

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

```
/*=====
This study creates a custom tick index from a list of charts. See the
description in the Add Study window to learn how to use this study.
=====*/
```

```
SCSFExport scsf_CustomTickIndex(SCStudyInterfaceRef sc)
```

```
{
    SCSubgraphRef Subgraph_Tick = sc.Subgraph[0];
```

```
    if (sc.SetDefaults)
```

```
    {
        // Set the configuration and defaults
```

```
        sc.GraphName = "Custom Tick Index";
```

```
        sc.StudyDescription = "This will calculate a custom tick index. Open all of the intraday charts you want to create a tick index from. This study needs to be applied to a separate chart. This chart can be one of the same symbols in the group. However, it needs to be a separate chart. This chart needs to have the Include Columns With No Data option on. The other charts should not. This chart needs to use a short time period per bar such as 10 seconds. A 10 second period would analyze the direction over a period of 10 seconds. Set the List of Chart Numbers input to a comma separated list of all the charts you want to create a tick index from. All the charts that the study refers to will automatically be set to the same time period as the chart it is applied to. A longer duration tick index consisting of bars can be created by referencing this study using the Study Overlay OHLC study on a longer duration chart. For efficiency, the chart that this study is applied to, and the charts which the study creates a tick index from, all should have the Days To Load setting set to a small number like 1. When the charts are opened and after data downloading from the data service, then to ensure the custom tick study has referenced all the data in all the charts it refers to, select Chart >> Reload on the menu to perform a full recalculation.";
```

```
        //sc.ReferencesOtherChart = 1; // true
```

```
        sc.UpdateAlways = 1; // true
```

```
        sc.DrawZeros = 1; // true
```

```
        Subgraph_Tick.Name = "Tick";
```

```
        Subgraph_Tick.DrawStyle = DRAWSTYLE_DASH;
```

```
        sc.TextInputName = "List of Chart Numbers (comma separated)";
```

```
        return;
```

```
    }
```

```
    // Do data processing
```

```
    // Get inputs
```

```
    // Make a copy of the list of chart numbers from the text input
```

```
    char ChartNumberList[256];
```

```
    strncpy(ChartNumberList, sc.TextInput.GetChars(), sizeof(ChartNumberList) - 1);
```

```
    // Get each of the chart numbers (from the text input) and put them into a
```

```
    // vector. The vector is static to limit the amount of memory
```

```
    // allocations and deallocations.
```

```
    static std::vector<int> ChartNumbers;
```

```
    ChartNumbers.clear();
```

```
    char* Token = strtok(ChartNumberList, ",");
```

```
    while (Token != NULL)
```

```
    {
        ChartNumbers.push_back(atoi(Token));
```

```
        // Get the next chart number
```

```
        Token = strtok(NULL, ",");
```

```
    }
```

```
    for (int DestIndex = sc.Index; DestIndex < sc.ArraySize; ++DestIndex)
```

```
        Subgraph_Tick[DestIndex] = 0.0f;
```

```

// Loop through each of the charts
for (std::vector<int>::iterator Iter_ChartNumber = ChartNumbers.begin(); Iter_ChartNumber != ChartNumbers.end();
++Iter_ChartNumber)
{
    int ChartNumber = *Iter_ChartNumber;

    // Get the base data for the chart
    SCGraphData BaseData;
    sc.GetChartBaseData(-ChartNumber, BaseData);

    // Get the closing price array for the current chart
    SCFloatArray SrcPriceArray = BaseData[SC_LAST];
    if (SrcPriceArray.GetArraySize() == 0)
        continue;

    // Loop through the indexes of the out array
    for (int DestIndex = sc.Index; DestIndex < sc.ArraySize; ++DestIndex)
    {
        // Find the index in the current chart matching our destination date
        int SrcIndex = sc.GetContainingIndexForDateTimeIndex(ChartNumber, DestIndex);

        if (SrcIndex < 1)
            continue; // Can't compare with the previous value

        // Count if it's an up tick or down tick
        if (SrcPriceArray[SrcIndex] > SrcPriceArray[SrcIndex - 1])
            ++Subgraph_Tick[DestIndex]; // Up
        else if (SrcPriceArray[SrcIndex] < SrcPriceArray[SrcIndex - 1])
            --Subgraph_Tick[DestIndex]; // Down
    }
}
}

```