLength.GetInt();

    int i = (sc.Index >= 1 ? sc.Index : 1);

    if(sc.Close[i] > sc.Close[i - 1])
    {
        Subgraph_OBV[i] = Subgraph_OBV[i - 1] + sc.Volume[i];
    }
    else if(sc.Close[i] < sc.Close[i - 1])
    {
        Subgraph_OBV[i] = Subgraph_OBV[i - 1] - sc.Volume[i];
    }
    else
        Subgraph_OBV[i] = Subgraph_OBV[i - 1];

    sc.SimpleMovAvg(Subgraph_OBV, Subgraph_Avg, Input_MovAvgLength.GetInt());
}
/*=====*/
SCSFExport scsf_OpenInterest(SCStudyInterfaceRef sc)
{

```

```

SCSubgraphRef Subgraph_OI = sc.Subgraph[0];

if (sc.SetDefaults)
{
    sc.GraphName = "Open Interest";

    sc.ValueFormat = 0;
    sc.AutoLoop = 1;

    Subgraph_OI.Name = "OI";
    Subgraph_OI.PrimaryColor = RGB(0,255,0);
    Subgraph_OI.DrawStyle = DRAWSTYLE_BAR;
    Subgraph_OI.DrawZeros = false;

    return;
}

Subgraph_OI[sc.Index]= sc.OpenInterest[sc.Index];
}

/*=====*/
SCSFExport scsf_NumberOfTrades(SCStudyInterfaceRef sc)
{
    SCSubgraphRef NumTrades = sc.Subgraph[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Number of Trades";

        sc.ValueFormat = 0;
        sc.AutoLoop = 1;

        NumTrades.Name = "NumTrades";
        NumTrades.PrimaryColor = RGB(0,255,0);
        NumTrades.DrawStyle = DRAWSTYLE_BAR;
        NumTrades.LineWidth= 2;
        NumTrades.DrawZeros = false;

        return;
    }

    NumTrades[sc.Index]= sc.NumberOfTrades[sc.Index];
}

/*=====*/
SCSFExport scsf_VolumeDividedByNumberOfTrades(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_NumTrades = sc.Subgraph[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Volume / NumberTrades";

        sc.ValueFormat = 0;
        sc.AutoLoop = 1;

        Subgraph_NumTrades.Name = "NumTrades";
        Subgraph_NumTrades.PrimaryColor = RGB(0,255,0);
        Subgraph_NumTrades.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_NumTrades.LineWidth= 2;
        Subgraph_NumTrades.DrawZeros = false;

        return;
    }
}

```

```

if (sc.NumberOfTrades[sc.Index] != 0)
    Subgraph_NumTrades[sc.Index] = sc.Volume[sc.Index] / sc.NumberOfTrades[sc.Index];
else
    Subgraph_NumTrades[sc.Index] = 0.0f;
}
/*=====*/
SCSFExport scsf_SumBar(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Open = sc.Subgraph[0];
    SCSubgraphRef High = sc.Subgraph[1];
    SCSubgraphRef Low = sc.Subgraph[2];
    SCSubgraphRef Last = sc.Subgraph[3];
    SCSubgraphRef Volume = sc.Subgraph[4];
    SCSubgraphRef OpenInterest = sc.Subgraph[5];
    SCSubgraphRef OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef HLCAvg = sc.Subgraph[7];
    SCSubgraphRef HLAvg = sc.Subgraph[8];
    SCSubgraphRef BidVol = sc.Subgraph[9];
    SCSubgraphRef AskVol = sc.Subgraph[10];

    SCInputRef InChart2Number = sc.Input[3];
    SCInputRef InChart1Multiplier = sc.Input[4];
    SCInputRef InChart2Multiplier = sc.Input[5];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Sum (Bar)";

        sc.ValueFormat = 2;
        sc.GraphRegion = 1;
        sc.UseGlobalChartColors = 0;
        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        // We are using Manual looping.
        sc.AutoLoop = false;

        Open.Name = "Open";
        Open.DrawStyle = DRAWSTYLE_LINE;
        Open.PrimaryColor = RGB(0,255,0);
        Open.DrawZeros = false;

        High.Name = "High";
        High.DrawStyle = DRAWSTYLE_LINE;
        High.PrimaryColor = RGB(0,255,0);
        High.DrawZeros = false;

        Low.Name = "Low";
        Low.DrawStyle = DRAWSTYLE_LINE;
        Low.PrimaryColor = RGB(0,255,0);
        Low.DrawZeros = false;

        Last.Name = "Last";
        Last.DrawStyle = DRAWSTYLE_LINE;
        Last.PrimaryColor = RGB(0,255,0);
        Last.DrawZeros = false;

        Volume.Name = "Volume";
        Volume.DrawStyle = DRAWSTYLE_IGNORE;
        Volume.PrimaryColor = RGB(255,0,0);
        Volume.DrawZeros = false;

        OpenInterest.Name = "# of Trades / OI";
        OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
        OpenInterest.PrimaryColor = RGB(0,0,255);
    }
}

```



```

OpenInterest.DrawZeros = false;

OHLCAvg.Name = "OHLC Avg";
OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
OHLCAvg.DrawZeros = false;
OHLCAvg.PrimaryColor = RGB(127,0,255);

HLCAvg.Name = "HLC Avg";
HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
HLCAvg.DrawZeros = false;
HLCAvg.PrimaryColor = RGB(0,255,255);

HLAvg.Name = "HL Avg";
HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
HLAvg.DrawZeros = false;
HLAvg.PrimaryColor = RGB(0,127,255);

BidVol.Name = "Bid Vol";
BidVol.DrawStyle = DRAWSTYLE_IGNORE;
BidVol.DrawZeros = false;
BidVol.PrimaryColor = RGB(0,255,0);

AskVol.Name = "Ask Vol";
AskVol.DrawStyle = DRAWSTYLE_IGNORE;
AskVol.DrawZeros = false;
AskVol.PrimaryColor = RGB(0,255,0);


InChart2Number.Name = "Chart 2 Number";
InChart2Number.SetChartNumber(1);

InChart1Multiplier.Name = "Chart 1 Multiplier";
InChart1Multiplier.SetFloat(1.0f);

InChart2Multiplier.Name = "Chart 2 Multiplier";
InChart2Multiplier.SetFloat(1.0f);

return;
}

// Obtain a reference to the Base Data in the specified chart. This call
// is relatively efficient, but it should be called as minimally as
// possible. To reduce the number of calls we have it outside of the
// primary "for" loop in this study function. And we also use Manual
// Looping by not defining sc.AutoLoop = 1. In this way,
// sc.GetChartBaseData is called only once per call to this study
// function. sc.GetChartBaseData is a new function to get all of the Base
// Data arrays with one efficient call.
SCGraphData Chart2BaseData;
sc.GetChartBaseData(-InChart2Number.GetChartNumber(), Chart2BaseData);

for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
{
    int Chart2Index = sc.GetNearestMatchForDateTimeIndex(InChart2Number.GetChartNumber(), Index);

    for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_NUM_TRADES; SubgraphIndex++)
    {
        sc.Subgraph[SubgraphIndex][Index]
            = (sc.BaseDataIn[SubgraphIndex][Index] * InChart1Multiplier.GetFloat())
            + (Chart2BaseData[SubgraphIndex][Chart2Index] * InChart2Multiplier.GetFloat());
    }

    sc.Subgraph[SC_HIGH][Index]
        = max(sc.Subgraph[SC_OPEN][Index],
            max(sc.Subgraph[SC_HIGH][Index],

```

```

        max(sc.Subgraph[SC_LOW][Index], sc.Subgraph[SC_LAST][Index])
    )
);

sc.Subgraph[SC_LOW][Index]
= min(sc.Subgraph[SC_OPEN][Index],
    min(sc.Subgraph[SC_HIGH][Index],
        min(sc.Subgraph[SC_LOW][Index], sc.Subgraph[SC_LAST][Index])
    )
);

sc.CalculateOHLCAverages(Index);
}

SCString Chart1Name = sc.GetStudyNameFromChart(sc.ChartNumber, 0);
SCString Chart2Name = sc.GetStudyNameFromChart(InChart2Number.GetChartNumber(), 0);
sc.GraphName.Format("Sum %s + %s", Chart1Name.GetChars(), Chart2Name.GetChars());
}

/*=====*/
SCSFExport scsf_RatioBar(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
    SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
    SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
    SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];
    SCSubgraphRef Subgraph_Unused11 = sc.Subgraph[11];
    SCSubgraphRef Subgraph_Unused12 = sc.Subgraph[12];

    SCInputRef Input_Chart2Number = sc.Input[3];
    SCInputRef Input_Chart1Multiplier = sc.Input[4];
    SCInputRef Input_Chart2Multiplier = sc.Input[5];
    SCInputRef Input_ZeroOutputWhenNonExactDateMatch = sc.Input[6];
    SCInputRef Input_UseLatestSourceDataForLastBar = sc.Input[7];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Ratio (Bar)";

        sc.UseGlobalChartColors = 0;

        sc.ValueFormat = 2;
        sc.GraphRegion = 1;

        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        sc.GraphUsesChartColors = 1;

        Subgraph_Open.Name = "Open";
        Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Open.PrimaryColor = RGB(0,255,0);
        Subgraph_Open.DrawZeros = false;

        Subgraph_High.Name = "High";
        Subgraph_High.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_High.PrimaryColor = RGB(0,255,0);

```

Subgraph_High.DrawZeros = false;

Subgraph_Low.Name = "Low";
Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Low.PrimaryColor = RGB(0,255,0);
Subgraph_Low.DrawZeros = false;

Subgraph_Last.Name = "Last";
Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Last.PrimaryColor = RGB(0,255,0);
Subgraph_Last.DrawZeros = false;

Subgraph_Volume.Name = "Volume";
Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Volume.PrimaryColor = RGB(255,0,0);
Subgraph_Volume.DrawZeros = false;

Subgraph_OpenInterest.Name = "# of Trades / OI";
Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OpenInterest.PrimaryColor = RGB(0,0,255);
Subgraph_OpenInterest.DrawZeros = false;

Subgraph_OHLCAvg.Name = "OHLC Avg";
Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OHLCAvg.PrimaryColor = RGB(127,0,255);
Subgraph_OHLCAvg.DrawZeros = false;

Subgraph_HLCAvg.Name = "HLC Avg";
Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);
Subgraph_HLCAvg.DrawZeros = false;

Subgraph_HLAvg.Name = "HL Avg";
Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLAvg.PrimaryColor = RGB(0,127,255);
Subgraph_HLAvg.DrawZeros = false;

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);
Subgraph_BidVol.DrawZeros = false;

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);
Subgraph_AskVol.DrawZeros = false;

Subgraph_Unused11.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Unused11.DrawZeros = false;

Subgraph_Unused12.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Unused12.DrawZeros = false;

Input_Chart2Number.Name = "Chart 2 Number";
Input_Chart2Number.SetChartNumber(1);

Input_Chart1Multiplier.Name = "Chart 1 Multiplier";
Input_Chart1Multiplier.SetFloat(1.0f);

Input_Chart2Multiplier.Name = "Chart 2 Multiplier";
Input_Chart2Multiplier.SetFloat(1.0f);

Input_ZeroOutputWhenNonExactDateMatch.Name = "Zero Output When Non Exact Date Match";
Input_ZeroOutputWhenNonExactDateMatch.SetYesNo(false);

```

Input_UseLatestSourceDataForLastBar.Name = "Use Latest Source Data For Last Bar";
Input_UseLatestSourceDataForLastBar.SetYesNo(0);

return;
}

SCGraphData Chart2Arrays;
sc.GetChartBaseData(-Input_Chart2Number.GetChartNumber(), Chart2Arrays);

if (Chart2Arrays[0].GetArraySize() == 0)
return;

for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
{
    int Chart2Index = 0;
    if(Input_ZeroOutputWhenNonExactDateMatch.GetYesNo() == false)
    {
        Chart2Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart2Number.GetChartNumber(), Index);

        //When 'use latest source data for last bar' is set to Yes, replay is not running and at last bar in the destination
        chart, then use the data from the very latest bar in the source chart.
        if(Input_UseLatestSourceDataForLastBar.GetYesNo() && !sc.IsReplayRunning() && Index == sc.ArraySize - 1)
            Chart2Index = Chart2Arrays[0].GetArraySize() - 1;

    }
    else
    {
        Chart2Index = sc.GetExactMatchForSCDateTime(Input_Chart2Number.GetChartNumber(),
sc.BaseDateTimeln[Index]);
        if(Chart2Index == -1)
        {
            for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_HL_AVG; SubgraphIndex++)
            {
                sc.Subgraph[SubgraphIndex][Index] = 0.0;
            }

            continue;
        }
    }

for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_NUM_TRADES; SubgraphIndex++)
{
    if (sc.BaseDataIn[SubgraphIndex][Index] == 0 || Chart2Arrays[SubgraphIndex][Chart2Index] == 0)
    {
        sc.Subgraph[SubgraphIndex][Index] = 0;
        continue;
    }

    sc.Subgraph[SubgraphIndex][Index] = (sc.BaseDataIn[SubgraphIndex][Index] * Input_Chart1Multiplier.GetFloat())
        / (Chart2Arrays[SubgraphIndex][Chart2Index] * Input_Chart2Multiplier.GetFloat());
}

sc.Subgraph[SC_HIGH][Index]
= max(sc.Subgraph[0][Index],
    max(sc.Subgraph[1][Index],
        max(sc.Subgraph[2][Index],sc.Subgraph[3][Index])
    )
);

sc.Subgraph[SC_LOW][Index]
= min(sc.Subgraph[0][Index],

```

```

        min(sc.Subgraph[1][Index],
        min(sc.Subgraph[2][Index],sc.Subgraph[3][Index])
        )
    );

    sc.CalculateOHLCAverages(Index);
}

SCString Chart1Name = sc.GetStudyNameFromChart(sc.ChartNumber, 0);
SCString Chart2Name = sc.GetStudyNameFromChart(Input_Chart2Number.GetChartNumber(), 0);
sc.GraphName.Format("Ratio %s / %s", Chart1Name.GetChars(), Chart2Name.GetChars());
}

/*=====*/
SCSFExport scsf_MultiplyBar(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
    SCSubgraphRef Subgraph_HLAvG = sc.Subgraph[8];
    SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
    SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];
    SCSubgraphRef Subgraph_Unused11 = sc.Subgraph[11];
    SCSubgraphRef Subgraph_Unused12 = sc.Subgraph[12];

    SCInputRef Input_Chart2Number = sc.Input[3];
    SCInputRef Input_Chart1Multiplier = sc.Input[4];
    SCInputRef Input_Chart2Multiplier = sc.Input[5];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Multiply Charts - Bar";

        sc.GraphUsesChartColors = true;

        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.GraphRegion = 1;
        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        Subgraph_Open.Name = "Open";
        Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Open.PrimaryColor = RGB(0,255,0);
        Subgraph_Open.DrawZeros = false;

        Subgraph_High.Name = "High";
        Subgraph_High.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_High.PrimaryColor = RGB(0,255,0);
        Subgraph_High.DrawZeros = false;

        Subgraph_Low.Name = "Low";
        Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Low.PrimaryColor = RGB(0,255,0);
        Subgraph_Low.DrawZeros = false;

        Subgraph_Last.Name = "Last";
        Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Last.PrimaryColor = RGB(0,255,0);
        Subgraph_Last.DrawZeros = false;
    }
}

```

```

Subgraph_Volume.Name = "Volume";
Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Volume.PrimaryColor = RGB(255,0,0);
Subgraph_Volume.DrawZeros = false;

Subgraph_OpenInterest.Name = "# of Trades / OI";
Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OpenInterest.PrimaryColor = RGB(0,0,255);
Subgraph_OpenInterest.DrawZeros = false;

Subgraph_OHLCAvg.Name = "OHLC Avg";
Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OHLCAvg.PrimaryColor = RGB(127,0,255);
Subgraph_OHLCAvg.DrawZeros = false;

Subgraph_HLCAvg.Name = "HLC Avg";
Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);
Subgraph_HLCAvg.DrawZeros = false;

Subgraph_HLAvg.Name = "HL Avg";
Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLAvg.PrimaryColor = RGB(0,127,255);
Subgraph_HLAvg.DrawZeros = false;

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);
Subgraph_BidVol.DrawZeros = false;

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);
Subgraph_AskVol.DrawZeros = false;

Subgraph_Unused11.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Unused11.DrawZeros = false;

Subgraph_Unused12.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Unused12.DrawZeros = false;

Input_Chart2Number.Name = "Chart 2 Number";
Input_Chart2Number.SetChartNumber(1);

Input_Chart1Multiplier.Name = "Chart 1 Multiplier";
Input_Chart1Multiplier.SetFloat(1.0f);

Input_Chart2Multiplier.Name = "Chart 2 Multiplier";
Input_Chart2Multiplier.SetFloat(1.0f);

return;
}

SCGraphData Chart2Arrays;
sc.GetChartData(-Input_Chart2Number.GetChartNumber(), Chart2Arrays);

if (Chart2Arrays[0].GetArraySize() == 0)
    return;

for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
{
    int i1 = sc.GetNearestMatchForDateTimeIndex(Input_Chart2Number.GetChartNumber(), Index);

    for (int i2 = SC_OPEN; i2 <= SC_NUM_TRADES; i2++)
    {

```

```

        sc.Subgraph[i2][Index] = (sc.BaseDataIn[i2][Index] * Input_Chart1Multiplier.GetFloat())
        * (Chart2Arrays[i2][i1] * Input_Chart2Multiplier.GetFloat());
    }

    sc.Subgraph[1][Index]
    = max(sc.Subgraph[0][Index],
        max(sc.Subgraph[1][Index],
            max(sc.Subgraph[2][Index],sc.Subgraph[3][Index])
        )
    );

    sc.Subgraph[2][Index]
    = min(sc.Subgraph[0][Index],
        min(sc.Subgraph[1][Index],
            min(sc.Subgraph[2][Index],sc.Subgraph[3][Index])
        )
    );

    sc.CalculateOHLCAverages(Index);
}

SCString Chart1Name = sc.GetStudyNameFromChart(sc.ChartNumber, 0);
SCString Chart2Name = sc.GetStudyNameFromChart(Input_Chart2Number.GetChartNumber(), 0);
sc.GraphName.Format("Multiply %s * %s", Chart1Name.GetChars(), Chart2Name.GetChars());
}

/*=====*/
SCSFExport scsf_OnBalanceVolume(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_OBV = sc.Subgraph[0];
    SCInputRef Input_ResetAtStartOfTradingDay = sc.Input[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "On Balance Volume";
        sc.ValueFormat = 0;
        sc.AutoLoop = 0;

        Subgraph_OBV.Name = "OBV";
        Subgraph_OBV.PrimaryColor = RGB(0,255,0);
        Subgraph_OBV.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_OBV.DrawZeros = true;

        Input_ResetAtStartOfTradingDay.Name = "Reset at Start of Trading Day";
        Input_ResetAtStartOfTradingDay.SetYesNo(false);

        return;
    }

    sc.DataStartIndex = 1;

    for (int Index = max(1, sc.UpdateStartIndex); Index < sc.ArraySize; Index++)
    {
        bool NeedReset = false;

        if (Input_ResetAtStartOfTradingDay.GetYesNo())
        {
            SCDateTime PriorTradingDayDate = sc.GetTradingDayDate(sc.BaseDateTimeIn[Index - 1]);
            SCDateTime TradingDayDate = sc.GetTradingDayDate(sc.BaseDateTimeIn[Index]);
            NeedReset = PriorTradingDayDate != TradingDayDate;
        }

        if (NeedReset)

```

```

{
    if (sc.BaseDataIn[SC_LAST][Index] > sc.BaseDataIn[SC_LAST][Index - 1])
        Subgraph_OBV[Index] = sc.BaseDataIn[SC_VOLUME][Index];
    else if (sc.BaseDataIn[SC_LAST][Index] < sc.BaseDataIn[SC_LAST][Index - 1])
        Subgraph_OBV[Index] = sc.BaseDataIn[SC_VOLUME][Index] * -1;
    else // Equal
        Subgraph_OBV[Index] = 0;
}
else
{
    if (sc.BaseDataIn[SC_LAST][Index] > sc.BaseDataIn[SC_LAST][Index - 1])
        Subgraph_OBV[Index] = Subgraph_OBV[Index - 1] + sc.BaseDataIn[SC_VOLUME][Index];
    else if (sc.BaseDataIn[SC_LAST][Index] < sc.BaseDataIn[SC_LAST][Index - 1])
        Subgraph_OBV[Index] = Subgraph_OBV[Index - 1] - sc.BaseDataIn[SC_VOLUME][Index];
    else // Equal
        Subgraph_OBV[Index] = Subgraph_OBV[Index - 1];
}
}
}

/*=====*/
SCSFExport scsf_PriceOpenInterestVolume(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_POIV = sc.Subgraph[0];
    SCFloatArrayRef Array_TrueHigh = sc.Subgraph[0].Arrays[0];
    SCFloatArrayRef Array_TrueLow = sc.Subgraph[0].Arrays[1];
    SCFloatArrayRef Array_IntermediateCalculation = sc.Subgraph[0].Arrays[2];
    SCFloatArrayRef Array_OBV = sc.Subgraph[0].Arrays[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Price, Open Interest, Volume";
        sc.ValueFormat = 0;
        sc.AutoLoop = 1;

        Subgraph_POIV.Name = "POIV";
        Subgraph_POIV.PrimaryColor = RGB(0, 255, 0);
        Subgraph_POIV.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_POIV.DrawZeros = true;

        return;
    }

    sc.DataStartIndex = 1;

    if (sc.Index < 1)
        return;

    //CumulativeSum (Open Interest * (Close - Close.1) / (True High - True Low)) + OBV

    sc.OnBalanceVolume(sc.BaseDataIn, Array_OBV);

    float TrueHighLowDifference = sc.GetTrueHigh(sc.BaseDataIn) - sc.GetTrueLow(sc.BaseDataIn);

    if (TrueHighLowDifference != 0)
    {
        Array_IntermediateCalculation[sc.Index]
            = (
                (sc.BaseDataIn[SC_OPEN_INTEREST][sc.Index]
                 * (sc.BaseDataIn[SC_LAST][sc.Index] - sc.BaseDataIn[SC_LAST][sc.Index - 1])
                )
                / TrueHighLowDifference
            )
            + Array_OBV[sc.Index];
    }
}

```



```

        sc.CumulativeSummation(Array_IntermediateCalculation, Subgraph_POIV);
    }
    else
    {
        Subgraph_POIV[sc.Index] = Subgraph_POIV[sc.Index - 1];
    }
}

/*=====*/
SCSFExport scsf_Line(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Line = sc.Subgraph[0];
    SCInputRef Input_Value = sc.Input[3];
    SCInputRef Input_NumberOfDaysToCalculate = sc.Input[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Line";

        sc.GraphRegion = 1;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_Line.Name = "Line";
        Subgraph_Line.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Line.PrimaryColor = RGB(0,255,0);
        Subgraph_Line.DrawZeros = 1;

        Input_Value.Name = "Value";
        Input_Value.SetFloat(1.0f);

        Input_NumberOfDaysToCalculate.Name = "Number of Days to Calculate";
        Input_NumberOfDaysToCalculate.SetInt(0);

        sc.DataStartIndex = 0;

        return;
    }

    if(Input_NumberOfDaysToCalculate.GetInt() > 0)
    {
        Subgraph_Line.DrawZeros = 0;

        const SCDatetime TradingDayDateForLastBar = sc.GetTradingDayDate(sc.BaseDateTimeIn[sc.ArraySize - 1]);

        const SCDatetime TradingDayDateForBar = sc.GetTradingDayDate(sc.BaseDateTimeIn[sc.Index]);

        if ((TradingDayDateForLastBar.GetDate() - TradingDayDateForBar.GetDate() + 1)
        > Input_NumberOfDaysToCalculate.GetInt())
        {
            return;
        }
    }

    Subgraph_Line[sc.Index] = Input_Value.GetFloat();
}

/*=====*/
SCSFExport scsf_AverageTrueRange(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_ATR = sc.Subgraph[0];
    SCInputRef Subgraph_MAType = sc.Input[0];
    SCInputRef Subgraph_Length = sc.Input[3];

    if (sc.SetDefaults)

```

```

{
    sc.GraphName = "Average True Range";

    sc.GraphRegion = 1;
    sc.ValueFormat = VALUEFORMAT_INHERITED;
    sc.AutoLoop = 1;

    Subgraph_ATR.Name = "ATR";
    Subgraph_ATR.DrawZeros = false;
    Subgraph_ATR.PrimaryColor = RGB(0,255,0);
    Subgraph_ATR.DrawStyle = DRAWSTYLE_LINE;

    Subgraph_MAType.Name = "Moving Average Type";
    Subgraph_MAType.SetMovAvgType(MOVAVGTYPE_SIMPLE);

    Subgraph_Length.Name = "Moving Average Length";
    Subgraph_Length.SetInt(14);
    Subgraph_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

    return;
}

sc.DataStartIndex = Subgraph_Length.GetInt() - 1;

sc.ATR(sc.BaseDataIn, Subgraph_ATR, Subgraph_Length.GetInt(), Subgraph_MAType.GetMovAvgType());
}

/*=====*/
SCSFExport scsf_AverageTrueRangeExample(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_ATR = sc.Subgraph[0];
    SCInputRef Subgraph_Length = sc.Input[1];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Average True Range Example";

        sc.GraphRegion = 1;
        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.AutoLoop = 1; //Automatic looping

        Subgraph_ATR.Name = "ATR";
        Subgraph_ATR.DrawZeros = false;
        Subgraph_ATR.PrimaryColor = RGB(0,255,0);
        Subgraph_ATR.DrawStyle = DRAWSTYLE_LINE;

        Subgraph_Length.Name = "Moving Average Length";
        Subgraph_Length.SetInt(14);
        Subgraph_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.DataStartIndex = Subgraph_Length.GetInt() - 1;

    sc.ATR(sc.BaseDataIn, Subgraph_ATR.Arrays[0], Subgraph_ATR, Subgraph_Length.GetInt(),
    MOVAVGTYPE_SIMPLE);

    float TrueRange = Subgraph_ATR.Arrays[0][sc.Index];
}

/*=====*/
SCSFExport scsf_TrueRange(SCStudyInterfaceRef sc)
{

```

```

SCSubgraphRef Subgraph_TR = sc.Subgraph[0];

if (sc.SetDefaults)
{
    sc.GraphName = "True Range";

    sc.GraphRegion = 1;
    sc.ValueFormat = VALUEFORMAT_INHERITED;
    sc.AutoLoop = 1;

    Subgraph_TR.Name = "TR";
    Subgraph_TR.DrawZeros = false;
    Subgraph_TR.PrimaryColor = RGB(0,255,0);
    Subgraph_TR.DrawStyle = DRAWSTYLE_LINE;

    return;
}

sc.TrueRange(sc.BaseDataIn, Subgraph_TR);
}

/*=====*/
SCSFExport scsf_NormalizedAverageTrueRange(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_ATR = sc.Subgraph[0];
    SCSubgraphRef Subgraph_NATR = sc.Subgraph[1];

    SCInputRef Input_MAType = sc.Input[0];
    SCInputRef Input_Length = sc.Input[1];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Average True Range - Normalized";

        sc.GraphRegion = 1;
        sc.ValueFormat = 3;// VALUEFORMAT_INHERITED;
        sc.AutoLoop = 1;

        Subgraph_ATR.Name = "ATR";
        Subgraph_ATR.DrawStyle = DRAWSTYLE_IGNORE;

        Subgraph_NATR.Name = "NATR";
        Subgraph_NATR.DrawZeros = false;
        Subgraph_NATR.PrimaryColor = RGB(0, 255, 0);
        Subgraph_NATR.DrawStyle = DRAWSTYLE_LINE;

        Input_MAType.Name = "Moving Average Type";
        Input_MAType.SetMovAvgType(MOVAVGTYPE_SIMPLE);

        Input_Length.Name = "Moving Average Length";
        Input_Length.SetInt(14);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.DataStartIndex = Input_Length.GetInt() - 1;
    sc.ATR(sc.BaseDataIn, Subgraph_ATR, Input_Length.GetInt(), Input_MAType.GetMovAvgType());

    if (sc.BaseData[SC_LAST][sc.Index] == 0)
        Subgraph_NATR[sc.Index] = 0;
    else

```

```

        Subgraph_NATR[sc.Index] = 100 * Subgraph_ATR[sc.Index] / sc.BaseData[SC_LAST][sc.Index];
    }
/*=====*/
SCSFExport scsf_Vortex(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_VIPlus = sc.Subgraph[0];
    SCSubgraphRef Subgraph_VIMinus = sc.Subgraph[1];

    SCInputRef Input_VortexLength = sc.Input[1];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Vortex";

        sc.GraphRegion = 1;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;

        Subgraph_VIPlus.Name = "VI+";
        Subgraph_VIPlus.DrawZeros = false;
        Subgraph_VIPlus.PrimaryColor = RGB(0, 128, 255);
        Subgraph_VIPlus.DrawStyle = DRAWSTYLE_LINE;

        Subgraph_VIMinus.Name = "VI-";
        Subgraph_VIMinus.DrawZeros = false;
        Subgraph_VIMinus.PrimaryColor = COLOR_CRIMSON;
        Subgraph_VIMinus.DrawStyle = DRAWSTYLE_LINE;

        Input_VortexLength.Name = "Vortex Length";
        Input_VortexLength.SetInt(21);
        Input_VortexLength.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.Vortex(sc.BaseDataIn, Subgraph_VIPlus, Subgraph_VIMinus, Input_VortexLength.GetInt());
}

/*=====*/
SCSFExport scsf_DailyRangeBand(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_UpperBand = sc.Subgraph[0];
    SCSubgraphRef Subgraph_LowerBand = sc.Subgraph[1];

    SCInputRef Input_DailyRangeStudy = sc.Input[0];
    SCInputRef Input_GapAdjustWithYesterdaysClose = sc.Input[1];
    SCInputRef Input_NumDaysToDisplay = sc.Input[2];
    SCInputRef Input_BandMultiplier = sc.Input[3];
    SCInputRef Input_UseDailyHighLow = sc.Input[4];
    SCInputRef Input_DailyRangeDisplacement = sc.Input[5];
    SCInputRef Input_StopUpdatingOnRangeMet = sc.Input[6];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Daily Range Bands";

        sc.GraphRegion = 0;
        sc.AutoLoop = 0; // Manual Looping
        sc.DrawZeros = 0;

        // Subgraphs
        Subgraph_UpperBand.Name = "Upper Range Band";
        Subgraph_UpperBand.DrawZeros = false;
        Subgraph_UpperBand.PrimaryColor = RGB(0, 128, 192);
    }
}

```

```

Subgraph_UpperBand.DrawStyle = DRAWSTYLE_STAIR_STEP;

Subgraph_LowerBand.Name = "Lower Range Band";
Subgraph_LowerBand.DrawZeros = false;
Subgraph_LowerBand.PrimaryColor = RGB(192,0,0);
Subgraph_LowerBand.DrawStyle = DRAWSTYLE_STAIR_STEP;

// Inputs
Input_DailyRangeStudy.Name = "Range Study Reference Subgraph";
Input_DailyRangeStudy.SetChartStudySubgraphValues(1, 1, 0);

Input_GapAdjustWithYesterdaysClose.Name = "Gap Adjust With Yesterdays Close";
Input_GapAdjustWithYesterdaysClose.SetYesNo(1);

Input_NumDaysToDisplay.Name = "Number of Days To Display (0=All)";
Input_NumDaysToDisplay.SetInt(0);
Input_NumDaysToDisplay.SetIntLimits(0,MAX_STUDY_LENGTH);

Input_BandMultiplier.Name = "Band Multiplier";
Input_BandMultiplier.SetFloat(1.0f);
Input_BandMultiplier.SetFloatLimits(0, FLT_MAX);

Input_UseDailyHighLow.Name = "Use High/Low From Historical Daily Chart";
Input_UseDailyHighLow.SetYesNo(0);

Input_DailyRangeDisplacement.Name = "Daily Range Displacement";
Input_DailyRangeDisplacement.SetInt(0);
Input_DailyRangeDisplacement.SetIntLimits(0, 1000);

Input_StopUpdatingOnRangeMet.Name = "Stop Updating High/Low When Projected Range Met";
Input_StopUpdatingOnRangeMet.SetYesNo(0);

return;
}

float& TodaysHigh = sc.GetPersistentFloat(1);
float& TodaysLow = sc.GetPersistentFloat(2);
SCDateTime& CurrentSessionStartDateTime = sc.GetPersistentSCDateTime(1);

int DailyChartNumber = Input_DailyRangeStudy.GetChartNumber();

SCGraphData BaseData;
sc.GetChartBaseData(DailyChartNumber, BaseData);

SCFloatArrayRef DailyClose = BaseData[SC_LAST];
SCFloatArrayRef DailyHigh = BaseData[SC_HIGH];
SCFloatArrayRef DailyLow = BaseData[SC_LOW];

SCFloatArray DailyBandRange;
sc.GetStudyArrayFromChartUsingID(Input_DailyRangeStudy.GetChartStudySubgraphValues(), DailyBandRange);

for (int BarIndex = sc.UpdateStartIndex; BarIndex < sc.ArraySize; BarIndex++)
{
    int TradeDate = sc.GetTradingDayDate(sc.BaseDateTimeln[BarIndex]);

    bool NewDay = BarIndex == 0 || sc.BaseDateTimeln[BarIndex] >= CurrentSessionStartDateTime +
SCDateTime::DAYS(1);

    if (NewDay)
    {
        TodaysHigh = sc.High[BarIndex];
        TodaysLow = sc.Low[BarIndex];
        CurrentSessionStartDateTime = sc.GetStartDateTimeForTradingDate(TradeDate);
    }
}

```

```

    if (BarIndex != 0 && Input_NumDaysToDisplay.GetInt() != 0)
    {
        SCDateTime FirstCalcDateTime = CurrentSessionStartDateTime -
SCDateTime::DAYS(Input_NumDaysToDisplay.GetInt() - 1);
        int FirstCalcIndex = sc.GetContainingIndexForSCDateTime(sc.ChartNumber,
FirstCalcDateTime.GetAsDouble());
        for (int Index=0; Index<FirstCalcIndex; ++Index)
        {
            Subgraph_UpperBand[Index] = 0;
            Subgraph_LowerBand[Index] = 0;
        }
    }
}

int DailyIndex = sc.GetFirstIndexForDate(DailyChartNumber, TradeDate);

if (Input_UseDailyHighLow.GetYesNo())
{
    TodaysHigh = DailyHigh[DailyIndex];
    TodaysLow = DailyLow[DailyIndex];
}
else
{
    if (sc.High[BarIndex] > TodaysHigh)
        TodaysHigh = sc.High[BarIndex];
    if (sc.Low[BarIndex] < TodaysLow)
        TodaysLow = sc.Low[BarIndex];
}

float RangeStudyValue = 0;
if (DailyIndex >= Input_DailyRangeDisplacement.GetInt())
    RangeStudyValue = DailyBandRange[DailyIndex-Input_DailyRangeDisplacement.GetInt()];

if (Input_BandMultiplier.GetFloat() != 0)
    RangeStudyValue *= Input_BandMultiplier.GetFloat();

float GapAdjustedHigh = TodaysHigh;
float GapAdjustedLow = TodaysLow;

if (Input_GapAdjustWithYesterdaysClose.GetYesNo())
{
    if (DailyIndex > 0)
    {
        float PriorClose = DailyClose[DailyIndex-1];
        if (PriorClose != 0)
        {
            GapAdjustedLow = min(TodaysLow, PriorClose);
            GapAdjustedHigh = max(TodaysHigh, PriorClose);
        }
    }
}

if (Input_StopUpdatingOnRangeMet.GetYesNo() && !NewDay)
{
    if ((Subgraph_UpperBand[BarIndex-1] - Subgraph_LowerBand[BarIndex-1]) <= RangeStudyValue)
    {
        Subgraph_UpperBand[BarIndex] = Subgraph_UpperBand[BarIndex-1];
        Subgraph_LowerBand[BarIndex] = Subgraph_LowerBand[BarIndex-1];
    }
    else
    {
        if (TodaysHigh >= Subgraph_UpperBand[BarIndex-1])
            Subgraph_LowerBand[BarIndex] = Subgraph_UpperBand[BarIndex-1] - RangeStudyValue;
        else
            Subgraph_LowerBand[BarIndex] = GapAdjustedHigh - RangeStudyValue;
    }
}

```

```

        if (TodaysLow <= Subgraph_LowerBand[BarIndex-1])
            Subgraph_UpperBand[BarIndex] = Subgraph_LowerBand[BarIndex-1] + RangeStudyValue;
        else
            Subgraph_UpperBand[BarIndex] = GapAdjustedLow + RangeStudyValue;
    }
}
else
{
    Subgraph_UpperBand[BarIndex] = GapAdjustedLow + RangeStudyValue;
    Subgraph_LowerBand[BarIndex] = GapAdjustedHigh - RangeStudyValue;
}
}
}

/*=====*/
SCSFExport scsf_MovingAverageExponential(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Avg = sc.Subgraph[0];
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average - Exponential";

        sc.GraphRegion = 0;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;

        Subgraph_Avg.Name = "Avg";
        Subgraph_Avg.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg.PrimaryColor = RGB(0,255,0);
        Subgraph_Avg.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_Length.Name = "Length";
        Input_Length.SetInt(10);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.DataStartIndex = Input_Length.GetInt() - 1;

    sc.ExponentialMovAvg(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_Avg, Input_Length.GetInt());
}

/*=====*/
SCSFExport scsf_MovingAverageWeighted(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Avg = sc.Subgraph[0];
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average - Weighted";

        sc.GraphRegion = 0;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;
    }
}

```

```

Subgraph_Avg.Name = "Avg";
Subgraph_Avg.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Avg.PrimaryColor = RGB(0,255,0);
Subgraph_Avg.DrawZeros = true;

Input_Data.Name = "Input Data";
Input_Data.SetInputDataIndex(SC_LAST);

Input_Length.Name = "Length";
Input_Length.SetInt(10);
Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

return;
}

//sc.DataStartIndex = Length.GetInt() - 1;

sc.WeightedMovingAverage(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_Avg, Input_Length.GetInt());
}

/*=====*/
SCSFExport scsf_MovingAverageLeastSquares(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Avg = sc.Subgraph[0];
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average - Linear Regression";

        sc.GraphRegion = 0;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;

        Subgraph_Avg.Name = "Avg";
        Subgraph_Avg.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg.PrimaryColor = RGB(0,255,0);
        Subgraph_Avg.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_Length.Name = "Length";
        Input_Length.SetInt(10);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.DataStartIndex = Input_Length.GetInt() - 1;

    sc.LinearRegressionIndicator(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_Avg, Input_Length.GetInt());
}

/*=====*/
SCSFExport scsf_MovingAverageEnvelope(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_TopBand = sc.Subgraph[0];
    SCSubgraphRef Subgraph_BottomBand = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Average = sc.Subgraph[2];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_PercentageOrFixed = sc.Input[1];
    SCInputRef Input_Percentage = sc.Input[3];

```



```

SCInputRef Input_MAType = sc.Input[4];
SCInputRef Input_MALength = sc.Input[5];
SCInputRef Input_FixedValue = sc.Input[7];

if (sc.SetDefaults)
{
    sc.GraphName = "Moving Average Envelope";

    sc.GraphRegion = 0;
    sc.ValueFormat = 3;
    sc.AutoLoop = 1;

    Subgraph_TopBand.Name = "TopBand";
    Subgraph_TopBand.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_TopBand.PrimaryColor = RGB(0,255,0);
    Subgraph_TopBand.DrawZeros = true;

    Subgraph_BottomBand.Name = "BottomBand";
    Subgraph_BottomBand.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_BottomBand.PrimaryColor = RGB(255,0,255);
    Subgraph_BottomBand.DrawZeros = true;

    Subgraph_Average.Name = "Average";
    Subgraph_Average.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_Average.PrimaryColor = RGB(255,255,0);
    Subgraph_Average.DrawZeros = true;

    Input_Data.Name = "Input Data";
    Input_Data.SetInputDataIndex(SC_LAST);
    Input_Data.DisplayOrder = 1;

    Input_PercentageOrFixed.Name = "Percentage or Fixed Value";
    Input_PercentageOrFixed.SetCustomInputStrings( "Percentage;Fixed Value");
    Input_PercentageOrFixed.SetCustomInputIndex( 0);
    Input_PercentageOrFixed.DisplayOrder = 2;

    Input_Percentage.Name = "Percentage (.01 = 1%)";
    Input_Percentage.SetFloat(0.005f);
    Input_Percentage.DisplayOrder = 3;

    Input_FixedValue.Name = "Fixed Value";
    Input_FixedValue.SetFloat(10.0f);
    Input_FixedValue.DisplayOrder = 4;

    Input_MAType.Name = "Moving Average Type";
    Input_MAType.SetMovAvgType(MOVAVGTYPE_SIMPLE);
    Input_MAType.DisplayOrder = 5;

    Input_MALength.Name = "Moving Average Length";
    Input_MALength.SetInt(10);
    Input_MALength.SetIntLimits(1, MAX_STUDY_LENGTH);
    Input_MALength.DisplayOrder = 6;

    return;
}

sc.MovingAverage(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_Average,
Input_MAType.GetMovAvgType(), Input_MALength.GetInt());

sc.DataStartIndex = Input_MALength.GetInt() - 1;

if (sc.Index < sc.DataStartIndex)
    return;

if (Input_PercentageOrFixed.GetIndex() == 1)//fixed value

```

```

{
    Subgraph_TopBand[sc.Index] = Subgraph_Average[sc.Index] + Input_FixedValue.GetFloat();
    Subgraph_BottomBand[sc.Index] = Subgraph_Average[sc.Index] - Input_FixedValue.GetFloat();
}
else//percentage
{
    Subgraph_TopBand[sc.Index] = Subgraph_Average[sc.Index] * (1 + Input_Percentage.GetFloat());
    Subgraph_BottomBand[sc.Index] = Subgraph_Average[sc.Index] * (1 - Input_Percentage.GetFloat());
}
}

/*=====*/
SCSFExport scsf_MarketFacilitationIndex(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_MFI = sc.Subgraph[0];
    SCInputRef Input_Multiplier = sc.Input[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Market Facilitation Index";

        sc.GraphRegion = 1;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_MFI.Name = "MFI";
        Subgraph_MFI.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_MFI.PrimaryColor = RGB(0,255,0);
        Subgraph_MFI.DrawZeros = true;

        Input_Multiplier.Name = "Multiplier";
        Input_Multiplier.SetFloat(1.0f);

        return;
    }

    Subgraph_MFI[sc.Index] = Input_Multiplier.GetFloat() * (sc.High[sc.Index] - sc.Low[sc.Index]) /
        sc.Volume[sc.Index];
}

/*=====*/
SCSFExport scsf_Momentum(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Momentum = sc.Subgraph[0];
    SCSubgraphRef Subgraph_CenterLine = sc.Subgraph[1];
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];
    SCInputRef Input_MomentumType = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Momentum";

        sc.GraphRegion = 1;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_Momentum.Name = "Momentum";
        Subgraph_Momentum.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Momentum.PrimaryColor = RGB(0, 255, 0);
        Subgraph_Momentum.DrawZeros = true;

        Subgraph_CenterLine.Name = "Line";
        Subgraph_CenterLine.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_CenterLine.PrimaryColor = RGB(128, 128, 128);
    }
}

```

```

Subgraph_CenterLine.DrawZeros = true;

Input_Data.Name = "Input Data";
Input_Data.SetInputDataIndex(SC_LAST);

Input_Length.Name = "Length";
Input_Length.SetInt(10);
Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_MomentumType.Name = "Momentum Type";
Input_MomentumType.SetCustomInputStrings("Difference;Quotient");
Input_MomentumType.SetCustomInputIndex(0);

return;
}

sc.DataStartIndex = Input_Length.GetInt();

if (sc.Index < sc.DataStartIndex)
    return;

const int SelectedIndex = Input_MomentumType.GetIndex();
switch (SelectedIndex)
{
case 0:
{
    Subgraph_Momentum[sc.Index] = sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index] -
sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_Length.GetInt()];

    Subgraph_CenterLine[sc.Index] = 0.0;
}
break;

case 1:
{
    if (sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_Length.GetInt()] != 0.0f)
        Subgraph_Momentum[sc.Index] = (sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index] /
sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_Length.GetInt()]) * 100;
    else
        Subgraph_Momentum[sc.Index] = 0.0f;

    Subgraph_CenterLine[sc.Index] = 100.0;
}
break;
}
}

/*=====*/
SCSFExport scsf_OnBalanceOpenInterest(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_OBOI = sc.Subgraph[0];
    SCInputRef Input_Length = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "On Balance Open Interest";

        sc.AutoLoop = 1;

        Subgraph_OBOI.Name = "OBOI";
        Subgraph_OBOI.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_OBOI.PrimaryColor = RGB(0,255,0);
        Subgraph_OBOI.DrawZeros = true;
    }
}

```

```

    //Length.Name = "Length";
    Input_Length.SetInt(10);
    Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

    return;
}

sc.DataStartIndex = 1;

if (sc.Index == 0)
    return;

if (sc.Close[sc.Index] > sc.Close[sc.Index - 1])
    Subgraph_OBOI[sc.Index] = Subgraph_OBOI[sc.Index - 1] + sc.OpenInterest[sc.Index];
else if (sc.Close[sc.Index] < sc.Close[sc.Index - 1])
    Subgraph_OBOI[sc.Index] = Subgraph_OBOI[sc.Index - 1] - sc.OpenInterest[sc.Index];
else
    Subgraph_OBOI[sc.Index] = Subgraph_OBOI[sc.Index - 1];
}

/*=====*/
SCSFExport scsf_RateOfChangePercentage(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_ROC = sc.Subgraph[0];
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];
    SCInputRef Input_MultFactor = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Rate Of Change - Percentage";

        sc.GraphRegion = 1;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_ROC.Name = "ROC";
        Subgraph_ROC.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_ROC.PrimaryColor = RGB(0,255,0);
        Subgraph_ROC.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_Length.Name = "Length";
        Input_Length.SetInt(10);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_MultFactor.Name = "Multiplication Factor";
        Input_MultFactor.SetFloat(100.0f);
        Input_MultFactor.SetFloatLimits(1.0f, static_cast<float>(MAX_STUDY_LENGTH));

        return;
    }

    sc.DataStartIndex = Input_Length.GetInt();

    if (sc.Index < sc.DataStartIndex)
        return;

    Subgraph_ROC[sc.Index] = (sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index] -
        sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_Length.GetInt()]) /
        sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_Length.GetInt()] *
        Input_MultFactor.GetFloat();
}

```

```

/*=====*/
SCSFExport scsf_KeltnerChannel(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_KeltnerAverage = sc.Subgraph[0];
    SCSubgraphRef Subgraph_TopBand = sc.Subgraph[1];
    SCSubgraphRef Subgraph_BottomBand = sc.Subgraph[2];
    SCSubgraphRef Subgraph_AtrTemp3 = sc.Subgraph[3];
    SCSubgraphRef Subgraph_AtrTemp4 = sc.Subgraph[4];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_KeltnerMALength = sc.Input[3];
    SCInputRef Input_TrueRangeAvgLength = sc.Input[4];
    SCInputRef Input_TopBandMult = sc.Input[5];
    SCInputRef Input_BottomBandMult = sc.Input[6];
    SCInputRef Input_KeltnerMAType = sc.Input[7];
    SCInputRef Input_ATR_MAType = sc.Input[8];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Keltner Channel";

        sc.GraphRegion = 0;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;

        Subgraph_KeltnerAverage.Name = "Keltner Average";
        Subgraph_KeltnerAverage.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_KeltnerAverage.PrimaryColor = RGB(0,255,0);
        Subgraph_KeltnerAverage.DrawZeros = true;

        Subgraph_TopBand.Name = "Top";
        Subgraph_TopBand.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_TopBand.PrimaryColor = RGB(255,0,255);
        Subgraph_TopBand.DrawZeros = true;

        Subgraph_BottomBand.Name = "Bottom";
        Subgraph_BottomBand.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_BottomBand.PrimaryColor = RGB(255,255,0);
        Subgraph_BottomBand.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_OHLC_AVG);

        Input_KeltnerMALength.Name = "Keltner Mov Avg Length";
        Input_KeltnerMALength.SetInt(10);
        Input_KeltnerMALength.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_TrueRangeAvgLength.Name = "True Range Avg Length";
        Input_TrueRangeAvgLength.SetInt(10);
        Input_TrueRangeAvgLength.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_TopBandMult.Name = "Top Band Multiplier";
        Input_TopBandMult.SetFloat(1.6f);

        Input_BottomBandMult.Name="Bottom Band Multiplier";
        Input_BottomBandMult.SetFloat(1.6f);

        Input_KeltnerMAType.Name = "Keltner Mov Avg Type (Center line)";
        Input_KeltnerMAType.SetMovAvgType(MOVAVGTYPE_SIMPLE);

        Input_ATR_MAType.Name = "ATR Mov Avg Type";
        Input_ATR_MAType.SetMovAvgType(MOVAVGTYPE_WILDERS);

        return;
    }
}

```

```

}

sc.DataStartIndex = max(Input_KeltnerMALength.GetInt(), Input_TrueRangeAvgLength.GetInt());

sc.MovingAverage(sc.BaseDataIn[static_cast<int>(Input_Data.GetInputDataIndex())], Subgraph_KeltnerAverage,
Input_KeltnerMAType.GetMovAvgType(), Input_KeltnerMALength.GetInt());

sc.ATR(sc.BaseDataIn, Subgraph_AtrTemp3, Input_TrueRangeAvgLength.GetInt(),
Input_ATR_MAType.GetMovAvgType());
sc.ATR(sc.BaseDataIn, Subgraph_AtrTemp4, Input_TrueRangeAvgLength.GetInt(),
Input_ATR_MAType.GetMovAvgType());

Subgraph_TopBand[sc.Index] = Subgraph_AtrTemp3[sc.Index] * Input_TopBandMult.GetFloat() +
Subgraph_KeltnerAverage[sc.Index];
Subgraph_BottomBand[sc.Index] = Subgraph_KeltnerAverage[sc.Index] - (Subgraph_AtrTemp4[sc.Index] *
Input_BottomBandMult.GetFloat());
}

/*=====*/
SCSFExport scsf_OnBalanceVolumeShortTerm(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_OBV = sc.Subgraph[0];
    SCInputRef Input_Length = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "On Balance Volume - Short Term";

        sc.ValueFormat = 0;
        sc.AutoLoop = 1;

        Subgraph_OBV.Name = "OBV Length";
        Subgraph_OBV.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_OBV.PrimaryColor = RGB(0,255,0);
        Subgraph_OBV.DrawZeros = true;

        Input_Length.Name = "Length";
        Input_Length.SetInt(10);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.OnBalanceVolumeShortTerm(sc.BaseDataIn, Subgraph_OBV, Input_Length.GetInt());
}

/*=====*/
SCSFExport scsf_RSI(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_RSI = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Line1 = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Line2 = sc.Subgraph[2];
    SCSubgraphRef Subgraph_RSIAvg = sc.Subgraph[3];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_RSILength = sc.Input[3];
    SCInputRef Input_RSI_MALength = sc.Input[4];
    SCInputRef Input_Line1Value = sc.Input[5];
    SCInputRef Input_Line2Value = sc.Input[6];
    SCInputRef Input_AvgType = sc.Input[7];
    SCInputRef Input_UseRSIMinus50 = sc.Input[8];

    if (sc.SetDefaults)
    {
        sc.GraphName = "RSI";
    }
}

```

```

sc.GraphRegion = 1;
sc.ValueFormat = 2;
sc.AutoLoop = 1;

Subgraph_RSI.Name = "RSI";
Subgraph_RSI.DrawStyle = DRAWSTYLE_LINE;
Subgraph_RSI.PrimaryColor = RGB(0,255,0);
Subgraph_RSI.DrawZeros = true;

Subgraph_RSIAvg.Name = "RSI Avg";
Subgraph_RSIAvg.DrawStyle = DRAWSTYLE_LINE;
Subgraph_RSIAvg.PrimaryColor = RGB(255, 127, 0);
Subgraph_RSIAvg.DrawZeros = true;

Subgraph_Line1.Name = "Line1";
Subgraph_Line1.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Line1.PrimaryColor = RGB(255,0,255);
Subgraph_Line1.DrawZeros = true;

Subgraph_Line2.Name = "Line2";
Subgraph_Line2.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Line2.PrimaryColor = RGB(255,255,0);
Subgraph_Line2.DrawZeros = true;

Input_Data.Name = "Input Data";
Input_Data.SetInputDataIndex(SC_LAST);

Input_RSILength.Name = "RSI Length";
Input_RSILength.SetInt(10);
Input_RSILength.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_RSI_MALength.Name = "RSI Mov Avg Length";
Input_RSI_MALength.SetInt(3);
Input_RSI_MALength.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_UseRSIMinus50.Name = "Use RSI - 50";
Input_UseRSIMinus50.SetYesNo(0);

Input_Line1Value.Name = "Line1 Value";
Input_Line1Value.SetFloat(70.0f);

Input_Line2Value.Name = "Line2 Value";
Input_Line2Value.SetFloat(30.0f);

Input_AvgType.Name = "Average Type";
Input_AvgType.SetMovAvgType(MOVAVGTYPE_SIMPLE);

return;
}

sc.DataStartIndex = (Input_RSILength.GetInt() + Input_RSI_MALength.GetInt());

sc.RSI(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_RSI, Input_AvgType.GetMovAvgType(),
Input_RSILength.GetInt());

Subgraph_Line1[sc.Index] = Input_Line1Value.GetFloat();
Subgraph_Line2[sc.Index] = Input_Line2Value.GetFloat();

if (Input_UseRSIMinus50.GetYesNo() == 1)
{
    Subgraph_RSI[sc.Index] = Subgraph_RSI[sc.Index] - 50.0f;
    Subgraph_Line1[sc.Index] = Input_Line1Value.GetFloat() - 50.0f;
    Subgraph_Line2[sc.Index] = Input_Line2Value.GetFloat() - 50.0f;
}

```

```

    sc.MovingAverage(Subgraph_RSI, Subgraph_RSIAvg, Input_AvgType.GetMovAvgType(),
Input_RSI_MALength.GetInt());
}

```

```

/*=====*/

```

```

SCSFExport scsf_AdaptiveRSIMovingAverage(SCStudyInterfaceRef sc)

```

```

{
    SCSubgraphRef Subgraph_ARSI = sc.Subgraph[0];
    SCSubgraphRef Subgraph_RSI = sc.Subgraph[1]; // subgraph and arrays used for base RSI calc

```

```

    SCFloatArrayRef Array_Price = Subgraph_ARSI.Arrays[0];
    SCFloatArrayRef Array_RSISmoothed = Subgraph_ARSI.Arrays[1];

```

```

    SCInputRef Input_ARSIPeriod = sc.Input[0];
    SCInputRef Input_ARSIMAType = sc.Input[1];
    SCInputRef Input_Data = sc.Input[2];
    SCInputRef Input_PriceSetSmoothing = sc.Input[3];
    SCInputRef Input_PriceSmoothingPeriod = sc.Input[4];
    SCInputRef Input_PriceSmoothingMAType = sc.Input[5];
    SCInputRef Input_RSISetSmoothing = sc.Input[6];
    SCInputRef Input_RSISmoothingPeriod = sc.Input[7];
    SCInputRef Input_RSISmoothingMAType = sc.Input[8];
    SCInputRef Input_ARSI scaleFactor = sc.Input[9];

```

```

    if (sc.SetDefaults)
    {
        sc.GraphName = "Adaptive RSI Moving Average With Smoothing";

```

```

        // Data for the "Display Study Documentation" button
        sc.StudyDescription = "Adaptive RSI Moving Average With Smoothing. Traditional Adaptive RSI (ARSI) uses straight price and then RSI as the scaling factor without smoothing. The \"Set Price Smoothing?\" option enables cleaning up the price before calculations. If price is really noisy, recommended settings are EMA 3-10. Otherwise leave price smoothing off. The \"Set RSI Smoothing?\" option enables cleaning up some of the RSI noise before ARSI is calculated. It is recommended to smooth RSI out with an EMA 3-7. This allows for less volatility and more consistent movements.<p>To use traditional ARSI without any enhancements, turn off both smoothing options.<p>Observations: When there's a strong reversal, ARSI will go flat and then bend to follow the price. On weaker events, ARSI will go flat while price bounces around above or below and then returns to the ARSI line. Check indicators and wave counts to verify final motion.<p>To visualize what the scaling factor is doing, a separate RSI graph can be added with the selected smoothing values. When RSI is close to 50, the ARSI line will go flat and move less. Further away from 50 will increase the line movement.";

```

```

        sc.StudyVersion = 1;
        sc.GraphRegion = 0; // main price graph

        sc.AutoLoop = 1; // auto looping is enabled

```

```

        // subgraphs
        Subgraph_ARSI.Name = "ARSI";
        Subgraph_ARSI.PrimaryColor = RGB(0, 0, 255);
        Subgraph_ARSI.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_ARSI.LineStyle = LINESTYLE_SOLID;
        Subgraph_ARSI.LineWidth = 2;
        Subgraph_ARSI.DrawZeros = 0;

```

```

        // inputs
        Input_ARSIPeriod.Name = "ARSI Period";
        Input_ARSIPeriod.SetInt(35);
        Input_ARSIPeriod.SetIntLimits(3, 300);
        Input_ARSIPeriod.SetDescription("Number of bars for the period. Useful values are 25 and higher.");

```

```

        Input_ARSIMAType.Name = "ARSI Moving Average Type";
        Input_ARSIMAType.SetMovAvgType(MOVAVGTYPE_EXPONENTIAL);
        Input_ARSIMAType.SetDescription("Moving average for the base RSI calculation.");

```

```

        Input_ARSI scaleFactor.Name = "ARSI Scale Factor";
        Input_ARSI scaleFactor.SetFloat(1.0f);

```



```

Input_ARSScaleFactor.SetFloatLimits(0.1f, 200.0f);
Input_ARSScaleFactor.SetDescription("Additional Scale Factor for Adaptive calculation.");

Input_Data.Name = "Input Data";
Input_Data.SetInputDataIndex(SC_LAST); //default to bar close.
Input_Data.SetDescription("Usually \"Last\" for bar close when viewing stock prices. One of the averages can be
used for forex and other less time synchronized data.");

Input_PriceSetSmoothing.Name = "Set Price Smoothing";
Input_PriceSetSmoothing.SetYesNo(0);
Input_PriceSetSmoothing.SetDescription("Enables extended price smoothing options.");

Input_PriceSmoothingPeriod.Name = "Price Smoothing Period";
Input_PriceSmoothingPeriod.SetInt(3);
Input_PriceSmoothingPeriod.SetIntLimits(2, 50);
Input_PriceSmoothingPeriod.SetDescription("Number of bars for the period. Usually 3-10.");

Input_PriceSmoothingMAType.Name = "Price Smoothing Moving Average Type";
Input_PriceSmoothingMAType.SetMovAvgType(MOVAVGTYPE_EXPONENTIAL);
Input_PriceSmoothingMAType.SetDescription("Usually EMA, but others can be experimented with.");

Input_RSISetSmoothing.Name = "Set RSI Smoothing";
Input_RSISetSmoothing.SetYesNo(1);
Input_RSISetSmoothing.SetDescription("Enables extended RSI smoothing options.");

Input_RSISmoothingPeriod.Name = "RSI Smoothing Period";
Input_RSISmoothingPeriod.SetInt(5);
Input_RSISmoothingPeriod.SetIntLimits(2, 50);
Input_RSISmoothingPeriod.SetDescription("Number of bars for the period. Usually 3-7.");

Input_RSISmoothingMAType.Name = "RSI Smoothing Moving Average Type";
Input_RSISmoothingMAType.SetMovAvgType(MOVAVGTYPE_EXPONENTIAL);
Input_RSISmoothingMAType.SetDescription("Usually EMA, but others can be experimented with.");

//Set printed output value format. Number is the index of the selected format.
sc.ValueFormat = sc.BaseGraphValueFormat;

return;
}

if (sc.Index == 0)
{
    // Set the index of the first array element to begin drawing at
    sc.DataStartIndex = Input_ARSIPeriod.GetInt() - 1;
}

// Handle price smoothing.
if (Input_PriceSetSmoothing.GetYesNo())
{
    sc.MovingAverage(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Array_Price,
Input_PriceSmoothingMAType.GetMovAvgType(), Input_PriceSmoothingPeriod.GetInt() );
}
else
{
    Array_Price[sc.Index] = sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index];
}

// Not enough data yet. Pre-load existing data into array for later calculations.
if (sc.Index < Input_ARSIPeriod.GetInt())
{
    Subgraph_ARSI[sc.Index] = Array_Price[sc.Index];
    return;
}

// Do the ARSI calculations.

```

```

sc.RSI(Array_Price, Subgraph_RSI, Input_ARSIMAType.GetMovAvgType(), Input_ARSIPeriod.GetInt());

float ScalingFactor=0;

if (Input_RSISetSmoothing.GetYesNo())
{
    sc.MovingAverage(Subgraph_RSI, Array_RSISmoothed, Input_RSISmoothingMAType.GetMovAvgType(),
Input_RSISmoothingPeriod.GetInt());
    ScalingFactor = ((Array_RSISmoothed[sc.Index] / 100.0f) - 0.5f) * 2.0f; //range -1.0 to 1.0
    //Note that a MA may overshoot the range a little. For this it doesn't matter.
}
else
{
    ScalingFactor = ((Subgraph_RSI[sc.Index] / 100.0f) - 0.5f) * 2.0f; //range -1.0 to 1.0
}

ScalingFactor *= Input_ARSI scaleFactor.GetFloat();

if (ScalingFactor < 0.0f)
{
    ScalingFactor *= -1.0f; //range must be positive for the next calculation
}

Subgraph_ARSI[sc.Index] = ((Array_Price[sc.Index] - Subgraph_ARSI[sc.Index-1]) * ScalingFactor) +
Subgraph_ARSI[sc.Index-1];
}

/*=====*/
SCSFExport scsf_RSI_W(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_RSIW = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Line1 = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Line2 = sc.Subgraph[2];
    SCSubgraphRef Subgraph_RSIWAvg = sc.Subgraph[3];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_RSILength = sc.Input[3];
    SCInputRef Input_RSI_MALength = sc.Input[4];
    SCInputRef Input_Line1Value = sc.Input[5];
    SCInputRef Input_Line2Value = sc.Input[6];
    SCInputRef Input_AvgType = sc.Input[7];
    SCInputRef Input_UseRSIMinus50 = sc.Input[8];

    if (sc.SetDefaults)
    {
        sc.GraphName = "RSI-W";

        sc.GraphRegion = 1;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_RSIW.Name = "RSI-W";
        Subgraph_RSIW.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_RSIW.PrimaryColor = RGB(0,255,0);
        Subgraph_RSIW.DrawZeros = true;

        Subgraph_RSIWAvg.Name = "RSI-W Avg";
        Subgraph_RSIWAvg.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_RSIWAvg.PrimaryColor = RGB(255, 127, 0);
        Subgraph_RSIWAvg.DrawZeros = true;

        Subgraph_Line1.Name = "Line1";
        Subgraph_Line1.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Line1.PrimaryColor = RGB(255,0,255);
    }
}

```

```

Subgraph_Line1.DrawZeros = true;

Subgraph_Line2.Name = "Line2";
Subgraph_Line2.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Line2.PrimaryColor = RGB(255,255,0);
Subgraph_Line2.DrawZeros = true;

Input_Data.Name = "Input Data";
Input_Data.SetInputDataIndex(SC_LAST);

Input_RSILength.Name = "RSI Length";
Input_RSILength.SetInt(10);
Input_RSILength.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_RSI_MALength.Name = "RSI Mov Avg Length";
Input_RSI_MALength.SetInt(3);
Input_RSI_MALength.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_UseRSIMinus50.Name = "Use RSI - 50";
Input_UseRSIMinus50.SetYesNo(0);

Input_Line1Value.Name = "Line1 Value";
Input_Line1Value.SetFloat(70.0f);

Input_Line2Value.Name = "Line2 Value";
Input_Line2Value.SetFloat(30.0f);

Input_AvgType.Name = "Average Type";
Input_AvgType.SetMovAvgType(MOVAVGTYPE_WILDERS);

return;
}

sc.DataStartIndex = (Input_RSILength.GetInt() + Input_RSI_MALength.GetInt());

sc.RSI(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_RSIW, Input_AvgType.GetMovAvgType(),
Input_RSILength.GetInt());

Subgraph_Line1[sc.Index] = Input_Line1Value.GetFloat();
Subgraph_Line2[sc.Index] = Input_Line2Value.GetFloat();

if (Input_UseRSIMinus50.GetYesNo() == 1)
{
    Subgraph_RSIW[sc.Index] = Subgraph_RSIW[sc.Index] - 50.0f;
    Subgraph_Line1[sc.Index] = Input_Line1Value.GetFloat() - 50.0f;
    Subgraph_Line2[sc.Index] = Input_Line2Value.GetFloat() - 50.0f;
}

sc.MovingAverage(Subgraph_RSIW, Subgraph_RSIWAvg, Input_AvgType.GetMovAvgType(),
Input_RSI_MALength.GetInt());
}

/*=====*/
SCSFExport scsf_AccumulationDistributionFlow(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_AccumulationDistribution = sc.Subgraph[0];
    SCSubgraphRef Subgraph_ADFlowAvg = sc.Subgraph[1];
    SCInputRef Input_MALength = sc.Input[3];
    SCInputRef Input_UsePrevClose = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Accumulation/Distribution Flow";
    }

```

```

sc.ValueFormat = 2;
sc.AutoLoop = 1;

Subgraph_AccumulationDistribution.Name = "A/D Flow";
Subgraph_AccumulationDistribution.DrawStyle = DRAWSTYLE_LINE;
Subgraph_AccumulationDistribution.PrimaryColor = RGB(0,255,0);
Subgraph_AccumulationDistribution.DrawZeros = true;

Subgraph_ADFlowAvg.Name = "A/D Flow Avg";
Subgraph_ADFlowAvg.DrawStyle = DRAWSTYLE_LINE;
Subgraph_ADFlowAvg.PrimaryColor = RGB(255,0,255);
Subgraph_ADFlowAvg.DrawZeros = true;

Input_MALength.Name = "Moving Average Length";
Input_MALength.SetInt(10);
Input_MALength.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_UsePrevClose.Name = "Use Previous Close";
Input_UsePrevClose.SetYesNo(true);

return;
}

sc.DataStartIndex = Input_MALength.GetInt();

if (sc.Index == 0)
{
    Subgraph_AccumulationDistribution[sc.Index] = 5000.0f;
    return;
}

float OpenOrClose = 0.0f;
if (Input_UsePrevClose.GetYesNo())
{
    OpenOrClose = sc.Close[sc.Index - 1];
}
else
{
    OpenOrClose = sc.Open[sc.Index];
}

if (sc.High[sc.Index] - sc.Low[sc.Index] == 0.0f)
{
    Subgraph_AccumulationDistribution[sc.Index] = Subgraph_AccumulationDistribution[sc.Index - 1];
}
else
{
    Subgraph_AccumulationDistribution[sc.Index] = ((sc.Close[sc.Index] - OpenOrClose) / (sc.High[sc.Index] -
sc.Low[sc.Index]) *
    sc.Volume[sc.Index]) + Subgraph_AccumulationDistribution[sc.Index - 1];
}

sc.SimpleMovAvg(Subgraph_AccumulationDistribution, Subgraph_ADFlowAvg, Input_MALength.GetInt());
}

/*=====*/
SCSFExport scsf_AccumulationDistribution(SCStudyInterfaceRef sc)
{

//Subgraph references
SCSubgraphRef Subgraph_AccumulationDistribution = sc.Subgraph[0];

if (sc.SetDefaults)
{

```

```

sc.GraphName = "Accumulation/Distribution";

sc.ValueFormat = 3;
sc.AutoLoop = 1;

Subgraph_AccumulationDistribution.Name = "A/D";
Subgraph_AccumulationDistribution.DrawStyle = DRAWSTYLE_LINE;
Subgraph_AccumulationDistribution.PrimaryColor = RGB(0,255,0);
Subgraph_AccumulationDistribution.DrawZeros = true;

return;
}

sc.DataStartIndex = 0;

float RangeVolume = (sc.High[sc.Index] - sc.Low[sc.Index]) * sc.Volume[sc.Index];

//In the case where RangeVolume equals 0 at sc.Index = 0, then AccumulationDistribution[sc.Index-1] is going to equal
0. Otherwise, it is the prior AccumulationDistribution value.
if(RangeVolume == 0)
    Subgraph_AccumulationDistribution[sc.Index] = Subgraph_AccumulationDistribution[sc.Index - 1];
else
    // Accumulation/Distribution = ((Close - Low) - (High - Close)) / (High - Low) * Bar Volume
    Subgraph_AccumulationDistribution[sc.Index] = ((sc.Close[sc.Index] - sc.Low[sc.Index]) - (sc.High[sc.Index] -
sc.Close[sc.Index])) / RangeVolume;

}

/*=====*/
SCSFExport scsf_VolumeBarRangeRatio(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Ratio = sc.Subgraph[0];

    SCFloatArrayRef Array_High = sc.High;
    SCFloatArrayRef Array_Low = sc.Low;
    SCFloatArrayRef Array_Volume = sc.Volume;

    if (sc.SetDefaults)
    {
        sc.GraphName = "Volume Bar Range Ratio";

        sc.StudyDescription = "Formula: Volume / (High - Low)";
        sc.AutoLoop = 1;

        Subgraph_Ratio.Name = "Ratio";
        Subgraph_Ratio.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_Ratio.PrimaryColor = RGB(0,255,0);

        return;
    }

    float Range = Array_High[sc.Index] - Array_Low[sc.Index];

    if (Range>0.0)
        Subgraph_Ratio[sc.Index] = Array_Volume[sc.Index] / Range;
    else//If we have no range, then use the prior calculated value.
        Subgraph_Ratio[sc.Index] = Subgraph_Ratio[sc.Index-1];

}

/*=====*/
SCSFExport scsf_MovingAverages(SCStudyInterfaceRef sc)

```

```

{
    SCSubgraphRef Subgraph_Avg1 = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Avg2 = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Avg3 = sc.Subgraph[2];

    SCInputRef Input_MAType1 = sc.Input[0];
    SCInputRef Input_Data1 = sc.Input[1];
    SCInputRef Input_Length1 = sc.Input[2];

    SCInputRef Input_MAType2 = sc.Input[3];
    SCInputRef Input_Data2 = sc.Input[4];
    SCInputRef Input_Length2 = sc.Input[5];

    SCInputRef Input_MAType3 = sc.Input[6];
    SCInputRef Input_Data3 = sc.Input[7];
    SCInputRef Input_Length3 = sc.Input[8];

    SCInputRef Input_Version = sc.Input[12];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Averages";

        sc.GraphRegion = 0;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;

        Subgraph_Avg1.Name = "Avg1";
        Subgraph_Avg1.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg1.PrimaryColor = RGB(0,255,0);

        Subgraph_Avg2.Name = "Avg2";
        Subgraph_Avg2.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg2.PrimaryColor = RGB(255,0,255);

        Subgraph_Avg3.Name = "Avg3";
        Subgraph_Avg3.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg3.PrimaryColor = RGB(255,255,0);

        Input_MAType1.Name = "Moving Average Type 1";
        Input_MAType1.SetMovAvgType(MOVAVGTYPE_SIMPLE);

        Input_Data1.Name = "Input Data 1";
        Input_Data1.SetInputDataIndex(SC_LAST);

        Input_Length1.Name = "Length 1";
        Input_Length1.SetInt(5);
        Input_Length1.SetIntLimits(1, INT_MAX);

        Input_MAType2.Name = "Moving Average Type 2";
        Input_MAType2.SetMovAvgType(MOVAVGTYPE_SIMPLE);

        Input_Data2.Name = "Input Data 2";
        Input_Data2.SetInputDataIndex(SC_LAST);

        Input_Length2.Name = "Length 2";
        Input_Length2.SetInt(10);
        Input_Length2.SetIntLimits(1, INT_MAX);

        Input_MAType3.Name = "Moving Average Type 3";
        Input_MAType3.SetMovAvgType(MOVAVGTYPE_SIMPLE);

        Input_Data3.Name = "Input Data 3";
        Input_Data3.SetInputDataIndex(SC_LAST);
    }
}

```

```

    Input_Length3.Name = "Length 3";
    Input_Length3.SetInt(15);
    Input_Length3.SetIntLimits(1, INT_MAX);

    Input_Version.SetInt(2);

    return;
}

if (Input_Version.GetInt() < 2)
{
    uint32_t OldInputData = sc.Input[0].GetInputDataIndex();
    int OldLength1 = sc.Input[3].GetInt();
    int OldLength2 = sc.Input[4].GetInt();
    int OldLength3 = sc.Input[5].GetInt();
    uint32_t OldMAtype = sc.Input[7].GetMovAvgType();

    Input_MAType1.SetMovAvgType(OldMAtype);
    Input_Data1.SetInputDataIndex(OldInputData);
    Input_Length1.SetInt(OldLength1);

    Input_MAType2.SetMovAvgType(OldMAtype);
    Input_Data2.SetInputDataIndex(OldInputData);
    Input_Length2.SetInt(OldLength2);

    Input_MAType3.SetMovAvgType(OldMAtype);
    Input_Data3.SetInputDataIndex(OldInputData);
    Input_Length3.SetInt(OldLength3);

    Input_Version.SetInt(2);
}

//
if(Input_Length1.GetInt() <= 0)
{
    Input_Length1.SetInt(10);
    Input_MAType1.SetMovAvgType(MOVAVGTYPE_SIMPLE);
    Input_Data1.SetInputDataIndex(SC_LAST);
}

if(Input_Length2.GetInt() <= 0)
{
    Input_Length2.SetInt(20);
    Input_MAType2.SetMovAvgType(MOVAVGTYPE_SIMPLE);
    Input_Data2.SetInputDataIndex(SC_LAST);
}

if(Input_Length3.GetInt() <= 0)
{
    Input_Length3.SetInt(50);
    Input_MAType3.SetMovAvgType(MOVAVGTYPE_SIMPLE);
    Input_Data3.SetInputDataIndex(SC_LAST);
}

sc.DataStartIndex = max(Input_Length3.GetInt(), max(Input_Length1.GetInt(), Input_Length2.GetInt()));

sc.MovingAverage(sc.BaseDataIn[Input_Data1.GetInputDataIndex()], Subgraph_Avg1,
Input_MAType1.GetMovAvgType(), Input_Length1.GetInt());
sc.MovingAverage(sc.BaseDataIn[Input_Data2.GetInputDataIndex()], Subgraph_Avg2,
Input_MAType2.GetMovAvgType(), Input_Length2.GetInt());

```

```

    sc.MovingAverage(sc.BaseDataIn[Input_Data3.GetInputDataIndex()], Subgraph_Avg3,
Input_MAType3.GetMovAvgType(), Input_Length3.GetInt());
}

```

```

/*=====*/

```

```

SCSFExport scsf_MovingAverageCrossover(SCStudyInterfaceRef sc)
{

```

```

    SCSubgraphRef Subgraph_Avg1 = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Avg2 = sc.Subgraph[1];
    SCSubgraphRef Subgraph_CrossFromBottom = sc.Subgraph[2];
    SCSubgraphRef Subgraph_CrossFromTop = sc.Subgraph[3];

```

```

    SCInputRef Input_MAType1 = sc.Input[0];
    SCInputRef Input_Data1 = sc.Input[1];
    SCInputRef Input_Length1 = sc.Input[2];

```

```

    SCInputRef Input_MAType2 = sc.Input[3];
    SCInputRef Input_Data2 = sc.Input[4];
    SCInputRef Input_Length2 = sc.Input[5];

```

```

    if (sc.SetDefaults)
    {

```

```

        sc.GraphName = "Moving Average Crossover";

```

```

        sc.GraphRegion = 0;
        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.AutoLoop = 1;

```

```

        Subgraph_Avg1.Name = "Avg1";
        Subgraph_Avg1.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg1.PrimaryColor = RGB(0,255,0);

```

```

        Subgraph_Avg2.Name = "Avg2";
        Subgraph_Avg2.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Avg2.PrimaryColor = RGB(255,0,255);

```

```

        Subgraph_CrossFromBottom.Name = "Cross From Bottom";
        Subgraph_CrossFromBottom.DrawStyle = DRAWSTYLE_ARROW_UP;
        Subgraph_CrossFromBottom.PrimaryColor = RGB(0,255,0);
        Subgraph_CrossFromBottom.LineWidth = 2;

```

```

        Subgraph_CrossFromTop.Name = "Cross From Top";
        Subgraph_CrossFromTop.DrawStyle = DRAWSTYLE_ARROW_DOWN;
        Subgraph_CrossFromTop.PrimaryColor = RGB(255,0,255);
        Subgraph_CrossFromTop.LineWidth = 2;

```

```

        Input_MAType1.Name = "Moving Average Type 1";
        Input_MAType1.SetMovAvgType(MOVAVGTYPE_SIMPLE);

```

```

        Input_Data1.Name = "Input Data 1";
        Input_Data1.SetInputDataIndex(SC_LAST);

```

```

        Input_Length1.Name = "Length 1";
        Input_Length1.SetInt(10);
        Input_Length1.SetIntLimits(1, MAX_STUDY_LENGTH);

```

```

        Input_MAType2.Name = "Moving Average Type 2";
        Input_MAType2.SetMovAvgType(MOVAVGTYPE_SIMPLE);

```

```

        Input_Data2.Name = "Input Data 2";
        Input_Data2.SetInputDataIndex(SC_LAST);

```

```

        Input_Length2.Name = "Length 2";
        Input_Length2.SetInt(20);

```



```

    Input_Length2.SetIntLimits(1, MAX_STUDY_LENGTH);
    return;
}

sc.DataStartIndex = max(Input_Length1.GetInt(), Input_Length2.GetInt());

sc.MovingAverage(sc.BaseDataIn[Input_Data1.GetInputDataIndex()], Subgraph_Avg1,
Input_MAType1.GetMovAvgType(), Input_Length1.GetInt());
sc.MovingAverage(sc.BaseDataIn[Input_Data2.GetInputDataIndex()], Subgraph_Avg2,
Input_MAType2.GetMovAvgType(), Input_Length2.GetInt());

if(Input_Length1.GetInt() <= Input_Length2.GetInt() && sc.CrossOver(Subgraph_Avg1, Subgraph_Avg2) ==
CROSS_FROM_TOP)//Sell
{
    Subgraph_CrossFromBottom[sc.Index] = 0;
    Subgraph_CrossFromTop[sc.Index] = sc.High[sc.Index];
}
else if(Input_Length1.GetInt() <= Input_Length2.GetInt() && sc.CrossOver(Subgraph_Avg1, Subgraph_Avg2) ==
CROSS_FROM_BOTTOM)//Buy
{
    Subgraph_CrossFromBottom[sc.Index] = sc.Low[sc.Index];
    Subgraph_CrossFromTop[sc.Index] = 0;
}
else if(Input_Length1.GetInt() > Input_Length2.GetInt() && sc.CrossOver(Subgraph_Avg1, Subgraph_Avg2) ==
CROSS_FROM_TOP)//Buy
{
    Subgraph_CrossFromBottom[sc.Index] = sc.Low[sc.Index];
    Subgraph_CrossFromTop[sc.Index] = 0;
}
else if(Input_Length1.GetInt() > Input_Length2.GetInt() && sc.CrossOver(Subgraph_Avg1, Subgraph_Avg2) ==
CROSS_FROM_BOTTOM)//Sell
{
    Subgraph_CrossFromBottom[sc.Index] = 0;
    Subgraph_CrossFromTop[sc.Index] = sc.High[sc.Index];
}
else
{
    Subgraph_CrossFromBottom[sc.Index] = 0;
    Subgraph_CrossFromTop[sc.Index] = 0;
}
}
/*=====*/
SCSFExport scsf_STIX(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_STIX = sc.Subgraph[0];
    SCInputRef Input_DecliningIssuesSymbolStudy = sc.Input[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "STIX";
        sc.GraphRegion = 1;
        sc.ValueFormat = 0;

        Subgraph_STIX.Name = "STIX";
        Subgraph_STIX.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_STIX.PrimaryColor = RGB(0,255,0);
        Subgraph_STIX.DrawZeros = true;

        Input_DecliningIssuesSymbolStudy.Name = "Declining Issues Add Additional Symbol Study";
        Input_DecliningIssuesSymbolStudy.SetStudyID(1);

        return;
    }
}

```

```
SCFloatArray Chart2Array;
sc.GetStudyArrayUsingID(Input_DecliningIssuesSymbolStudy.GetStudyID(), SC_LAST, Chart2Array);
```

```
if (Chart2Array.GetArraySize() == 0)
    return;
```

```
for (int BarIndex = sc.UpdateStartIndex; BarIndex < sc.ArraySize; BarIndex++)
{
    if (sc.Close[BarIndex] == 0 || Chart2Array[BarIndex] == 0)
    {
        Subgraph_STIX[BarIndex] = Subgraph_STIX[BarIndex - 1];
    }
    else
    {
        Subgraph_STIX[BarIndex] = sc.Close[BarIndex] / (sc.Close[BarIndex] + Chart2Array[BarIndex]) * 9.0f +
            (Subgraph_STIX[BarIndex - 1] * 0.91f);
    }
}
}
```

```
/*=====*/
```

```
SCSFExport scsf_AwesomeOscillator(SCStudyInterfaceRef sc)
```

```
{
    SCSubgraphRef Subgraph_AO = sc.Subgraph[0];
    SCSubgraphRef Subgraph_ZeroLine = sc.Subgraph[1];
```

```
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_MA1Length = sc.Input[2];
    SCInputRef Input_MA2Length = sc.Input[3];
    SCInputRef Input_MAType = sc.Input[4];
```

```
    if (sc.SetDefaults)
    {
        sc.GraphName = "Awesome Oscillator";
```

```
        sc.GraphRegion = 1;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;
```

```
        Subgraph_AO.Name = "AO";
        Subgraph_AO.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_AO.LineWidth = 2;
        Subgraph_AO.SecondaryColor = RGB(255,0,0);
        Subgraph_AO.AutoColoring = AUTOCOLOR_SLOPE;
        Subgraph_AO.DrawZeros = false;
```

```
        Subgraph_ZeroLine.Name = "Zero Line";
        Subgraph_ZeroLine.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_ZeroLine.LineWidth = 1;
        Subgraph_ZeroLine.PrimaryColor = RGB(255,0,255);
        Subgraph_ZeroLine.DrawZeros = true;
```

```
        Input_Data.SetInputDataIndex(SC_HL_AVG);
```

```
        Input_MA1Length.Name = "Moving Average 1 Length";
        Input_MA1Length.SetInt(34);
        Input_MA1Length.SetIntLimits(1, MAX_STUDY_LENGTH);
```

```
        Input_MA2Length.Name = "Moving Average 2 Length";
        Input_MA2Length.SetInt(5);
        Input_MA2Length.SetIntLimits(1, MAX_STUDY_LENGTH);
```

```
        Input_MAType.Name = "Moving Average Type";
        Input_MAType.SetMovAvgType(MOVAVGTYPE_SIMPLE);
```

```

    return;
}

sc.DataStartIndex = max(Input_MA1Length.GetInt(), Input_MA2Length.GetInt()) - 1;

    sc.MovingAverage(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_AO.Arrays[0],
Input_MAType.GetMovAvgType(), Input_MA1Length.GetInt());
    sc.MovingAverage(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_AO.Arrays[1],
Input_MAType.GetMovAvgType(), Input_MA2Length.GetInt());

    Subgraph_AO.Data[sc.Index] = Subgraph_AO.Arrays[1][sc.Index] - Subgraph_AO.Arrays[0][sc.Index];

}

/*=====*/
SCSFExport scsf_MovingAverageDifference(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_MovAvgDiff = sc.Subgraph[0];
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_MA1Length = sc.Input[2];
    SCInputRef Input_MA2Length = sc.Input[3];
    SCInputRef Input_MovAvgType = sc.Input[4];
    SCInputRef Input_Version = sc.Input[5];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average Difference";

        sc.GraphRegion = 1;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;

        Subgraph_MovAvgDiff.Name = "MovAvg Diff";
        Subgraph_MovAvgDiff.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_MovAvgDiff.SecondaryColor = RGB(255,0,0);
        Subgraph_MovAvgDiff.AutoColoring = AUTOCOLOR_SLOPE;
        Subgraph_MovAvgDiff.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_HL_AVG);

        Input_MA1Length.Name = "Moving Average 1 Length";
        Input_MA1Length.SetInt(30);
        Input_MA1Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_MA2Length.Name = "Moving Average 2 Length";
        Input_MA2Length.SetInt(10);
        Input_MA2Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_MovAvgType.Name = "Moving Average Type";
        Input_MovAvgType.SetMovAvgType(MOVAVGTYPE_SIMPLE);

        Input_Version.SetInt(1);

        return;
    }

    if(Input_Version.GetInt() < 1)
    {
        Input_MovAvgType.SetMovAvgType(MOVAVGTYPE_SIMPLE);
        Input_Version.SetInt(1);
    }
}

```

```

}

sc.DataStartIndex = max(Input_MA1Length.GetInt(), Input_MA2Length.GetInt());

    sc.MovingAverage(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_MovAvgDiff.Arrays[0],
Input_MovAvgType.GetMovAvgType(), Input_MA1Length.GetInt());
    sc.MovingAverage(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_MovAvgDiff.Arrays[1],
Input_MovAvgType.GetMovAvgType(), Input_MA2Length.GetInt());
    Subgraph_MovAvgDiff[sc.Index] = Subgraph_MovAvgDiff.Arrays[0][sc.Index] - Subgraph_MovAvgDiff.Arrays[1]
[sc.Index];
}

/*=====*/

SCSFExport scsf_Volume(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Volume";

        sc.ValueFormat = 0;
        sc.AutoLoop = false;
        sc.ScaleRangeType = SCALE_ZEROBASED;

        Subgraph_Volume.Name = "Volume";
        Subgraph_Volume.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_Volume.AutoColoring = AUTOCOLOR_BASEGRAPH;
        Subgraph_Volume.PrimaryColor = RGB(0, 255, 0);
        Subgraph_Volume.SecondaryColor = RGB(255, 0, 0);
        Subgraph_Volume.DrawZeros = false;
        Subgraph_Volume.LineWidth = 2;

        sc.DisplayStudyInputValues = false;

        return;
    }

    int CalculationStartIndex = sc.GetCalculationStartIndexForStudy();
    if (CalculationStartIndex > sc.UpdateStartIndex)
        CalculationStartIndex = sc.UpdateStartIndex;

    for (int BarIndex = CalculationStartIndex; BarIndex < sc.ArraySize; BarIndex++)
    {
        Subgraph_Volume[BarIndex] = sc.Volume[BarIndex];
    }
}

/*=====*/

SCSFExport scsf_VolumeWithZeroColor(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[0];
    SCSubgraphRef Subgraph_NoPriceChangeColor = sc.Subgraph[1];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Volume with 0 Color";

        sc.ValueFormat = 0;

```

```

sc.AutoLoop = true;
sc.ScaleRangeType = SCALE_ZEROBASED;

Subgraph_Volume.Name = "Volume";
Subgraph_Volume.DrawStyle = DRAWSTYLE_BAR;
Subgraph_Volume.PrimaryColor = RGB(0, 255, 0);
Subgraph_Volume.SecondaryColor = RGB(255, 0, 0);
Subgraph_Volume.SecondaryColorUsed = true;
Subgraph_Volume.DrawZeros = false;
Subgraph_Volume.LineWidth = 2;

Subgraph_NoPriceChangeColor.Name = "No Price Change Color";
Subgraph_NoPriceChangeColor.DrawStyle = DRAWSTYLE_CUSTOM_TEXT;
Subgraph_NoPriceChangeColor.PrimaryColor = RGB(0, 255, 0);
Subgraph_NoPriceChangeColor.DrawZeros = false;
Subgraph_NoPriceChangeColor.DisplayNameValueInWindowsFlags = 0;
sc.DisplayStudyInputValues = false;

return;
}

Subgraph_Volume[sc.Index] = sc.Volume[sc.Index];

if(sc.FormattedEvaluate(sc.BaseData[SC_OPEN][sc.Index], sc.BaseGraphValueFormat, LESS_OPERATOR,
sc.BaseData[SC_LAST][sc.Index], sc.BaseGraphValueFormat) )
    Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume.PrimaryColor;
else if(sc.FormattedEvaluate(sc.BaseData[SC_OPEN][sc.Index], sc.BaseGraphValueFormat, GREATER_OPERATOR,
sc.BaseData[SC_LAST][sc.Index], sc.BaseGraphValueFormat) )
    Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume.SecondaryColor;
else
    Subgraph_Volume.DataColor[sc.Index] = Subgraph_NoPriceChangeColor.PrimaryColor;
}

/*=====*/

SCSFExport scsf_VolumeColoredBasedOnBarCloses(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Volume-Colored Based on Bar Closes";

        sc.ValueFormat = 0;

        sc.AutoLoop = true;
        sc.ScaleRangeType = SCALE_ZEROBASED;

        Subgraph_Volume.Name = "Volume";
        Subgraph_Volume.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_Volume.PrimaryColor = RGB(0, 255, 0);
        Subgraph_Volume.SecondaryColor = RGB(255, 0, 0);
        Subgraph_Volume.SecondaryColorUsed = 1;
        Subgraph_Volume.LineWidth = 2;
        Subgraph_Volume.DrawZeros = false;

        sc.DisplayStudyInputValues = false;

        return;
    }
}

```

```

Subgraph_Volume[sc.Index] = sc.Volume[sc.Index];

if (sc.FormattedEvaluate(sc.Close[sc.Index], sc.BaseGraphValueFormat, GREATER_EQUAL_OPERATOR,
sc.Close[sc.Index - 1], sc.BaseGraphValueFormat))
{
    Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume.PrimaryColor;
}
else
    Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume.SecondaryColor;
}

/*=====*/

SCSFExport scsf_VolumeWithNegativeDownVolume(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Volume - With Negative Down Volume";

        sc.ValueFormat = 0;

        sc.AutoLoop = true;
        sc.ScaleRangeType = SCALE_AUTO;

        Subgraph_Volume.Name = "Volume";
        Subgraph_Volume.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_Volume.PrimaryColor = RGB(0, 255, 0);
        Subgraph_Volume.SecondaryColor = RGB(255, 0, 0);
        Subgraph_Volume.SecondaryColorUsed = 1;
        Subgraph_Volume.LineWidth = 2;
        Subgraph_Volume.DrawZeros = false;

        sc.DisplayStudyInputValues = false;

        return;
    }

    if(sc.Close[sc.Index] >= sc.Open[sc.Index])
    {
        Subgraph_Volume[sc.Index] = sc.Volume[sc.Index];
        Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume.PrimaryColor;
    }
    else
    {
        Subgraph_Volume[sc.Index] = sc.Volume[sc.Index] * -1;
        Subgraph_Volume.DataColor[sc.Index] = Subgraph_Volume.SecondaryColor;
    }
}

/*=====*/

SCSFExport scsf_Spread3Chart(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];

```

```

SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];

SCInputRef Input_InChart2Number = sc.Input[3];
SCInputRef Input_InChart3Number = sc.Input[4];
SCInputRef Input_InChart1Multiplier = sc.Input[5];
SCInputRef Input_InChart2Multiplier = sc.Input[6];
SCInputRef Input_InChart3Multiplier = sc.Input[7];

if (sc.SetDefaults)
{
    sc.GraphName = "Spread - 3 Chart";

    sc.AutoLoop = 0;

    sc.ValueFormat = 2;
    sc.GraphRegion = 1;

    sc.UseGlobalChartColors = 0;
    sc.GraphDrawType = GDT_OHLCBAR;
    sc.StandardChartHeader = 1;

    Subgraph_Open.Name = "Open";
    Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_Open.PrimaryColor = RGB(0,255,0);
    Subgraph_Open.SecondaryColor = RGB(0,255,0);

    Subgraph_High.Name = "High";
    Subgraph_High.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_High.PrimaryColor = RGB(0,255,0);
    Subgraph_High.SecondaryColor = RGB(0,255,0);

    Subgraph_Low.Name = "Low";
    Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_Low.PrimaryColor = RGB(0,255,0);
    Subgraph_Low.SecondaryColor = RGB(0,255,0);

    Subgraph_Last.Name = "Last";
    Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_Last.PrimaryColor = RGB(0,255,0);
    Subgraph_Last.SecondaryColor = RGB(0,255,0);

    Subgraph_Volume.Name = "Volume";
    Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_Volume.PrimaryColor = RGB(255,0,0);

    Subgraph_OpenInterest.Name = "# of Trades / OI";
    Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_OpenInterest.PrimaryColor = RGB(0,0,255);

    Subgraph_OHLCAvg.Name = "OHLC Avg";
    Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_OHLCAvg.PrimaryColor = RGB(0,255,0);

    Subgraph_HLCAvg.Name = "HLC Avg";
    Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);

    Subgraph_HLAvg.Name = "HL Avg";
    Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_HLAvg.PrimaryColor = RGB(0,127,255);

```

```

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);

Input_InChart2Number.Name = "Chart 2 Number";
Input_InChart2Number.SetChartNumber(1);

Input_InChart3Number.Name = "Chart 3 Number";
Input_InChart3Number.SetChartNumber(1);

Input_InChart1Multiplier.Name = "Chart 1 Multiplier";
Input_InChart1Multiplier.SetFloat(1.0f);

Input_InChart2Multiplier.Name = "Chart 2 Multiplier";
Input_InChart2Multiplier.SetFloat(1.0f);

Input_InChart3Multiplier.Name = "Chart 3 Multiplier";
Input_InChart3Multiplier.SetFloat(1.0f);

return;
}

if(sc.IsFullRecalculation && sc.UpdateStartIndex == 0)
{
    SCString Chart1Name = sc.GetChartName(sc.ChartNumber());
    SCString Chart2Name = sc.GetChartName(Input_InChart2Number.GetChartNumber());
    SCString Chart3Name = sc.GetChartName(Input_InChart3Number.GetChartNumber());
    sc.GraphName.Format("Spread - 3 Chart: %s + %s - %s", Chart1Name.GetChars(), Chart2Name.GetChars(),
Chart3Name.GetChars());
}

SCGraphData Chart2BaseData;
sc.GetChartData(-Input_InChart2Number.GetChartNumber(), Chart2BaseData);

SCGraphData Chart3BaseData;
sc.GetChartData(-Input_InChart3Number.GetChartNumber(), Chart3BaseData);

// Iterate through the bars
for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
{
    int Chart2Index = sc.GetNearestMatchForDateTimeIndex(Input_InChart2Number.GetChartNumber(), Index);
    int Chart3Index = sc.GetNearestMatchForDateTimeIndex(Input_InChart3Number.GetChartNumber(), Index);

    // Iterate through the subgraphs
    for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_NUM_TRADES; SubgraphIndex++)
    {
        sc.Subgraph[SubgraphIndex][Index]
            = (sc.BaseDataIn[SubgraphIndex][Index] * Input_InChart1Multiplier.GetFloat())
            + (Chart2BaseData[SubgraphIndex][Chart2Index] * Input_InChart2Multiplier.GetFloat())
            - (Chart3BaseData[SubgraphIndex][Chart3Index] * Input_InChart3Multiplier.GetFloat())
            ;
    }

    sc.Subgraph[SC_HIGH][Index]
        = max(
            sc.Subgraph[SC_OPEN][Index],
            max(
                sc.Subgraph[SC_HIGH][Index],
                max(

```



```

        sc.Subgraph[SC_LOW][Index],
        sc.Subgraph[SC_LAST][Index]
    )
)
);

sc.Subgraph[SC_LOW][Index]
= min(
    sc.Subgraph[SC_OPEN][Index],
    min(
        sc.Subgraph[SC_HIGH][Index],
        min(
            sc.Subgraph[SC_LOW][Index],
            sc.Subgraph[SC_LAST][Index]
        )
    )
);

sc.CalculateOHLCAverages(Index);
}

}

/*=====*/
SCSFExport scsf_SpreadButterfly(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
    SCSubgraphRef Subgraph_HLAv = sc.Subgraph[8];
    SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
    SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];

    SCInputRef Input_UseLatestSourceDataForLastBar = sc.Input[2];
    SCInputRef Input_InChart2Number = sc.Input[3];
    SCInputRef Input_InChart3Number = sc.Input[4];
    SCInputRef Input_InChart1Multiplier = sc.Input[5];
    SCInputRef Input_InChart2Multiplier = sc.Input[6];
    SCInputRef Input_InChart3Multiplier = sc.Input[7];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Spread - Butterfly";

        sc.ValueFormat = 2;
        sc.GraphRegion = 1;

        sc.UseGlobalChartColors = 0;
        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        Subgraph_Open.Name = "Open";
        Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Open.PrimaryColor = RGB(0,255,0);
        Subgraph_Open.SecondaryColor = RGB(0,255,0);

        Subgraph_High.Name = "High";
        Subgraph_High.DrawStyle = DRAWSTYLE_LINE;

```

```

Subgraph_High.PrimaryColor = RGB(0,255,0);
Subgraph_High.SecondaryColor = RGB(0,255,0);

Subgraph_Low.Name = "Low";
Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Low.PrimaryColor = RGB(0,255,0);
Subgraph_Low.SecondaryColor = RGB(0,255,0);

Subgraph_Last.Name = "Last";
Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Last.PrimaryColor = RGB(0,255,0);
Subgraph_Last.SecondaryColor = RGB(0,255,0);

Subgraph_Volume.Name = "Volume";
Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Volume.PrimaryColor = RGB(255,0,0);

Subgraph_OpenInterest.Name = "# of Trades / OI";
Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OpenInterest.PrimaryColor = RGB(0,0,255);

Subgraph_OHLCAvg.Name = "OHLC Avg";
Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OHLCAvg.PrimaryColor = RGB(0,255,0);

Subgraph_HLCAvg.Name = "HLC Avg";
Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);

Subgraph_HLAvg.Name = "HL Avg";
Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLAvg.PrimaryColor = RGB(0,127,255);

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);

Input_UseLatestSourceDataForLastBar.Name = "Use Latest Source Data For Last Bar";
Input_UseLatestSourceDataForLastBar.SetYesNo(0);

Input_InChart2Number.Name = "Chart 2 Number";
Input_InChart2Number.SetChartNumber(1);

Input_InChart3Number.Name = "Chart 3 Number";
Input_InChart3Number.SetChartNumber(1);

Input_InChart1Multiplier.Name = "Chart 1 Multiplier";
Input_InChart1Multiplier.SetFloat(1.0f);

Input_InChart2Multiplier.Name = "Chart 2 Multiplier";
Input_InChart2Multiplier.SetFloat(2.0f);

Input_InChart3Multiplier.Name = "Chart 3 Multiplier";
Input_InChart3Multiplier.SetFloat(1.0f);

return;
}

```

//Formula: Buy1 Sell2 Buy1

```
SCGraphData Chart2BaseData;
sc.GetChartData(-Input_InChart2Number.GetChartNumber(), Chart2BaseData);
```

```
SCGraphData Chart3BaseData;
sc.GetChartData(-Input_InChart3Number.GetChartNumber(), Chart3BaseData);
```

```
// Iterate through the bars
```

```
for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
{
    int Chart2Index = sc.GetNearestMatchForDateTimeIndex(Input_InChart2Number.GetChartNumber(), Index);
    int Chart3Index = sc.GetNearestMatchForDateTimeIndex(Input_InChart3Number.GetChartNumber(), Index);
```

```
//When use latest source data for last bar is set to Yes, replay is not running and at last bar in the destination chart,
then use the data from the very latest bar in the source charts.
```

```
if(Input_UseLatestSourceDataForLastBar.GetYesNo() && !sc.IsReplayRunning() && Index == sc.ArraySize - 1)
{
    Chart2Index = Chart2BaseData[SC_OPEN].GetArraySize() - 1;
    Chart3Index = Chart3BaseData[SC_OPEN].GetArraySize() - 1;
}
```

```
// Iterate through the subgraphs
```

```
for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_NUM_TRADES; SubgraphIndex++)
{
    sc.Subgraph[SubgraphIndex][Index]
        = (sc.BaseDataIn[SubgraphIndex][Index] * Input_InChart1Multiplier.GetFloat())
        - (Chart2BaseData[SubgraphIndex][Chart2Index] * Input_InChart2Multiplier.GetFloat())
        + (Chart3BaseData[SubgraphIndex][Chart3Index] * Input_InChart3Multiplier.GetFloat())
        ;
}
```

```
sc.Subgraph[SC_HIGH][Index]
    = max(
        sc.Subgraph[SC_OPEN][Index],
        max(
            sc.Subgraph[SC_HIGH][Index],
            max(
                sc.Subgraph[SC_LOW][Index],
                sc.Subgraph[SC_LAST][Index]
            )
        )
    );
```

```
sc.Subgraph[SC_LOW][Index]
    = min(
        sc.Subgraph[SC_OPEN][Index],
        min(
            sc.Subgraph[SC_HIGH][Index],
            min(
                sc.Subgraph[SC_LOW][Index],
                sc.Subgraph[SC_LAST][Index]
            )
        )
    );
```

```
sc.CalculateOHLCAverages(Index);
```

```
}
```

```
sc.GraphName.Format("Butterfly Spread: +%s - %s + %s",
    sc.GetChartName(sc.ChartNumber).GetChars(),
```

```

        sc.GetChartName(Input_InChart2Number.GetChartNumber()).GetChars(),
        sc.GetChartName(Input_InChart3Number.GetChartNumber()).GetChars());
    }

    /*=====*/

SCSFExport scsf_Sum3Chart(SCStudyInterfaceRef sc)
{
    SCInputRef Input_InChart2Number = sc.Input[3];
    SCInputRef Input_InChart3Number = sc.Input[4];
    SCInputRef Input_InChart1Multiplier = sc.Input[5];
    SCInputRef Input_InChart2Multiplier = sc.Input[6];
    SCInputRef Input_InChart3Multiplier = sc.Input[7];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Sum - 3 Chart";

        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.GraphRegion = 1;
        sc.AutoLoop = 0;
        sc.GraphUsesChartColors = 1;
        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        Input_InChart2Number.Name = "Chart 2 Number";
        Input_InChart2Number.SetChartNumber(1);

        Input_InChart3Number.Name = "Chart 3 Number";
        Input_InChart3Number.SetChartNumber(1);

        Input_InChart1Multiplier.Name = "Chart 1 Multiplier";
        Input_InChart1Multiplier.SetFloat(1.0f);

        Input_InChart2Multiplier.Name = "Chart 2 Multiplier";
        Input_InChart2Multiplier.SetFloat(1.0f);

        Input_InChart3Multiplier.Name = "Chart 3 Multiplier";
        Input_InChart3Multiplier.SetFloat(1.0f);

        return;
    }

    if (sc.IsFullRecalculation && sc.UpdateStartIndex == 0)
    {
        for (int SubgraphIndex = 0; SubgraphIndex <= NUM_BASE_GRAPH_ARRAYS; ++SubgraphIndex)
        {
            sc.Subgraph[SubgraphIndex].Name = sc.GetStudySubgraphName(0, SubgraphIndex);
        }

        SCString Chart1Name = sc.GetChartName(sc.ChartNumber);
        SCString Chart2Name = sc.GetChartName(Input_InChart2Number.GetChartNumber());
        SCString Chart3Name = sc.GetChartName(Input_InChart3Number.GetChartNumber());
        sc.GraphName.Format("Sum - 3 Chart: %s + %s + %s", Chart1Name.GetChars(), Chart2Name.GetChars(),
Chart3Name.GetChars());
    }

    SCGraphData Chart2BaseData;
    sc.GetChartBaseData(-Input_InChart2Number.GetChartNumber(), Chart2BaseData);

    SCGraphData Chart3BaseData;
    sc.GetChartBaseData(-Input_InChart3Number.GetChartNumber(), Chart3BaseData);

```

```

// Iterate through the bars
for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
{
    int Chart2Index = sc.GetNearestMatchForDateTimeIndex(Input_InChart2Number.GetChartNumber(), Index);
    int Chart3Index = sc.GetNearestMatchForDateTimeIndex(Input_InChart3Number.GetChartNumber(), Index);

    // Iterate through the subgraphs
    for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_NUM_TRADES; SubgraphIndex++)
    {
        sc.Subgraph[SubgraphIndex][Index]
            = (sc.BaseDataIn[SubgraphIndex][Index] * Input_InChart1Multiplier.GetFloat())
            + (Chart2BaseData[SubgraphIndex][Chart2Index] * Input_InChart2Multiplier.GetFloat())
            + (Chart3BaseData[SubgraphIndex][Chart3Index] * Input_InChart3Multiplier.GetFloat())
            ;
    }

    sc.Subgraph[SC_HIGH][Index]
        = max(
            sc.Subgraph[SC_OPEN][Index],
            max(
                sc.Subgraph[SC_HIGH][Index],
                max(
                    sc.Subgraph[SC_LOW][Index],
                    sc.Subgraph[SC_LAST][Index]
                )
            )
        );

    sc.Subgraph[SC_LOW][Index]
        = min(
            sc.Subgraph[SC_OPEN][Index],
            min(
                sc.Subgraph[SC_HIGH][Index],
                min(
                    sc.Subgraph[SC_LOW][Index],
                    sc.Subgraph[SC_LAST][Index]
                )
            )
        );

    sc.CalculateOHLCAverages(Index);
}

}
/*=====*/
SCSFExport scsf_HighestHighLowestLowOverNBars(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_HighestHigh = sc.Subgraph[0];
    SCSubgraphRef Subgraph_LowestLow = sc.Subgraph[1];

    SCInputRef Input_Length = sc.Input[3];
    SCInputRef Input_DataHigh = sc.Input[5];
    SCInputRef Input_DataLow = sc.Input[6];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Highest High / Lowest Low Over N Bars";

        sc.GraphRegion = 0;
        sc.AutoLoop = 0;

        Subgraph_HighestHigh.Name= "Highest High";
        Subgraph_HighestHigh.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_HighestHigh.PrimaryColor = RGB(0,255,0);
        Subgraph_HighestHigh.DrawZeros = true;
    }
}

```

```

Subgraph_LowestLow.Name = "Lowest Low";
Subgraph_LowestLow.DrawStyle = DRAWSTYLE_LINE;
Subgraph_LowestLow.PrimaryColor = RGB(255,0,255);
Subgraph_LowestLow.DrawZeros = true;

Input_Length.Name = "Length";
Input_Length.SetInt(5);
Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_DataHigh.Name = "Input Data High";
Input_DataHigh.SetInputDataIndex(SC_HIGH);

Input_DataLow.Name = "Input Data Low";
Input_DataLow.SetInputDataIndex(SC_LOW);

return;
}

sc.DataStartIndex = Input_Length.GetInt() - 1;

int & HighestIndex = sc.GetPersistentInt(1);
int & LowestIndex = sc.GetPersistentInt(2);
float & CurrentHighest = sc.GetPersistentFloat(1);
float & CurrentLowest = sc.GetPersistentFloat(2);
if (sc.UpdateStartIndex == 0)
{
    HighestIndex = -1;
    LowestIndex = -1;
    CurrentHighest = -FLT_MAX;
    CurrentLowest = FLT_MAX;
}

int HighDataIndex = Input_DataHigh.GetInputDataIndex();
int LowDataIndex = Input_DataLow.GetInputDataIndex();
for (int BarIndex = sc.UpdateStartIndex; BarIndex < sc.ArraySize; BarIndex++)
{
    int EarliestIndex = BarIndex - (Input_Length.GetInt() - 1);
    bool NeedToDetermineHigh =
        (HighestIndex < EarliestIndex || sc.BaseDataIn[HighDataIndex][BarIndex] > CurrentHighest);

    bool NeedToDetermineLow =
        (LowestIndex < EarliestIndex || sc.BaseDataIn[LowDataIndex][BarIndex] < CurrentLowest);

    if (NeedToDetermineHigh || NeedToDetermineLow)
    {
        if (NeedToDetermineHigh)
            CurrentHighest = -FLT_MAX;

        if (NeedToDetermineLow)
            CurrentLowest = FLT_MAX;

        for (int PriorBarIndex = EarliestIndex; PriorBarIndex <= BarIndex; PriorBarIndex++)
        {
            if (NeedToDetermineHigh)
            {
                if (CurrentHighest < sc.BaseDataIn[HighDataIndex][PriorBarIndex])
                {
                    CurrentHighest = sc.BaseDataIn[HighDataIndex][PriorBarIndex];
                    HighestIndex = PriorBarIndex;
                }
            }
        }

        if (NeedToDetermineLow)

```

```

        {
            if (CurrentLowest > sc.BaseDataIn[LowDataIndex][PriorBarIndex])
            {
                CurrentLowest = sc.BaseDataIn[LowDataIndex][PriorBarIndex];
                LowestIndex = PriorBarIndex;
            }
        }
    }

    Subgraph_HighestHigh[BarIndex] = CurrentHighest;
    Subgraph_LowestLow[BarIndex] = CurrentLowest;
}

}

/*=====*/
SCSFExport scsf_DonchianChannel(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_HighestHigh = sc.Subgraph[0];
    SCSubgraphRef Subgraph_LowestLow = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Midline = sc.Subgraph[2];
    SCInputRef Input_Length = sc.Input[3];
    SCInputRef Input_UseClose = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Donchian Channel";

        sc.GraphRegion = 0;
        sc.AutoLoop = true;

        Subgraph_HighestHigh.Name= "Highest High";
        Subgraph_HighestHigh.DrawStyle= DRAWSTYLE_LINE;
        Subgraph_HighestHigh.DrawZeros= true;
        Subgraph_HighestHigh.GraphicalDisplacement= 1;

        Subgraph_LowestLow.Name = "Lowest Low";
        Subgraph_LowestLow.DrawStyle= DRAWSTYLE_LINE;
        Subgraph_LowestLow.DrawZeros= true;
        Subgraph_LowestLow.GraphicalDisplacement= 1;

        Subgraph_Midline.Name = "Mid Line";
        Subgraph_Midline.DrawStyle= DRAWSTYLE_LINE;
        Subgraph_Midline.LineStyle= LINESTYLE_DOT;
        Subgraph_Midline.DrawZeros= true;
        Subgraph_Midline.GraphicalDisplacement= 1;

        Input_Length.Name = "Length ";
        Input_Length.SetInt(5);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_UseClose.Name = "Use Close instead of High and Low";
        Input_UseClose.SetYesNo(false);

        return;
    }

    float Lowest = FLT_MAX;
    float Highest = -FLT_MAX;

    if (Input_UseClose.GetYesNo())
    {
        for (int BarIndex = sc.Index - (Input_Length.GetInt() - 1); BarIndex <= sc.Index; BarIndex++)
        {
            Highest = (Highest < sc.Close[BarIndex]) ? sc.Close[BarIndex] : Highest;

```

```

        Lowest = (Lowest > sc.Close[BarIndex]) ? sc.Close[BarIndex] : Lowest;
    }
}
else
{
    for (int BarIndex = sc.Index - (Input_Length.GetInt() - 1); BarIndex <= sc.Index; BarIndex++)
    {
        if (Highest < sc.High[BarIndex])
            Highest = sc.High[BarIndex];

        if (Lowest > sc.Low[BarIndex])
            Lowest = sc.Low[BarIndex];
    }
}

Subgraph_HighestHigh[sc.Index] = Highest;
Subgraph_LowestLow[sc.Index] = Lowest;
Subgraph_Midline[sc.Index] = (Highest + Lowest)/2.0f;
}

/*=====*/
SCSFExport scsf_McClellanOscillator(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_MO = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Temp2 = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Temp3 = sc.Subgraph[3];

    SCInputRef Input_DecliningIssuesChartNumber = sc.Input[3];
    SCInputRef Input_UseAbsValue = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "McClellan Oscillator" ;
        sc.GraphRegion = 1;
        sc.ValueFormat = 3;

        Subgraph_MO.Name = "MO";
        Subgraph_MO.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_MO.PrimaryColor = RGB(0,255,0);
        Subgraph_MO.DrawZeros = true;

        Input_DecliningIssuesChartNumber.Name = "Declining Issues Chart #";
        Input_DecliningIssuesChartNumber.SetChartNumber(1);

        Input_UseAbsValue.Name = "Use ABS Value";
        Input_UseAbsValue.SetYesNo(0);

        return;
    }

    SCGraphData ChartBaseData;
    sc.GetChartBaseData(-Input_DecliningIssuesChartNumber.GetChartNumber(), ChartBaseData);

    SCFloatArrayRef Chart2Array = ChartBaseData[SC_LAST];

    if (Chart2Array.GetArraySize() == 0)
        return;

    int i1 = 0;
    int i = 0;
    sc.DataStartIndex = 1;

    Subgraph_Temp3[i] = Subgraph_Temp2[i] = sc.Close[i] - Chart2Array[i1];

```



```

for (i = max(1, sc.UpdateStartIndex); i < sc.ArraySize; i++)
{
    int i1 = sc.GetNearestMatchForDateTimeIndex(Input_DecliningIssuesChartNumber.GetChartNumber(), i);

    Subgraph_Temp3[i] = Subgraph_Temp3[i - 1] * 0.9f +
        ((sc.Close[i] - Chart2Array[i1]) * 0.1f);

    if (Input_UseAbsValue.GetYesNo())
    {
        Subgraph_Temp2[i] = Subgraph_Temp2[i - 1] * 0.95f +
            (fabs(
                (sc.Close[i] - Chart2Array[i1]) * 0.05f));
    }
    else
    {
        Subgraph_Temp2[i] = Subgraph_Temp2[i - 1] * 0.95f +
            ((
                (sc.Close[i] - Chart2Array[i1]) * 0.05f));
    }

    Subgraph_MO[i] = Subgraph_Temp3[i] - Subgraph_Temp2[i];
}
}

/*=====*/
SCSFExport scsf_BarDifference(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Difference = sc.Subgraph[0];
    SCInputRef Input_Data1 = sc.Input[0];
    SCInputRef Input_Data2 = sc.Input[1];
    SCInputRef Input_Data1Offset = sc.Input[2];
    SCInputRef Input_Data2Offset = sc.Input[3];
    SCInputRef Input_CalculateDifferenceInPriceTicks = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Bar Difference";

        sc.GraphRegion = 1;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_Difference.Name = "Difference";
        Subgraph_Difference.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Difference.PrimaryColor = RGB(0,255,0);
        Subgraph_Difference.DrawZeros = true;

        Input_Data1.Name = "Input Data 1";
        Input_Data1.SetInputDataIndex(SC_HIGH);

        Input_Data2.Name = "Input Data 2";
        Input_Data2.SetInputDataIndex(SC_LOW);

        Input_Data1Offset.Name = "Input Data 1 Offset";
        Input_Data1Offset.SetInt(0);
        Input_Data1Offset.SetIntLimits(0, MAX_STUDY_LENGTH);

        Input_Data2Offset.Name = "Input Data 2 Offset";
        Input_Data2Offset.SetInt(0);
        Input_Data2Offset.SetIntLimits(0, MAX_STUDY_LENGTH);

        Input_CalculateDifferenceInPriceTicks.Name = "Calculate Difference in Price Ticks";
        Input_CalculateDifferenceInPriceTicks.SetYesNo(false);
    }
}

```

```

    return;
}

if (sc.Index < Input_Data2Offset.GetInt()
    || sc.Index < Input_Data1Offset.GetInt())
{
    return;
}

Subgraph_Difference[sc.Index] =
    sc.BaseDataIn[Input_Data1.GetInputDataIndex()][sc.Index - Input_Data1Offset.GetInt()] -
    sc.BaseDataIn[Input_Data2.GetInputDataIndex()][sc.Index - Input_Data2Offset.GetInt()];

if (Input_CalculateDifferenceInPriceTicks.GetYesNo() && sc.TickSize != 0)
    Subgraph_Difference[sc.Index] /= sc.TickSize;
}

/*=====*/
SCSFExport scsf_McClellanSummationIndex(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Sum = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Temp2 = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Temp3 = sc.Subgraph[3];

    SCInputRef DecliningIssuesChartNumber = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "McClellan Summation Index";

        sc.GraphRegion = 1;
        sc.ValueFormat = 0;

        Subgraph_Sum.Name = "Sum";
        Subgraph_Sum.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Sum.PrimaryColor = RGB(0,255,0);
        Subgraph_Sum.DrawZeros = true;

        DecliningIssuesChartNumber.Name = "Declining Issues Chart #";
        DecliningIssuesChartNumber.SetChartNumber(1);

        return;
    }

    SCGraphData ChartBaseData;
    sc.GetChartBaseData(-DecliningIssuesChartNumber.GetChartNumber(), ChartBaseData);

    SCFloatArrayRef Chart2Array = ChartBaseData[SC_LAST];
    if (Chart2Array.GetArraySize() == 0)
        return;

    sc.DataStartIndex = 1;

    Subgraph_Temp3[0] = Subgraph_Temp2[0] = sc.Close[0] - Chart2Array[0];

    for (int i = max(1, sc.UpdateStartIndex); i < sc.ArraySize; i++)
    {
        int i1 = sc.GetNearestMatchForDateTimeIndex(DecliningIssuesChartNumber.GetChartNumber(), i);

        if (sc.Close[i] == 0 || Chart2Array[i1] == 0)
        {
            Subgraph_Sum[i] = Subgraph_Sum[i - 1];
            continue;
        }
    }

```

```

Subgraph_Temp3[i] = Subgraph_Temp3[i - 1] * 0.9f +
((sc.Close[i] - Chart2Array[i1]) * 0.1f);

Subgraph_Temp2[i] = Subgraph_Temp2[i - 1] * 0.95f +
((
(sc.Close[i] - Chart2Array[i1]) * 0.05f));

Subgraph_Sum[i] = Subgraph_Sum[i - 1] + (Subgraph_Temp3[i] - Subgraph_Temp2[i]);
}
}

/*=====*/
SCSFExport scsf_MomentumTrend(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_MomentumUp = sc.Subgraph[0];
    SCSubgraphRef Subgraph_MomentumDown = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Momentum = sc.Subgraph[2];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];
    SCInputRef Input_MomentumType = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Momentum Trend";

        sc.GraphRegion = 0;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_MomentumUp.Name = "Momentum Up";
        Subgraph_MomentumUp.DrawStyle = DRAWSTYLE_POINT;
        Subgraph_MomentumUp.PrimaryColor = RGB(0,255,0);
        Subgraph_MomentumUp.LineWidth = 5;
        Subgraph_MomentumUp.DrawZeros = false;

        Subgraph_MomentumDown.Name = "Momentum Down";
        Subgraph_MomentumDown.DrawStyle = DRAWSTYLE_POINT;
        Subgraph_MomentumDown.PrimaryColor = RGB(255,0,255);
        Subgraph_MomentumDown.LineWidth = 5;
        Subgraph_MomentumDown.DrawZeros = false;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_Length.Name = "Length";
        Input_Length.SetInt(10);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_MomentumType.Name = "Momentum Type";
        Input_MomentumType.SetCustomInputStrings("Difference;Quotient");
        Input_MomentumType.SetCustomInputIndex(0);

        return;
    }

    sc.DataStartIndex = Input_Length.GetInt() + 1;

    if (sc.Index < sc.DataStartIndex - 1)
        return;

    const int SelectedIndex = Input_MomentumType.GetIndex();
    switch (SelectedIndex)
    {

```

```

    case 0:
    {
        Subgraph_Momentum[sc.Index] = sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index] -
sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_Length.GetInt()];
    }
    break;

    case 1:
    {
        if (sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_Length.GetInt()] != 0.0f)
            Subgraph_Momentum[sc.Index] = (sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index] /
sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_Length.GetInt()]) * 100;
        else
            Subgraph_Momentum[sc.Index] = 0.0f;
    }
    break;
}

if (sc.Index < sc.DataStartIndex)
    return;

if (Subgraph_Momentum[sc.Index] > Subgraph_Momentum[sc.Index - 1])
{
    Subgraph_MomentumUp[sc.Index] = sc.High[sc.Index];
    Subgraph_MomentumDown[sc.Index] = 0;
}
else
{
    Subgraph_MomentumDown[sc.Index] = sc.Low[sc.Index];
    Subgraph_MomentumUp[sc.Index] = 0;
}
}

/*=====*/
SCSFExport scsf_SumAllCharts(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
    SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
    SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
    SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Sum All Charts";

        sc.ValueFormat      = 2;
        sc.GraphRegion      = 1;
        sc.UseGlobalChartColors = 0;
        sc.GraphDrawType    = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        // We are using Manual looping.
        sc.AutoLoop = false;

        Subgraph_Open.Name = "Open";
        Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Open.PrimaryColor = RGB(0,255,0);
    }
}

```

```

Subgraph_Open.DrawZeros = false;

Subgraph_High.Name = "High";
Subgraph_High.DrawStyle = DRAWSTYLE_LINE;
Subgraph_High.PrimaryColor = RGB(0,255,0);
Subgraph_High.DrawZeros = false;

Subgraph_Low.Name = "Low";
Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Low.PrimaryColor = RGB(0,255,0);
Subgraph_Low.DrawZeros = false;

Subgraph_Last.Name = "Last";
Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Last.PrimaryColor = RGB(0,255,0);
Subgraph_Last.DrawZeros = false;

Subgraph_Volume.Name = "Volume";
Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Volume.PrimaryColor = RGB(255,0,0);
Subgraph_Volume.DrawZeros = false;

Subgraph_OpenInterest.Name = "# of Trades / OI";
Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OpenInterest.PrimaryColor = RGB(0,0,255);
Subgraph_OpenInterest.DrawZeros = false;

Subgraph_OHLCAvg.Name = "OHLC Avg";
Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OHLCAvg.PrimaryColor = RGB(0,255,0);
Subgraph_OHLCAvg.DrawZeros = false;

Subgraph_HLCAvg.Name = "HLC Avg";
Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);
Subgraph_HLCAvg.DrawZeros = false;

Subgraph_HLAvg.Name = "HL Avg";
Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLAvg.PrimaryColor = RGB(0,127,255);
Subgraph_HLAvg.DrawZeros = false;

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);
Subgraph_BidVol.DrawZeros = false;

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);
Subgraph_AskVol.DrawZeros = false;

return;
}

// Initialize elements to 0
for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
{
    for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_NUM_TRADES; SubgraphIndex++)
    {
        sc.Subgraph[SubgraphIndex][Index] = 0;
    }
}

// Sum charts

```

```

const int MaxCharts = 200;
for (int ChartIndex = 1; ChartIndex <= MaxCharts; ChartIndex++)
{
    SCGraphData RefChartBaseData;
    sc.GetChartData(-ChartIndex, RefChartBaseData);

    if (RefChartBaseData[SC_OPEN].GetArraySize() == 0)
        continue;

    for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
    {
        int RefChartIndex = sc.GetNearestMatchForDateTimeIndex(ChartIndex, Index);

        for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_NUM_TRADES; SubgraphIndex++)
        {
            sc.Subgraph[SubgraphIndex][Index] += RefChartBaseData[SubgraphIndex][RefChartIndex];
        }

        sc.Subgraph[SC_HIGH][Index]
            = max(sc.Subgraph[SC_OPEN][Index],
                max(sc.Subgraph[SC_HIGH][Index],
                    max(sc.Subgraph[SC_LOW][Index], sc.Subgraph[SC_LAST][Index])
                )
            );

        sc.Subgraph[SC_LOW][Index]
            = min(sc.Subgraph[SC_OPEN][Index],
                min(sc.Subgraph[SC_HIGH][Index],
                    min(sc.Subgraph[SC_LOW][Index], sc.Subgraph[SC_LAST][Index])
                )
            );
    }
}

// Calculate averages
for (int Index = sc.UpdateStartIndex; Index < sc.ArraySize; Index++)
    sc.CalculateOHLCAverages(Index);
}

/*=====*/
SCSFExport scsf_Kurtosis(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_SecondCentralMoment = sc.Subgraph[0];
    SCSubgraphRef Subgraph_FourthCentralMoment = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Kurtosis = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Line = sc.Subgraph[3];

    SCFloatArrayRef Array_Avg = Subgraph_Kurtosis.Arrays[0];

    SCInputRef Input_InputData = sc.Input[0];
    SCInputRef Input_Length = sc.Input[1];
    SCInputRef Input_KurtosisType = sc.Input[2];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Kurtosis";

        sc.AutoLoop = 1;

        Subgraph_SecondCentralMoment.Name = "Second Central Moment";
        Subgraph_SecondCentralMoment.DrawStyle = DRAWSTYLE_IGNORE;

        Subgraph_FourthCentralMoment.Name = "Fourth Central Moment";
        Subgraph_FourthCentralMoment.DrawStyle = DRAWSTYLE_IGNORE;
    }
}

```

```

Subgraph_Kurtosis.Name = "Kurtosis";
Subgraph_Kurtosis.DrawStyle = DRAWSTYLE_BAR;
Subgraph_Kurtosis.PrimaryColor = RGB(0, 255, 0);
Subgraph_Kurtosis.SecondaryColor = RGB(255, 0, 0);
Subgraph_Kurtosis.SecondaryColorUsed = true;
Subgraph_Kurtosis.DrawZeros = false;

Subgraph_Line.Name = "Line";
Subgraph_Line.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Line.PrimaryColor = RGB(128, 0, 128);
Subgraph_Line.DrawZeros = true;

Input_InputData.Name = "Input Data";
Input_InputData.SetInputDataIndex(SC_LAST);

Input_Length.Name = "Length";
Input_Length.SetInt(10);
Input_Length.SetIntLimits(4, INT_MAX);

Input_KurtosisType.Name = "Kurtosis Type";
Input_KurtosisType.SetCustomInputStrings("Population;" "Sample");
Input_KurtosisType.SetCustomInputIndex(0);

return;
}

int Length = Input_Length.GetInt();

sc.DataStartIndex = Length - 1;

sc.SimpleMovAvg(sc.BaseDataIn[Input_InputData.GetInputDataIndex()], Array_Avg, Length);

// Compute second and fourth central moments.
float Deviation = 0.0f;
float DeviationSquared = 0.0f;
float SumDeviationSquared = 0.0f;
float DeviationFourthPower = 0.0f;
float SumDeviationFourthPower = 0.0f;

for (int i = sc.Index - Length + 1; i <= sc.Index; i++)
{
    Deviation = sc.BaseDataIn[Input_InputData.GetInputDataIndex()][i] - Array_Avg[i];
    DeviationSquared = Deviation * Deviation;
    SumDeviationSquared += DeviationSquared;
    DeviationFourthPower = DeviationSquared * DeviationSquared;
    SumDeviationFourthPower += DeviationFourthPower;
}

Subgraph_SecondCentralMoment[sc.Index] = SumDeviationSquared / Length;
Subgraph_FourthCentralMoment[sc.Index] = SumDeviationFourthPower / Length;

// Two cases for Kurtosis Type:
const int SelectedIndex = Input_KurtosisType.GetIndex();

switch (SelectedIndex)
{
{
case 0:
{
    Subgraph_Kurtosis[sc.Index] = Subgraph_FourthCentralMoment[sc.Index] /
(Subgraph_SecondCentralMoment[sc.Index] * Subgraph_SecondCentralMoment[sc.Index]) - 3.0f;

    if (Subgraph_Kurtosis[sc.Index] > 0)
        Subgraph_Kurtosis.DataColor[sc.Index] = Subgraph_Kurtosis.PrimaryColor;
    else
        Subgraph_Kurtosis.DataColor[sc.Index] = Subgraph_Kurtosis.SecondaryColor;

```

```

    }
    break;

    case 1:
    {
        Subgraph_Kurtosis[sc.Index] = ((Length - 1.0f)*(Length + 1.0f)) / ((Length - 2.0f)*(Length -
3.0f))*Subgraph_FourthCentralMoment[sc.Index] / (Subgraph_SecondCentralMoment[sc.Index] *
Subgraph_SecondCentralMoment[sc.Index]) - 3.0f*(Length - 1)*(Length - 1) / ((Length - 2)*(Length - 3));

        if (Subgraph_Kurtosis[sc.Index] > 0)
            Subgraph_Kurtosis.DataColor[sc.Index] = Subgraph_Kurtosis.PrimaryColor;
        else
            Subgraph_Kurtosis.DataColor[sc.Index] = Subgraph_Kurtosis.SecondaryColor;
    }
    break;
}

Subgraph_Line[sc.Index] = 0.0f;

}

/*=====*/
SCSFExport scsf_AC_DC_Histogram(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_ACDC = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Temp3 = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Temp4 = sc.Subgraph[4];
    SCSubgraphRef Subgraph_Temp5 = sc.Subgraph[5];
    SCSubgraphRef Subgraph_Temp6 = sc.Subgraph[6];
    SCSubgraphRef Subgraph_Temp7 = sc.Subgraph[7];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_MA1Length = sc.Input[2];
    SCInputRef Input_MA2Length = sc.Input[3];
    SCInputRef Input_MA3Length = sc.Input[4];
    SCInputRef Input_MA4Length = sc.Input[5];

    if (sc.SetDefaults)
    {
        sc.GraphName = "AC/DC Histogram";

        sc.GraphRegion = 1;
        sc.ValueFormat = 3;
        sc.AutoLoop = 1;

        Subgraph_ACDC.Name = "AC/DC";
        Subgraph_ACDC.DrawStyle = DRAWSTYLE_BAR;
        Subgraph_ACDC.SecondaryColor = RGB(255,0,0);
        Subgraph_ACDC.AutoColoring = AUTOCOLOR_SLOPE;
        Subgraph_ACDC.DrawZeros = false;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_MA1Length.Name = "Moving Average 1 Length";
        Input_MA1Length.SetInt(34);
        Input_MA1Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_MA2Length.Name = "Moving Average 2 Length";
        Input_MA2Length.SetInt(5);
        Input_MA2Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        Input_MA3Length.Name = "Moving Average 3 Length";
        Input_MA3Length.SetInt(34);

```



```

    Input_MA3Length.SetIntLimits(1, MAX_STUDY_LENGTH);

    Input_MA4Length.Name = "Moving Average 4 Length";
    Input_MA4Length.SetInt(5);
    Input_MA4Length.SetIntLimits(1, MAX_STUDY_LENGTH);

    return;
}

sc.DataStartIndex = max(Input_MA1Length.GetInt(), Input_MA2Length.GetInt()) + Input_MA3Length.GetInt() +
Input_MA4Length.GetInt();

sc.SimpleMovAvg(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_Temp3, Input_MA1Length.GetInt());
sc.SimpleMovAvg(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_Temp4, Input_MA2Length.GetInt());

if (sc.Index > 0)
    Subgraph_Temp5[sc.Index] = Subgraph_Temp4[sc.Index] - Subgraph_Temp3[sc.Index];

sc.SimpleMovAvg(Subgraph_Temp5, Subgraph_Temp6, Input_MA4Length.GetInt());
Subgraph_Temp7[sc.Index] = Subgraph_Temp5[sc.Index] - Subgraph_Temp6[sc.Index];
sc.SimpleMovAvg(Subgraph_Temp7, Subgraph_ACDC, Input_MA3Length.GetInt());
}

/*=====*/
SCSFExport scsf_MovingAverageTriangular(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_MA = sc.Subgraph[0];
    SCSubgraphRef Subgraph_TempMA = sc.Subgraph[1];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Length = sc.Input[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average - Triangular";

        sc.GraphRegion = 0;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_MA.Name = "MA";
        Subgraph_MA.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_MA.PrimaryColor = RGB(0,255,0);
        Subgraph_MA.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_Length.Name = "Length";
        Input_Length.SetInt(9);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.DataStartIndex = Input_Length.GetInt() ;

    sc.TriangularMovingAverage(sc.BaseDataIn[Input_Data.GetInputDataIndex()], Subgraph_MA, Input_Length.GetInt());
}

/*=====*/
SCSFExport scsf_1DividedByPrice(SCStudyInterfaceRef sc)
{

```

```

SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
SCSubgraphRef Subgraph_High = sc.Subgraph[1];
SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];

if (sc.SetDefaults)
{
    sc.GraphName = "1 Divided by Price";

    sc.GraphUsesChartColors= true;

    sc.ValueFormat = VALUEFORMAT_INHERITED;
    sc.GraphRegion = 0;
    sc.DisplayAsMainPriceGraph= true;
    sc.AutoLoop = 1;

    sc.GraphDrawType = GDT_OHLCBAR;
    sc.StandardChartHeader = 1;

    Subgraph_Open.Name = "Open";
    Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_Open.PrimaryColor = RGB(0,255,0);
    Subgraph_Open.DrawZeros = false;

    Subgraph_High.Name = "High";
    Subgraph_High.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_High.PrimaryColor = RGB(0,255,0);
    Subgraph_High.DrawZeros = false;

    Subgraph_Low.Name = "Low";
    Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_Low.PrimaryColor = RGB(0,255,0);
    Subgraph_Low.DrawZeros = false;

    Subgraph_Last.Name = "Last";
    Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;
    Subgraph_Last.PrimaryColor = RGB(0,255,0);
    Subgraph_Last.DrawZeros = false;

    Subgraph_Volume.Name = "Volume";
    Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_Volume.PrimaryColor = RGB(255,0,0);
    Subgraph_Volume.DrawZeros = false;

    Subgraph_OpenInterest.Name = "Num Trades / OpenInterest";
    Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_OpenInterest.PrimaryColor = RGB(0,0,255);
    Subgraph_OpenInterest.DrawZeros = false;

    Subgraph_OHLCAvg.Name = "OHLC Avg";
    Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_OHLCAvg.PrimaryColor = RGB(0,255,0);
    Subgraph_OHLCAvg.DrawZeros = false;

    Subgraph_HLCAvg.Name = "HLC Avg";
    Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
    Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);
    Subgraph_HLCAvg.DrawZeros = false;

```

```

Subgraph_HLAvg.Name = "HL Avg";
Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLAvg.PrimaryColor = RGB(0,127,255);
Subgraph_HLAvg.DrawZeros = false;

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);
Subgraph_BidVol.DrawZeros = false;

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);
Subgraph_AskVol.DrawZeros = false;

return;
}

if (sc.BaseDataIn[SC_OPEN][sc.Index] == 0.0f)
    sc.Subgraph[SC_OPEN][sc.Index] = 0.0f;
else
    sc.Subgraph[SC_OPEN][sc.Index] = 1.0f / sc.BaseDataIn[SC_OPEN][sc.Index];

if (sc.BaseDataIn[SC_HIGH][sc.Index] == 0.0f)
    sc.Subgraph[SC_LOW][sc.Index] = 0.0f;
else
    sc.Subgraph[SC_LOW][sc.Index] = 1.0f / sc.BaseDataIn[SC_HIGH][sc.Index];

if (sc.BaseDataIn[SC_LOW][sc.Index] == 0.0f)
    sc.Subgraph[SC_HIGH][sc.Index] = 0.0f;
else
    sc.Subgraph[SC_HIGH][sc.Index] = 1.0f / sc.BaseDataIn[SC_LOW][sc.Index];

if (sc.BaseDataIn[SC_LAST][sc.Index] == 0.0f)
    sc.Subgraph[SC_LAST][sc.Index] = 0.0f;
else
    sc.Subgraph[SC_LAST][sc.Index] = 1.0f / sc.BaseDataIn[SC_LAST][sc.Index];

sc.Subgraph[SC_VOLUME][sc.Index] = sc.BaseDataIn[SC_VOLUME][sc.Index];
sc.Subgraph[SC_NUM_TRADES][sc.Index] = sc.BaseDataIn[SC_NUM_TRADES][sc.Index];

if (sc.BaseDataIn[SC_OHLC_AVG][sc.Index] == 0.0f)
    sc.Subgraph[SC_OHLC_AVG][sc.Index] = 0.0f;
else
    sc.Subgraph[SC_OHLC_AVG][sc.Index] = 1.0f / sc.BaseDataIn[SC_OHLC_AVG][sc.Index];

if (sc.BaseDataIn[SC_HLC_AVG][sc.Index] == 0.0f)
    sc.Subgraph[SC_HLC_AVG][sc.Index] = 0.0f;
else
    sc.Subgraph[SC_HLC_AVG][sc.Index] = 1.0f / sc.BaseDataIn[SC_HLC_AVG][sc.Index];

if (sc.BaseDataIn[SC_HL_AVG][sc.Index] == 0.0f)
    sc.Subgraph[SC_HL_AVG][sc.Index] = 0.0f;
else
    sc.Subgraph[SC_HL_AVG][sc.Index] = 1.0f / sc.BaseDataIn[SC_HL_AVG][sc.Index];

sc.Subgraph[SC_ASKVOL][sc.Index] = sc.BaseDataIn[SC_ASKVOL][sc.Index];
sc.Subgraph[SC_BIDVOL][sc.Index] = sc.BaseDataIn[SC_BIDVOL][sc.Index];

```

```

SCString ChartName = sc.GetStudyNameFromChart(sc.ChartNumber, 0);
sc.GraphName.Format("1/P %s", ChartName.GetChars());
}

```

```

/*=====*/

```

```

SCSFExport scsf_RoundPriceBarsToTickSize(SCStudyInterfaceRef sc)
{

```

```

    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
    SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
    SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
    SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];

```

```

    if (sc.SetDefaults)
    {
        sc.GraphName = "Round Price Bars To Tick Size";

        sc.StudyDescription= "This study when applied to a chart will round the bar values to the Tick Size set in Chart >> Chart Settings.";

```

```

        sc.GraphUsesChartColors= true;

```

```

        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.GraphRegion = 0;
        sc.DisplayAsMainPriceGraph= true;
        sc.AutoLoop = 1;

```

```

        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

```

```

        Subgraph_Open.Name = "Open";
        Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Open.PrimaryColor = RGB(255, 255, 255);

```

```

        Subgraph_High.Name = "High";
        Subgraph_High.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_High.PrimaryColor = RGB(255, 255, 255);

```

```

        Subgraph_Low.Name = "Low";
        Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Low.PrimaryColor = RGB(255, 255, 255);

```

```

        Subgraph_Last.Name = "Last";
        Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Last.PrimaryColor = RGB(255, 255, 255);

```

```

        Subgraph_Volume.Name = "Volume";
        Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
        Subgraph_Volume.PrimaryColor = RGB(255, 0, 0);

```

```

        Subgraph_OpenInterest.Name = "Num Trades / OpenInterest";
        Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
        Subgraph_OpenInterest.PrimaryColor = RGB(0, 0, 255);

```

```

        Subgraph_OHLCAvg.Name = "OHLC Avg";
        Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
        Subgraph_OHLCAvg.PrimaryColor = RGB(0, 255, 0);

```

```

Subgraph_HLCAvg.Name = "HLC Avg";
Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);

Subgraph_HLAvG.Name = "HL Avg";
Subgraph_HLAvG.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLAvG.PrimaryColor = RGB(0,127,255);

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);

return;
}

sc.Subgraph[SC_OPEN][sc.Index] = sc.RoundToTickSize(sc.BaseDataIn[SC_OPEN][sc.Index] , sc.TickSize);
sc.Subgraph[SC_HIGH][sc.Index] = sc.RoundToTickSize(sc.BaseDataIn[SC_HIGH][sc.Index] , sc.TickSize);
sc.Subgraph[SC_LOW][sc.Index] = sc.RoundToTickSize(sc.BaseDataIn[SC_LOW][sc.Index] , sc.TickSize);
sc.Subgraph[SC_LAST][sc.Index] = sc.RoundToTickSize(sc.BaseDataIn[SC_LAST][sc.Index] , sc.TickSize);

for(int SubgraphIndex =SC_VOLUME; SubgraphIndex <=SC_ASKVOL;SubgraphIndex++)
    sc.Subgraph[SubgraphIndex][sc.Index] =sc.BaseDataIn[SubgraphIndex][sc.Index];

sc.GraphName= sc.GetStudyNameFromChart(sc.ChartNumber, 0);
}

/*=====*/

SCSFExport scsf_DifferenceSingleLine(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Difference = sc.Subgraph[0];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_UseLatestSourceDataForLastBar = sc.Input[1];
    SCInputRef Input_Chart2Number = sc.Input[3];
    SCInputRef Input_Chart1Multiplier = sc.Input[4];
    SCInputRef Input_Chart2Multiplier = sc.Input[5];
    SCInputRef Input_Chart1Addition = sc.Input[6];
    SCInputRef Input_ZeroOutputWhenOneSourceGraphHasZeroValues = sc.Input[7];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Difference (Single Line)";

        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.GraphRegion = 1;

        Subgraph_Difference.Name = "Difference";
        Subgraph_Difference.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Difference.PrimaryColor = RGB(0,255,0);
        Subgraph_Difference.DrawZeros = false;

        Input_Data.Name = "Input Data";

```

```

Input_Data.SetInputDataIndex(SC_LAST);

Input_UseLatestSourceDataForLastBar.Name = "Use Latest Source Data For Last Bar";
Input_UseLatestSourceDataForLastBar.SetYesNo(0);

Input_Chart2Number.Name = "Chart 2 Number";
Input_Chart2Number.SetChartNumber(1);

Input_Chart1Multiplier.Name = "Chart 1 Multiplier";
Input_Chart1Multiplier.SetFloat(1.0f);

Input_Chart2Multiplier.Name = "Chart 2 Multiplier";
Input_Chart2Multiplier.SetFloat(1.0f);

Input_Chart1Addition.Name = "Chart 1 Addition";
Input_Chart1Addition.SetFloat(0.0f);

Input_ZeroOutputWhenOneSourceGraphHasZeroValues.Name = "Zero Output When one Source Graph has Zero Values";
Input_ZeroOutputWhenOneSourceGraphHasZeroValues.SetYesNo(0);

return;
}

if (sc.IsFullRecalculation)
{
    SCString Chart1Name = sc.GetStudyNameFromChart(sc.ChartNumber, 0);
    SCString Chart2Name = sc.GetStudyNameFromChart(Input_Chart2Number.GetChartNumber(), 0);
    sc.GraphName.Format("Diff %s - %s", Chart1Name.GetChars(), Chart2Name.GetChars());
}

SCFloatArray Chart2Array;
sc.GetCharArray(Input_Chart2Number.GetChartNumber()*-1, Input_Data.GetInputDataIndex(), Chart2Array);

if (Chart2Array.GetArraySize() == 0)
    return;

sc.DataStartIndex = 0;

int CalculationStartIndex = sc.GetCalculationStartIndexForStudy();

for (int DestinationIndex = CalculationStartIndex; DestinationIndex < sc.ArraySize; DestinationIndex++)
{
    int Chart2Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart2Number.GetChartNumber(),
DestinationIndex);

    //When use latest source data for last bar is set to Yes, replay is not running and at last bar in the destination chart,
    then use the data from the very latest bar in the source chart.
    if (Input_UseLatestSourceDataForLastBar.GetYesNo() && !sc.IsReplayRunning() && DestinationIndex ==
sc.ArraySize - 1)
    {
        Chart2Index = Chart2Array.GetArraySize() - 1;
    }

    if (Input_ZeroOutputWhenOneSourceGraphHasZeroValues.GetYesNo())
    {
        bool AllZeros = true;

        if (Chart2Array[Chart2Index] != 0)
        {
            AllZeros = false;
        }

        if (AllZeros)

```

```

        continue;

    AllZeros = true;

    if (sc.BaseDataIn[Input_Data.GetInputDataIndex()][DestinationIndex] != 0)
    {
        AllZeros = false;
    }

    if (AllZeros)
        continue;
}

    Subgraph_Difference[DestinationIndex] = (sc.BaseDataIn[Input_Data.GetInputDataIndex()][DestinationIndex] *
Input_Chart1Multiplier.GetFloat() + Input_Chart1Addition.GetFloat()) -
    (Chart2Array[Chart2Index] * Input_Chart2Multiplier.GetFloat());
}

    sc.EarliestUpdateSubgraphDataArrayIndex = CalculationStartIndex;

}
/*=====*/

SCSFExport scsf_DifferenceNetChange(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Difference = sc.Subgraph[0];
    SCInputRef Input_Chart2Number = sc.Input[3];

    if (sc.SetDefaults)
    {
        //Calculates the difference of the bar to bar price change between two charts
        sc.GraphName = "Bar Price Change Difference - 2 Chart";

        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.GraphRegion = 1;

        Subgraph_Difference.Name = "Difference";
        Subgraph_Difference.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Difference.PrimaryColor = RGB(0,255,0);
        Subgraph_Difference.DrawZeros = false;

        Input_Chart2Number.Name = "Chart 2 Number";
        Input_Chart2Number.SetChartNumber(1);

        return;
    }

    SCFloatArray Chart2Array;
    sc.GetCharArray(Input_Chart2Number.GetChartNumber()*-1, SC_LAST, Chart2Array);

    if (Chart2Array.GetArraySize() == 0)
        return;

    sc.DataStartIndex = 1;

    for (int Chart1BarIndex = max(1, sc.UpdateStartIndex); Chart1BarIndex < sc.ArraySize; Chart1BarIndex++)
    {
        int Chart2Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart2Number.GetChartNumber(),
Chart1BarIndex);

        Subgraph_Difference[Chart1BarIndex] = (sc.BaseDataIn[SC_LAST][Chart1BarIndex] - sc.BaseDataIn[SC_LAST]
[Chart1BarIndex - 1]) - (Chart2Array[Chart2Index] - Chart2Array[Chart2Index - 1]);
    }

```

```

SCString Chart1Name = sc.GetStudyNameFromChart(sc.ChartNumber, 0);
SCString Chart2Name = sc.GetStudyNameFromChart(Input_Chart2Number.GetChartNumber(), 0);
sc.GraphName.Format("Bar Change Difference %s - %s", Chart1Name.GetChars(), Chart2Name.GetChars());
}

/*=====*/
SCSFExport scsf_RatioSingleLine(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Ratio = sc.Subgraph[0];
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_Chart2Number = sc.Input[3];
    SCInputRef Input_Chart1Multiplier = sc.Input[4];
    SCInputRef Input_Chart2Multiplier = sc.Input[5];
    SCInputRef Input_UseLatestSourceDataForLastBar = sc.Input[6];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Ratio (Single Line)";

        sc.ValueFormat = 2;
        sc.GraphRegion = 1;

        Subgraph_Ratio.Name = "Ratio";
        Subgraph_Ratio.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Ratio.PrimaryColor = RGB(0,255,0);
        Subgraph_Ratio.DrawZeros = false;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_Chart2Number.Name = "Chart 2 Number";
        Input_Chart2Number.SetChartNumber(1);

        Input_Chart1Multiplier.Name = "Chart 1 Multiplier";
        Input_Chart1Multiplier.SetFloat(1.0f);

        Input_Chart2Multiplier.Name = "Chart 2 Multiplier";
        Input_Chart2Multiplier.SetFloat(1.0f);

        Input_UseLatestSourceDataForLastBar.Name = "Use Latest Source Data For Last Bar";
        Input_UseLatestSourceDataForLastBar.SetYesNo(0);

        return;
    }

    SCFloatArray Chart2Array;
    sc.GetCharArray(Input_Chart2Number.GetChartNumber()*-1, Input_Data.GetInputDataIndex(), Chart2Array);

    if (Chart2Array.GetArraySize() == 0)
        return;

    for (int BarIndex = sc.UpdateStartIndex; BarIndex < sc.ArraySize; BarIndex++)
    {
        int Chart2Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart2Number.GetChartNumber(), BarIndex);

        //When 'use latest source data for last bar' is set to Yes, replay is not running and at last bar in the destination chart,
        then use the data from the very latest bar in the source chart.
        if(Input_UseLatestSourceDataForLastBar.GetYesNo() && !sc.IsReplayRunning() && BarIndex == sc.ArraySize - 1)
            Chart2Index = Chart2Array.GetArraySize() - 1;
    }
}

```



```

    if (sc.BaseDataIn[Input_Data.GetInputDataIndex()][BarIndex] == 0
        || Chart2Array[Chart2Index] == 0)
    {
        Subgraph_Ratio[BarIndex] = 0;
        continue;
    }

    Subgraph_Ratio[BarIndex] = (sc.BaseDataIn[Input_Data.GetInputDataIndex()][BarIndex] *
Input_Chart1Multiplier.GetFloat())
        / (Chart2Array[Chart2Index] * Input_Chart2Multiplier.GetFloat());
}

SCString Chart1Name = sc.GetStudyNameFromChart(sc.ChartNumber, 0);
SCString Chart2Name = sc.GetStudyNameFromChart(Input_Chart2Number.GetChartNumber(), 0);
sc.GraphName.Format("Ratio %s / %s", Chart1Name.GetChars(), Chart2Name.GetChars());
}

/*=====*/
SCSFExport scsf_OverlayBar(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
    SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
    SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
    SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];
    SCSubgraphRef Subgraph_Unused11 = sc.Subgraph[11];
    SCSubgraphRef Subgraph_Unused12 = sc.Subgraph[12];

    SCInputRef Input_OverlayFromSameChartAsStudy = sc.Input[0];
    SCInputRef Input_ChartOverlayNumber = sc.Input[3];
    SCInputRef Input_Multiplier = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Overlay (Bar)";

        sc.UseGlobalChartColors = 0;

        //When the ability to access all the variables of another chart is supported,
        //this should be changed to get the value format from the source chart
        sc.ValueFormat = VALUEFORMAT_INHERITED;

        sc.GraphRegion = 1;
        sc.ScaleRangeType = SCALE_INDEPENDENT;
        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        Subgraph_Open.Name = "Open";
        Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Open.PrimaryColor = RGB(0,255,0);
        Subgraph_Open.SecondaryColor = RGB(0,255,0);
        Subgraph_Open.SecondaryColorUsed = 1;
        Subgraph_Open.DrawZeros = false;

        Subgraph_High.Name = "High";
        Subgraph_High.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_High.PrimaryColor = RGB(128,255,128);
        Subgraph_High.DrawZeros = false;

```

```

Subgraph_Low.Name = "Low";
Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Low.PrimaryColor = RGB(255,0,0);
Subgraph_Low.SecondaryColor = RGB(255,0,0);
Subgraph_Low.SecondaryColorUsed = 1;
Subgraph_Low.DrawZeros = false;

Subgraph_Last.Name = "Last";
Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Last.PrimaryColor = RGB(255,128,128);
Subgraph_Last.DrawZeros = false;
Subgraph_Last.LineLabel = LL_DISPLAY_VALUE | LL_VALUE_ALIGN_VALUES_SCALE |
LL_VALUE_ALIGN_CENTER;

Subgraph_Volume.Name = "Volume";
Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Volume.PrimaryColor = RGB(255,0,0);
Subgraph_Volume.DrawZeros = false;

Subgraph_OpenInterest.Name = "# of Trades / OI";
Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OpenInterest.PrimaryColor = RGB(0,0,255);
Subgraph_OpenInterest.DrawZeros = false;

Subgraph_OHLCAvg.Name = "OHLC Avg";
Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OHLCAvg.PrimaryColor = RGB(127,0,255);
Subgraph_OHLCAvg.DrawZeros = false;

Subgraph_HLCAvg.Name = "HLC Avg";
Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);
Subgraph_HLCAvg.DrawZeros = false;

Subgraph_HLAvg.Name = "HL Avg";
Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLAvg.PrimaryColor = RGB(0,127,255);
Subgraph_HLAvg.DrawZeros = false;

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);
Subgraph_BidVol.DrawZeros = false;

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);
Subgraph_AskVol.DrawZeros = false;

Subgraph_Unused11.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Unused12.DrawStyle = DRAWSTYLE_IGNORE;

Input_OverlayFromSameChartAsStudy.Name = "Overlay from Same Chart as Study";
Input_OverlayFromSameChartAsStudy.SetYesNo(false);

Input_ChartOverlayNumber.Name = "Chart Number To Overlay";
Input_ChartOverlayNumber.SetChartNumber(1);

Input_Multiplier.Name = "Multiplier";
Input_Multiplier.SetFloat(1.0f);

return;
}

```

```

if (Input_OverlayFromSameChartAsStudy.GetYesNo())
    Input_ChartOverlayNumber.SetChartNumber(sc.ChartNumber);

SCGraphData ChartOverlayArrays;
sc.GetChartData(-Input_ChartOverlayNumber.GetChartNumber(), ChartOverlayArrays);

if (ChartOverlayArrays[SC_OPEN].GetArraySize() == 0)
    return;

for (int DataIndex = sc.UpdateStartIndex; DataIndex < sc.ArraySize; DataIndex++)
{
    const int OverlayBarIndex = sc.GetNearestMatchForDateTimeIndex(Input_ChartOverlayNumber.GetChartNumber(),
DataIndex);

    for (int SubgraphIndex = 0; SubgraphIndex < ChartOverlayArrays.GetArraySize(); SubgraphIndex++)
    {
        sc.Subgraph[SubgraphIndex][DataIndex] = ChartOverlayArrays[SubgraphIndex][OverlayBarIndex] *
Input_Multiplier.GetFloat();
    }
}

SCString ChartOverlayName = sc.GetStudyNameFromChart(Input_ChartOverlayNumber.GetChartNumber(), 0);
sc.GraphName.Format("%s Overlay", ChartOverlayName.GetChars());
}

/*=====*/
SCSFExport scsf_HorizontalLines(SCStudyInterfaceRef sc)
{
    const int NumberOfLines = SC_SUBGRAPHS_AVAILABLE - 1;
    SCInputRef Input_SetAllSubgraphsToBeSameAsFirst = sc.Input[SC_SUBGRAPHS_AVAILABLE - 1];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Horizontal Lines";

        sc.ValueFormat = 3;
        sc.AutoLoop = 1;
        sc.DisplayStudyInputValues = false;

        for (int SubgraphIndex = 0; SubgraphIndex < NumberOfLines; SubgraphIndex++)
        {
            sc.Subgraph[SubgraphIndex].Name.Format("Line%d", SubgraphIndex + 1);
            sc.Subgraph[SubgraphIndex].DrawStyle = DRAWSTYLE_IGNORE;
            sc.Subgraph[SubgraphIndex].PrimaryColor = RGB(0,255,0);
            sc.Subgraph[SubgraphIndex].DrawZeros = true;
            sc.Subgraph[SubgraphIndex].DisplayNameValueInWindowsFlags = false;

            sc.Input[SubgraphIndex].Name.Format("Line%d Value", SubgraphIndex + 1);
            sc.Input[SubgraphIndex].SetFloat(1.0f);
            sc.Input[SubgraphIndex].SetFloatLimits(-FLT_MAX, FLT_MAX);
        }

        sc.Subgraph[0].DrawStyle = DRAWSTYLE_LINE;

        Input_SetAllSubgraphsToBeSameAsFirst.Name = "Set All Subgraph Display Settings to Be Same as First";
        Input_SetAllSubgraphsToBeSameAsFirst.SetYesNo(false);

        return;
    }

    for (int SubgraphIndex = 0; SubgraphIndex < NumberOfLines; SubgraphIndex++)

```

```

{
    if (sc.Subgraph[SubgraphIndex].DrawStyle == DRAWSTYLE_IGNORE)
        continue;

    if (sc.IsFullRecalculation
        && sc.Index == 0
        && Input_SetAllSubgraphsToBeSameAsFirst.GetYesNo()
        && SubgraphIndex > 0)
    {
        sc.Subgraph[SubgraphIndex].DrawStyle = sc.Subgraph[0].DrawStyle;
        sc.Subgraph[SubgraphIndex].PrimaryColor = sc.Subgraph[0].PrimaryColor;
        sc.Subgraph[SubgraphIndex].LineStyle = sc.Subgraph[0].LineStyle;
        sc.Subgraph[SubgraphIndex].LineWidth = sc.Subgraph[0].LineWidth;
        sc.Subgraph[SubgraphIndex].LineLabel = sc.Subgraph[0].LineLabel;
    }

    //Only do this on the first iteration for efficiency
    if (sc.IsFullRecalculation && sc.Index == 0)
        sc.Input[SubgraphIndex].Name.Format("%s Value",sc.Subgraph[SubgraphIndex].Name.GetChars());

    sc.Subgraph[SubgraphIndex][sc.Index] = sc.Input[SubgraphIndex].GetFloat();
}
}

/*=====*/
SCSFExport scsf_HorizontalLinesAtIncrement(SCStudyInterfaceRef sc)
{
    const int NumberOfLines = SC_SUBGRAPHS_AVAILABLE - 1;

    SCInputRef Input_StartValue = sc.Input[0];
    SCInputRef Input_LineIncrement = sc.Input[1];
    SCInputRef Input_SetAllSubgraphsToBeSameAsFirst = sc.Input[2];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Horizontal Lines at Increment";

        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.DisplayStudyInputValues = false;
        sc.ScaleRangeType = SCALE_SAMEASREGION;
        sc.GraphRegion = 0;

        for (int SubgraphIndex = 0; SubgraphIndex < NumberOfLines; SubgraphIndex++)
        {
            sc.Subgraph[SubgraphIndex].Name.Format("Line%d", SubgraphIndex + 1);
            sc.Subgraph[SubgraphIndex].DrawStyle = DRAWSTYLE_LINE;
            sc.Subgraph[SubgraphIndex].PrimaryColor = RGB(0,255,0);
            sc.Subgraph[SubgraphIndex].DrawZeros = false;
            sc.Subgraph[SubgraphIndex].DisplayNameValueInWindowsFlags = false;
        }

        Input_SetAllSubgraphsToBeSameAsFirst.Name = "Set All Subgraph Display Settings to Be Same as First";
        Input_SetAllSubgraphsToBeSameAsFirst.SetYesNo(false);

        Input_StartValue.Name = "Start Value";
        Input_StartValue.SetFloat(0);

        Input_LineIncrement.Name = "Line Increment";
        Input_LineIncrement.SetFloat(1);

        return;
    }

    for(int BarIndex = sc.UpdateStartIndex; BarIndex < sc.ArraySize; BarIndex++)

```

```

{
    float CurrentLineValue = Input_StartValue.GetFloat();

    for (int SubgraphIndex = 0; SubgraphIndex < NumberOfLines; SubgraphIndex++)
    {
        if (sc.Subgraph[SubgraphIndex].DrawStyle == DRAWSTYLE_IGNORE)
            continue;

        if (sc.IsFullRecalculation
            && BarIndex == 0
            && Input_SetAllSubgraphsToBeSameAsFirst.GetYesNo()
            && SubgraphIndex > 0)
        {
            sc.Subgraph[SubgraphIndex].DrawStyle = sc.Subgraph[0].DrawStyle;
            sc.Subgraph[SubgraphIndex].PrimaryColor = sc.Subgraph[0].PrimaryColor;
            sc.Subgraph[SubgraphIndex].LineStyle = sc.Subgraph[0].LineStyle;
            sc.Subgraph[SubgraphIndex].LineWidth = sc.Subgraph[0].LineWidth;
            sc.Subgraph[SubgraphIndex].LineLabel = sc.Subgraph[0].LineLabel;
        }

        sc.Subgraph[SubgraphIndex][BarIndex] = CurrentLineValue;

        CurrentLineValue += Input_LineIncrement.GetFloat();
    }
}

}

/*=====*/
SCSFExport scsf_RelativeMomentumIndex(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_RMI = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Line1 = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Line2 = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Midline = sc.Subgraph[3];

    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_RMILength = sc.Input[3];
    SCInputRef Input_RMI_MALength = sc.Input[4];
    SCInputRef Input_RMI_MAType = sc.Input[5];
    SCInputRef Input_Line1Value = sc.Input[6];
    SCInputRef Input_Line2Value = sc.Input[7];
    SCInputRef Input_MidlineValue = sc.Input[8];

    SCFloatArrayRef Array_UpTemp = Subgraph_RMI.Arrays[0];
    SCFloatArrayRef Array_DownTemp = Subgraph_RMI.Arrays[1];
    SCFloatArrayRef Array_SmoothedUpTemp = Subgraph_RMI.Arrays[2];
    SCFloatArrayRef Array_SmoothedDownTemp = Subgraph_RMI.Arrays[3];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Relative Momentum Index";

        sc.GraphRegion = 1;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_RMI.Name = "RMI";
        Subgraph_RMI.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_RMI.PrimaryColor = RGB(0, 255, 0);
        Subgraph_RMI.DrawZeros = true;

        Subgraph_Line1.Name = "Overbought";
        Subgraph_Line1.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Line1.PrimaryColor = RGB(128, 128, 128);
    }
}

```

```

Subgraph_Line1.DrawZeros = true;

Subgraph_Line2.Name = "Oversold";
Subgraph_Line2.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Line2.PrimaryColor = RGB(128, 128, 128);
Subgraph_Line2.DrawZeros = true;

Subgraph_Midline.Name = "Midline";
Subgraph_Midline.DrawStyle = DRAWSTYLE_LINE;
Subgraph_Midline.PrimaryColor = RGB(128, 128, 128);
Subgraph_Midline.DrawZeros = true;

Input_Data.Name = "Input Data";
Input_Data.SetInputDataIndex(SC_LAST);

Input_RMILength.Name = "RMI Length";
Input_RMILength.SetInt(5);
Input_RMILength.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_RMI_MALength.Name = "RMI Moving Average Length";
Input_RMI_MALength.SetInt(9);
Input_RMI_MALength.SetIntLimits(1, MAX_STUDY_LENGTH);

Input_RMI_MAType.Name = "RMI Moving Average Type";
Input_RMI_MAType.SetMovAvgType(MOVAVGTYPE_SIMPLE);

Input_Line1Value.Name = "Overbought Value";
Input_Line1Value.SetFloat(70.0f);

Input_Line2Value.Name = "Oversold Value";
Input_Line2Value.SetFloat(30.0f);

Input_MidlineValue.Name = "Midline Value";
Input_MidlineValue.SetFloat(50.0f);

return;
}

sc.DataStartIndex = max(Input_RMILength.GetInt(), Input_RMI_MALength.GetInt());

float previousValue = sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index - Input_RMILength.GetInt()];
float currentValue = sc.BaseDataIn[Input_Data.GetInputDataIndex()][sc.Index];

if (currentValue > previousValue)
{
    Array_UpTemp[sc.Index] = currentValue - previousValue;
    Array_DownTemp[sc.Index] = 0.0f;
}
else
{
    Array_UpTemp[sc.Index] = 0;
    Array_DownTemp[sc.Index] = previousValue - currentValue;
}

sc.MovingAverage(Array_UpTemp, Array_SmoothedUpTemp, Input_RMI_MAType.GetMovAvgType(),
Input_RMI_MALength.GetInt());
sc.MovingAverage(Array_DownTemp, Array_SmoothedDownTemp, Input_RMI_MAType.GetMovAvgType(),
Input_RMI_MALength.GetInt());

if (Array_SmoothedUpTemp[sc.Index] + Array_SmoothedDownTemp[sc.Index] != 0.0f)
{
    Subgraph_RMI[sc.Index] = 100 * Array_SmoothedUpTemp[sc.Index] / (Array_SmoothedUpTemp[sc.Index] +
Array_SmoothedDownTemp[sc.Index]);
}

```

```

}
else
{
    Subgraph_RMI[sc.Index] = Subgraph_RMI[sc.Index - 1];
}

Subgraph_Line1[sc.Index] = Input_Line1Value.GetFloat();
Subgraph_Line2[sc.Index] = Input_Line2Value.GetFloat();
Subgraph_Midline[sc.Index] = Input_MidlineValue.GetFloat();
}

/*=====*/
SCSFExport scsf_OverlayNonSync(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Open = sc.Subgraph[0];
    SCSubgraphRef Subgraph_High = sc.Subgraph[1];
    SCSubgraphRef Subgraph_Low = sc.Subgraph[2];
    SCSubgraphRef Subgraph_Last = sc.Subgraph[3];
    SCSubgraphRef Subgraph_Volume = sc.Subgraph[4];
    SCSubgraphRef Subgraph_OpenInterest = sc.Subgraph[5];
    SCSubgraphRef Subgraph_OHLCAvg = sc.Subgraph[6];
    SCSubgraphRef Subgraph_HLCAvg = sc.Subgraph[7];
    SCSubgraphRef Subgraph_HLAvg = sc.Subgraph[8];
    SCSubgraphRef Subgraph_BidVol = sc.Subgraph[9];
    SCSubgraphRef Subgraph_AskVol = sc.Subgraph[10];
    SCSubgraphRef Subgraph_Unused11 = sc.Subgraph[11];
    SCSubgraphRef Subgraph_Unused12 = sc.Subgraph[12];

    SCInputRef Input_NumberOfBars = sc.Input[2];
    SCInputRef Input_ChartOverlayNumber = sc.Input[3];
    SCInputRef Input_Multiplier = sc.Input[4];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Overlay (Non-Sync)";

        sc.UseGlobalChartColors = 0;

        sc.ValueFormat = 2;
        sc.GraphRegion = 0;
        sc.ScaleRangeType = SCALE_INDEPENDENT;
        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        Subgraph_Open.Name = "Open";
        Subgraph_Open.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Open.PrimaryColor = RGB(0,255,0);
        Subgraph_Open.SecondaryColor = RGB(0,255,0);
        Subgraph_Open.SecondaryColorUsed = 1;
        Subgraph_Open.DrawZeros = false;

        Subgraph_High.Name = "High";
        Subgraph_High.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_High.PrimaryColor = RGB(128,255,128);
        Subgraph_High.DrawZeros = false;

        Subgraph_Low.Name = "Low";
        Subgraph_Low.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Low.PrimaryColor = RGB(255,0,0);
        Subgraph_Low.SecondaryColor = RGB(255,0,0);
        Subgraph_Low.SecondaryColorUsed = 1;
        Subgraph_Low.DrawZeros = false;

        Subgraph_Last.Name = "Last";
        Subgraph_Last.DrawStyle = DRAWSTYLE_LINE;

```



```

Subgraph_Last.PrimaryColor = RGB(255,128,128);
Subgraph_Last.DrawZeros = false;

Subgraph_Volume.Name = "Volume";
Subgraph_Volume.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Volume.PrimaryColor = RGB(255,0,0);
Subgraph_Volume.DrawZeros = false;

Subgraph_OpenInterest.Name = "# of Trades / OI";
Subgraph_OpenInterest.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OpenInterest.PrimaryColor = RGB(0,0,255);
Subgraph_OpenInterest.DrawZeros = false;

Subgraph_OHLCAvg.Name = "OHLC Avg";
Subgraph_OHLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_OHLCAvg.PrimaryColor = RGB(127,0,255);
Subgraph_OHLCAvg.DrawZeros = false;

Subgraph_HLCAvg.Name = "HLC Avg";
Subgraph_HLCAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLCAvg.PrimaryColor = RGB(0,255,255);
Subgraph_HLCAvg.DrawZeros = false;

Subgraph_HLAvg.Name = "HL Avg";
Subgraph_HLAvg.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_HLAvg.PrimaryColor = RGB(0,127,255);
Subgraph_HLAvg.DrawZeros = false;

Subgraph_BidVol.Name = "Bid Vol";
Subgraph_BidVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_BidVol.PrimaryColor = RGB(0,255,0);
Subgraph_BidVol.DrawZeros = false;

Subgraph_AskVol.Name = "Ask Vol";
Subgraph_AskVol.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_AskVol.PrimaryColor = RGB(0,255,0);
Subgraph_AskVol.DrawZeros = false;


Subgraph_Unused11.DrawStyle = DRAWSTYLE_IGNORE;
Subgraph_Unused12.DrawStyle = DRAWSTYLE_IGNORE;

Input_NumberOfBars.Name = "Number of bars to Overlay";
Input_NumberOfBars.SetInt(500);
Input_NumberOfBars.SetIntLimits(1,MAX_STUDY_LENGTH);

Input_ChartOverlayNumber.Name = "Chart Number To Overlay";
Input_ChartOverlayNumber.SetChartNumber(1);

Input_Multiplier.Name = "Multiplier";
Input_Multiplier.SetFloat(1.0f);

return;
}

SCGraphData ChartOverlayArrays;
sc.GetChartData(Input_ChartOverlayNumber.GetChartNumber(), ChartOverlayArrays);

int OverlayArraySize = ChartOverlayArrays[0].GetArraySize();
if (OverlayArraySize == 0)
    return;

for (int Index = sc.ArraySize - 1; (Index >= sc.ArraySize - Input_NumberOfBars.GetInt()) && (Index >= 0); Index--)
{
    OverlayArraySize--;

```



```

    for (int i2 = 0; i2 < ChartOverlayArrays.GetArraySize(); i2++)
    {
        if (OverlayArraySize >= 0)
        {
            sc.Subgraph[i2][Index] = ChartOverlayArrays[i2][OverlayArraySize] * Input_Multiplier.GetFloat();
        }
        else
            sc.Subgraph[i2][Index] = 0.0f;
    }
}

for (int Index = sc.ArraySize - Input_NumberOfBars.GetInt() - 1; (Index >= sc.ArraySize - sc.UpdateStartIndex) &&
(Index >= 0); Index--)
{
    for (int i2 = 0; i2 < ChartOverlayArrays.GetArraySize(); i2++)
        sc.Subgraph[i2][Index] = 0.0f;
}

SCString ChartOverlayName = sc.GetStudyNameFromChart(Input_ChartOverlayNumber.GetChartNumber(), 0);
sc.GraphName.Format("%s Overlay", ChartOverlayName.GetChars());
}

/*=====*/
SCSFExport scsf_OnBalanceOpenInterestShortTerm(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_OBOI = sc.Subgraph[0];
    SCSubgraphRef Subgraph_Temp = sc.Subgraph[1];
    SCInputRef Input_Length = sc.Input[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "On Balance Open Interest - Short Term";

        sc.ValueFormat = 0;
        sc.AutoLoop = 1;

        Subgraph_OBOI.Name = "OBOI";
        Subgraph_OBOI.PrimaryColor = RGB(0,255,0);
        Subgraph_OBOI.DrawStyle= DRAWSTYLE_LINE;
        Subgraph_OBOI.DrawZeros= true;

        Input_Length.Name = "Length";
        Input_Length.SetInt(10);
        Input_Length.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }

    sc.DataStartIndex= Input_Length.GetInt();
    if (sc.Index < 1)
        return;

    if(sc.Close[sc.Index ] > sc.Close[sc.Index-1])
    {
        Subgraph_Temp[sc.Index] = sc.OpenInterest[sc.Index];
    }
    else if(sc.Close[sc.Index ] < sc.Close[sc.Index-1])
    {
        Subgraph_Temp[sc.Index] = -sc.OpenInterest[sc.Index];
    }
    else
    {
        Subgraph_Temp[sc.Index] = 0.0f;
    }
}

```

```

if(sc.Index < Input_Length.GetInt())
{
    Subgraph_OBOI[sc.Index] = Subgraph_OBOI[sc.Index - 1] + Subgraph_Temp[sc.Index];
}
else
{
    Subgraph_OBOI[sc.Index] = Subgraph_OBOI[sc.Index - 1] + Subgraph_Temp[sc.Index] - Subgraph_Temp[sc.Index
- Input_Length.GetInt()];
}
}

```

```

/*=====*/
/*

```

Coppock Curve = 10-period WMA of 14-period RoC + 11-period RoC

WMA = Weighted moving average

RoC = Rate-of-Change

ROC = [(Close - Close n periods ago) / (Close n periods ago)] * 100

*/

```

SCSFExport scsf_CoppockCurve(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_CoppockCurve = sc.Subgraph[0];
    SCFloatArrayRef Array_RateOfChangeSum = sc.Subgraph[0].Arrays[0];
    SCInputRef Input_Data = sc.Input[0];
    SCInputRef Input_MovingAverageLength = sc.Input[1];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Coppock Curve";

        sc.GraphRegion = 1;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_CoppockCurve.Name = "CC";
        Subgraph_CoppockCurve.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_CoppockCurve.PrimaryColor = RGB(0, 255, 0);
        Subgraph_CoppockCurve.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        Input_MovingAverageLength.Name = "Moving Average Length";
        Input_MovingAverageLength.SetInt(10);
        Input_MovingAverageLength.SetIntLimits(1, MAX_STUDY_LENGTH);

        return;
    }
}

```

```

//sc.DataStartIndex = MovingAverageLength.GetInt();

```

```

if (sc.Index < sc.DataStartIndex)
    return;

```

```

float ROC1 = 0.0;

```

```

float ROC2 = 0.0;

```

```

int InputDataIndex = Input_Data.GetInputDataIndex();

```

```
ROC1 = ((sc.BaseData[InputDataIndex][sc.Index] - sc.BaseData[InputDataIndex][sc.Index - 14]) /  
sc.BaseData[InputDataIndex][sc.Index - 14]) * 100;
```

```
ROC2 = ((sc.BaseData[InputDataIndex][sc.Index] - sc.BaseData[InputDataIndex][sc.Index - 11]) /  
sc.BaseData[InputDataIndex][sc.Index - 11]) * 100;
```

```
Array_RateOfChangeSum[sc.Index] = ROC1 + ROC2;
```

```
sc.WeightedMovingAverage(Array_RateOfChangeSum, Subgraph_CoppockCurve,  
Input_MovingAverageLength.GetInt());
```

```
}
```

```
/*=====*/
```

```
SCSFExport scsf_HurstExponent(SCStudyInterfaceRef sc)
```

```
{
```

```
SCSubgraphRef Subgraph_HurstExponent = sc.Subgraph[0];
```

```
SCInputRef Input_Data = sc.Input[0];
```

```
SCInputRef Input_LengthIndex = sc.Input[1];
```

```
SCInputRef Input_PerformCumulativeCalculation = sc.Input[2];
```

```
if (sc.SetDefaults)
```

```
{
```

```
sc.GraphName = "Hurst Exponent";
```

```
sc.GraphRegion = 1;
```

```
sc.ValueFormat = 3;
```

```
sc.AutoLoop = 0;
```

```
Subgraph_HurstExponent.Name = "HE";
```

```
Subgraph_HurstExponent.DrawStyle = DRAWSTYLE_LINE;
```

```
Subgraph_HurstExponent.PrimaryColor = RGB(0, 255, 0);
```

```
Subgraph_HurstExponent.LineWidth = 1;
```

```
Subgraph_HurstExponent.DrawZeros = true;
```

```
Input_Data.Name = "Input Data";
```

```
Input_Data.SetInputDataIndex(SC_LAST);
```

```
Input_LengthIndex.Name = "Length";
```

```
Input_LengthIndex.SetCustomInputStrings("32;64;128");
```

```
Input_LengthIndex.SetCustomInputIndex(0);
```

```
Input_PerformCumulativeCalculation.Name = "Perform Cumulative Calculation Across All Chart Bars";
```

```
Input_PerformCumulativeCalculation.SetYesNo(false);
```

```
return;
```

```
}
```

```
sc.DataStartIndex = atoi(Input_LengthIndex.GetSelectedCustomString().GetChars())-1;
```

```
for (int BarIndex = max(1,sc.UpdateStartIndex); BarIndex < sc.ArraySize; BarIndex++)
```

```
{
```

```
if(Input_PerformCumulativeCalculation.GetYesNo())
```

```
{
```

```
CumulativeHurstExponent_S(sc.BaseData[Input_Data.GetInputDataIndex()],
```

```
Subgraph_HurstExponent,
```

```
Subgraph_HurstExponent.Arrays[0],
```

```
Subgraph_HurstExponent.Arrays[1],
```

```
Subgraph_HurstExponent.Arrays[2],
```

```
Subgraph_HurstExponent.Arrays[3],
```

```

        Subgraph_HurstExponent.Arrays[4],
        Subgraph_HurstExponent.Arrays[5],
        Subgraph_HurstExponent.Arrays[6],
        Subgraph_HurstExponent.Arrays[7],
        Subgraph_HurstExponent.Arrays[8],
        Subgraph_HurstExponent.Arrays[9],
        Subgraph_HurstExponent.Arrays[10],
        Subgraph_HurstExponent.Arrays[11],
        sc.Subgraph[1].Arrays[0],
        sc.Subgraph[1].Arrays[1],
        BarIndex);

    }
    else
    {
        SCFloatArrayRef InputDataArray = sc.BaseData[Input_Data.GetInputDataIndex()];

        //Calculate Logarithmic Return
        sc.Subgraph[0].Arrays[0][BarIndex] = static_cast<float>((log(InputDataArray[BarIndex]/InputDataArray[BarIndex -
1])));

        sc.HurstExponent(sc.Subgraph[0].Arrays[0], Subgraph_HurstExponent, BarIndex, Input_LengthIndex.GetIndex());
    }
}

}

/*=====*/
SCSFExport scsf_MovingAverageCumulative(SCStudyInterfaceRef sc)
{
    SCSubgraphRef Subgraph_Average = sc.Subgraph[0];

    SCInputRef Input_Data = sc.Input[0];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Moving Average - Cumulative";

        sc.GraphRegion = 0;
        sc.ValueFormat = 2;
        sc.AutoLoop = 1;

        Subgraph_Average.Name = "Avg";
        Subgraph_Average.DrawStyle = DRAWSTYLE_LINE;
        Subgraph_Average.PrimaryColor = RGB(0, 255, 0);
        Subgraph_Average.LineWidth = 1;
        Subgraph_Average.DrawZeros = true;

        Input_Data.Name = "Input Data";
        Input_Data.SetInputDataIndex(SC_LAST);

        return;
    }

    sc.DataStartIndex = 1;

    sc.MovingAverageCumulative(sc.BaseData[Input_Data.GetInputDataIndex()], Subgraph_Average);
}

```

```

/*=====*/
SCSFExport scsf_CombinationSymbolChart(SCStudyInterfaceRef sc)
{

    SCInputRef Input_UseLatestSourceDataForLastBar = sc.Input[0];

    SCInputRef Input_Chart1Multiplier = sc.Input[1];

    SCInputRef Input_UseChart2 = sc.Input[2];
    SCInputRef Input_Chart2Number = sc.Input[3];
    SCInputRef Input_Chart2Multiplier = sc.Input[4];

    SCInputRef Input_UseChart3 = sc.Input[5];
    SCInputRef Input_Chart3Number = sc.Input[6];
    SCInputRef Input_Chart3Multiplier = sc.Input[7];

    SCInputRef Input_UseChart4 = sc.Input[8];
    SCInputRef Input_Chart4Number = sc.Input[9];
    SCInputRef Input_Chart4Multiplier = sc.Input[10];

    SCInputRef Input_UseChart5 = sc.Input[11];
    SCInputRef Input_Chart5Number = sc.Input[12];
    SCInputRef Input_Chart5Multiplier = sc.Input[13];

    SCInputRef Input_UseChart6 = sc.Input[14];
    SCInputRef Input_Chart6Number = sc.Input[15];
    SCInputRef Input_Chart6Multiplier = sc.Input[16];

    if (sc.SetDefaults)
    {
        sc.GraphName = "Combination Symbol Chart";
        sc.AutoLoop = 0;
        sc.ValueFormat = VALUEFORMAT_INHERITED;
        sc.GraphRegion = 1;

        sc.GraphUsesChartColors = 1;//Use the global color settings
        sc.GraphDrawType = GDT_OHLCBAR;
        sc.StandardChartHeader = 1;

        Input_UseLatestSourceDataForLastBar.Name = "Use Latest Source Data For Last Bar";
        Input_UseLatestSourceDataForLastBar.SetYesNo(false);

        Input_Chart1Multiplier.Name = "Chart 1 Multiplier";
        Input_Chart1Multiplier.SetFloat(1.0f);

        Input_UseChart2.Name = "Use Chart 2";
        Input_UseChart2.SetYesNo(true);

        Input_Chart2Number.Name = "Chart 2 Number";
        Input_Chart2Number.SetChartNumber(1);

        Input_Chart2Multiplier.Name = "Chart 2 Multiplier";
        Input_Chart2Multiplier.SetFloat(1.0f);

        Input_UseChart3.Name = "Use Chart 3";
        Input_UseChart3.SetYesNo(true);

        Input_Chart3Number.Name = "Chart 3 Number";
        Input_Chart3Number.SetChartNumber(1);

        Input_Chart3Multiplier.Name = "Chart 3 Multiplier";
        Input_Chart3Multiplier.SetFloat(1.0f);

        Input_UseChart4.Name = "Use Chart 4";

```

```

Input_UseChart4.SetYesNo(true);

Input_Chart4Number.Name = "Chart 4 Number";
Input_Chart4Number.SetChartNumber(1);

Input_Chart4Multiplier.Name = "Chart 4 Multiplier";
Input_Chart4Multiplier.SetFloat(1.0f);

Input_UseChart5.Name = "Use Chart 5";
Input_UseChart5.SetYesNo(true);

Input_Chart5Number.Name = "Chart 5 Number";
Input_Chart5Number.SetChartNumber(1);

Input_Chart5Multiplier.Name = "Chart 5 Multiplier";
Input_Chart5Multiplier.SetFloat(1.0f);

Input_UseChart6.Name = "Use Chart 6";
Input_UseChart6.SetYesNo(true);

Input_Chart6Number.Name = "Chart 6 Number";
Input_Chart6Number.SetChartNumber(1);

Input_Chart6Multiplier.Name = "Chart 6 Multiplier";
Input_Chart6Multiplier.SetFloat(1.0f);
return;
}

if (sc.IsFullRecalculation && sc.UpdateStartIndex == 0)
{
    for (int SubgraphIndex = 0; SubgraphIndex <= NUM_BASE_GRAPH_ARRAYS; ++SubgraphIndex)
    {
        sc.Subgraph[SubgraphIndex].Name = sc.GetStudySubgraphName(0, SubgraphIndex);
    }

    sc.GraphName = "Combination Chart: ";
    sc.GraphName += sc.GetChartName(sc.ChartNumber);

    if (Input_UseChart2.GetYesNo())
    {
        sc.GraphName += ", ";
        sc.GraphName += sc.GetChartName(Input_Chart2Number.GetChartNumber());
    }

    if (Input_UseChart3.GetYesNo())
    {
        sc.GraphName += ", ";
        sc.GraphName += sc.GetChartName(Input_Chart3Number.GetChartNumber());
    }

    if (Input_UseChart4.GetYesNo())
    {
        sc.GraphName += ", ";
        sc.GraphName += sc.GetChartName(Input_Chart4Number.GetChartNumber());
    }

    if (Input_UseChart5.GetYesNo())
    {
        sc.GraphName += ", ";
        sc.GraphName += sc.GetChartName(Input_Chart5Number.GetChartNumber());
    }

    if (Input_UseChart6.GetYesNo())
    {
        sc.GraphName += ", ";

```

```

        sc.GraphName += sc.GetChartName(Input_Chart6Number.GetChartNumber());
    }
}

SCGraphData Chart2BaseData;
if(Input_UseChart2.GetYesNo())
    sc.GetChartBaseData(-Input_Chart2Number.GetChartNumber(), Chart2BaseData);

SCGraphData Chart3BaseData;
if (Input_UseChart3.GetYesNo())
    sc.GetChartBaseData(-Input_Chart3Number.GetChartNumber(), Chart3BaseData);

SCGraphData Chart4BaseData;
if (Input_UseChart4.GetYesNo())
    sc.GetChartBaseData(-Input_Chart4Number.GetChartNumber(), Chart4BaseData);

SCGraphData Chart5BaseData;
if (Input_UseChart5.GetYesNo())
    sc.GetChartBaseData(-Input_Chart5Number.GetChartNumber(), Chart5BaseData);

SCGraphData Chart6BaseData;
if (Input_UseChart6.GetYesNo())
    sc.GetChartBaseData(-Input_Chart6Number.GetChartNumber(), Chart6BaseData);

int CalculationStartIndex = sc.GetCalculationStartIndexForStudy();

// Iterate through the bars
for (int Index = CalculationStartIndex; Index < sc.ArraySize; ++Index)
{
    int Chart2Index = 0;
    if (Input_UseChart2.GetYesNo())
        Chart2Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart2Number.GetChartNumber(), Index);

    int Chart3Index = 0;
    if (Input_UseChart3.GetYesNo())
        Chart3Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart3Number.GetChartNumber(), Index);

    int Chart4Index = 0;
    if (Input_UseChart4.GetYesNo())
        Chart4Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart4Number.GetChartNumber(), Index);

    int Chart5Index = 0;
    if (Input_UseChart5.GetYesNo())
        Chart5Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart5Number.GetChartNumber(), Index);

    int Chart6Index = 0;
    if (Input_UseChart6.GetYesNo())
        Chart6Index = sc.GetNearestMatchForDateTimeIndex(Input_Chart6Number.GetChartNumber(), Index);

    //When use latest source data for last bar is set to Yes, replay is not running and at last bar in the destination chart,
    then use the data from the very latest bar in the source charts.
    if (Input_UseLatestSourceDataForLastBar.GetYesNo() && !sc.IsReplayRunning()
        && Index == sc.ArraySize - 1)
    {
        if(Input_UseChart2.GetYesNo())
            Chart2Index = Chart2BaseData[SC_OPEN].GetArraySize() - 1;

        if (Input_UseChart3.GetYesNo())
            Chart3Index = Chart3BaseData[SC_OPEN].GetArraySize() - 1;

        if (Input_UseChart4.GetYesNo())
            Chart4Index = Chart4BaseData[SC_OPEN].GetArraySize() - 1;

        if (Input_UseChart5.GetYesNo())

```

```

        Chart5Index = Chart5BaseData[SC_OPEN].GetArraySize() - 1;

        if (Input_UseChart6.GetYesNo())
            Chart6Index = Chart6BaseData[SC_OPEN].GetArraySize() - 1;
    }

    // Iterate through the subgraphs
    for (int SubgraphIndex = SC_OPEN; SubgraphIndex <= SC_NUM_TRADES; SubgraphIndex++)
    {
        sc.Subgraph[SubgraphIndex][Index] = sc.BaseDataIn[SubgraphIndex][Index] * Input_Chart1Multiplier.GetFloat();

        if (Input_UseChart2.GetYesNo())
            sc.Subgraph[SubgraphIndex][Index] += Chart2BaseData[SubgraphIndex][Chart2Index] *
Input_Chart2Multiplier.GetFloat();

        if (Input_UseChart3.GetYesNo())
            sc.Subgraph[SubgraphIndex][Index] += Chart3BaseData[SubgraphIndex][Chart3Index] *
Input_Chart3Multiplier.GetFloat();

        if (Input_UseChart4.GetYesNo())
            sc.Subgraph[SubgraphIndex][Index] += Chart4BaseData[SubgraphIndex][Chart4Index] *
Input_Chart4Multiplier.GetFloat();

        if (Input_UseChart5.GetYesNo())
            sc.Subgraph[SubgraphIndex][Index] += Chart5BaseData[SubgraphIndex][Chart5Index] *
Input_Chart5Multiplier.GetFloat();

        if (Input_UseChart6.GetYesNo())
            sc.Subgraph[SubgraphIndex][Index] += Chart6BaseData[SubgraphIndex][Chart6Index] *
Input_Chart6Multiplier.GetFloat();
    }

    sc.Subgraph[SC_HIGH][Index]
        = max(
            sc.Subgraph[SC_OPEN][Index],
            max(
                sc.Subgraph[SC_HIGH][Index],
                max(
                    sc.Subgraph[SC_LOW][Index],
                    sc.Subgraph[SC_LAST][Index]
                )
            )
        );

    sc.Subgraph[SC_LOW][Index]
        = min(
            sc.Subgraph[SC_OPEN][Index],
            min(
                sc.Subgraph[SC_HIGH][Index],
                min(
                    sc.Subgraph[SC_LOW][Index],
                    sc.Subgraph[SC_LAST][Index]
                )
            )
        );

    sc.CalculateOHLCAverages(Index);
}

sc.EarliestUpdateSubgraphDataArrayIndex = CalculationStartIndex;
}

```