20091220 StockChartX Settings 03

Malcolm Moore

Contents

Building the Standard Indicators

Managing the Three Stages

Chronologically, this “Indicators” area should be covered now, because at this stage the data has been loaded and the indicators are now due to be loaded – but, again this is rather complex, and it is not imperative to have the indicators included. So this area can be skipped and the Presentation stage can be worked through without having any reference to indicators.

The practice in developing a clean and fault minimised StockChartX presentation is that the presentation stage is much like putting on a finish coat of paint after the undercoats have been applied. With indicators, the problem is that these are also loaded like series, but the presentation of indicators is often mixed up with loading the indicators and this makes a messy software structure.

This part of these notes will not go into producing Custom Indicators as the structure for these is significantly more complex than described here, but once the concept of Standard (inbuilt) Indicators is understood, then it becomes practical to introduce and use Custom Indicators.

Creating a Simple Moving Average Indicator

When a standard indicator is added to a StockChartX object, the coding for doing this is rather straightforward. The easiest way to describe how to do this is with some sample code snippets together with some notes to explain what is going on.

Here is a broken up and heavily commented series of code snippets of a Simple Moving Average (“SMA”) being added to a Candlestick chart.

```
With frmDoc.StockChartX1
 .AddIndicatorSeries indSimpleMovingAverage, "SMA", 0, True
```

The “With” structure greatly simplifies the coding and minimises the amount of wording required to get the code in place.

On typing in part of .AddindicatorSeries and highlight this and then type in a space, the dropdown menu provides for 64 choices of standard built-in indicators that can be selected and highlighted. This process is a little bit fiddly in that the drop-down list only appears the first time that you highlight and click on the .AddindicatorSeries and add a space!

The *Key as String* term “SMA” in this case is the name of the Simple Moving Average indicator; the 1 refers to panel number 0 and the True relates to the “User Parameters” being available to be loaded into the StockChartX object.

```
.IndPropStr("SMA", 1) = ThisSymbol + ".Close"
```

The first user parameter uses the Key as above (“SMA”) with the number 1 makes a uniquely new Table heading in the StockChartX database is called “SMA”, and the contents of this indicator will be created from the existing table series called by the reference ThisSymbol & “.Close”

```
.IndPropInt("SMA", 2) = ShortSMA
```

The second user parameter uses the Key as above (“SMA”) with the number 2, and this parameter is an integer that defines the time constant of the Simple Moving Average by the variable defined as ShortSMA.

So now we have a Table made into the StockChartX object that is called SMA and the Simple Moving Average series will automatically be created from the “Close” values, using the time constant “ShortSMA”. This series will be created when the StockChartX object is updated with the object command as follows:

```
StockChartX1.Update
```

At this stage the Simple Moving Average (SMA) indicator series is now built into the StockChartX object.

The following code refers to the presentation of the SMA indicator and strictly should be with the presentation phase, but this data is usually directly associated with the loading of the indicator series and therefore it is shown here for expediency.

```
 .SeriesColor("SMA") = SMASignalColour
 .SeriesWeight("SMA") = SMASignalWidth
 .SeriesVisible("SMA") = False
End With
```

The variable SMASignalColour is loaded into the StockChartX object. This colour is a 32-bit long integer described by the RGB(0-255, 0-255, 0-255) subroutine, and/or by the Visual Basic 6 colours (vbRed, vbGreen, vbMagenta, vbBlack, vbYellow, vbBlue etc.).

The variable SMASignalWidth is a long integer that specified the width of the line, where 1 is a fine line and 4 is a rather thick line.

It is normal practice to keep the indicator series not visible unless they are required, and remember that the layers are sequentially painted on over the previous layers so the last object component made visible is on the top of the last layer.

This concept is important because this explains the hierarchy of how and where adding indicators sits when it comes to structuring a StockChartX instance. When looking at a typical indicator being added; the properties now speak for themselves. Here is all the above code snippets put together:

```
With StockChartX1
 .AddIndicatorSeries indSimpleMovingAverage, "SMA", 0, True
```

```
     .IndPropStr("SMA", 1) = ThisSymbol + ".Close"
     .IndPropInt("SMA", 2) = ShortSMA
     .SeriesColor("SMA") = SMASignalColour
     .SeriesWeight("SMA") = SMASignalWidth
     .SeriesVisible("SMA") = False
    End With
```

This code is now fairly straightforward and it can be adapted to fit into a much larger structure so that it could be called up as a subroutine from anywhere else and work without flaws.

Creating an Exponential Moving Average Indicator

This is an example of adding an Exponential Moving Average Indicator for 15 periods (EMA15) in a programme is shown here:

```
    '  Set up the EMA Indicator in this Command Button Subroutine.
Private Sub IndicatorEMA()
    '  Details on describing what these next parameters mean are described
    '  in the paragraphs below this example.
  With StockChartX1
    '  This standard Indicator is the EMA which in this case has the name
    '  "EMA15", which is to be loaded into Panel number 0 and the
    '  assist level is "True".
    .AddIndicatorSeries = indExponentialMovingAverage, "EMA15", 0, True
    '  This Indicator's Property String number 1 is the Data Source, which
    '  in this case is the internal close prices (as a string)
    .IndPropStr("EMA15", 1) = frmDocument.StockChartX1.Symbol & ".Close"
    '  This Indicator's Property String number 2 is the Default Periods,
    '  which in this case is the EMA Periods (as an Integer) is 15
    .IndPropInt("EMA15", 2) = 15
    '  This Indicator's Colour is set to vbBlue (which is a Long Integer)
    .SeriesColor("EMA15") = vbBlue
    '  This Indicator's Weight is set to 2 (which is a Long Integer)
    .SeriesWeight("EMA15") = 2
    '  Show the Indicator Dialogue Input Box – to make changes
    .ShowIndicatorDialog "EMA15"
    '  Update the OCX to load the EMA15 Series Indicator into the OCX
    .Update
  End With
End Sub
```

This structure is almost identical to the structure for the Simple moving average as described in the previous section. With a little bit of tweaking this subroutine could be structured to be virtually universal as follows:

```
Private Sub ChartEMA(ByVal PanelNumber As Long, ByVal Symbol As String,
ByVal Source As String, ByVal Period As Long, ByVal Colour As Long, ByVal
LineWidth As Long)
    Dim EMAName As String
    EMAName = "EMA " & CStr(Period)
  With StockChartX1
     .AddIndicatorSeries indExponentialMovingAverage, EMAName, PanelNumber,
True
    '  This Indicator's Property String number 1 is the Data Source, which
    '  in this case is the internal close prices (as a string)
    .IndPropStr(EMAName, 1) = Symbol + ".close"
    '  This Indicator's Property String number 2 is the Default Periods,
    '  which in this case is the EMA Periods (as an Integer) is 15
    .IndPropInt(EMAName, 2) = Period
    '  This Indicator's Colour is set to vbBlue (which is a Long Integer)
```

```
    .SeriesColor(EMAName) = Colour
    '  This Indicator's Weight is set to 2 (which is a Long Integer)
    .SeriesWeight(EMAName) = LineWidth
    '  Update the DLL to load the EMA15 Series Indicator into the OCX
    .Update
  End With
End Sub
```

This is quite a tidy subroutine that creates the EMA indicator directly under control of the calling parameters. If the commenting is left out it is even tighter.

Creating an MACD Indicator

With the (exponential) MACD there are several properties that need to be loaded so that the MACD will work correctly. From the TA.CHM document the nomenclature for the MACD is:

Parameters:
str Symbol
int Short Cycle
int Long Cycle
int Signal Periods

Note that there is no “Source” entry for the historical data, as StockChartX blindly assumes that the “.close” price will be used as the data source for the MACD; so do not include the specific data source (with & “.close”) as this will cause an error.

The terms “Long Cycle” and “Short Cycle” specifically refer to the time constants of the exponential moving averages (EMAs) that are subtracted from one another to create the MACD indicator.

The “Signal Periods” is an EMA that uses the MACD Indicator as its historical data source and hence this EMA trails the MACD. In StockChartX, the name of the trailing / signalling MACD is internally called the “XXXX SIGNAL”, and this is important because if the colour and weight of this line is to be managed then it has to be addressed as the “XXXX SIGNAL”. (Note the space between the two upper case words.)

Unfortunately, StockChartX names the Exponential MACD as MACD, even though it is an exponential MACD. The indicator name should include the prefix of “E” to make it EMACD, showing that it is an Exponentially formed MACD. Further the periodicity of the MACD is not included in the naming, and this can be addressed by a little bit of coding:

```
IndicatorName = "EMACD " & CStr(LongCycle) & ":" & CStr(ShortCycle)
```

Now, where the naming is XXXX above, this will translate to the Indicator name in the following subroutine, and at least the heading for the Signal trace will include the details about the EMACD on the screen!

Here is a highly functional Exponential MACD indicator done as a subroutine:

```
Private Sub ChartEMACD(ByVal PanelNumber As Long, _
            ByVal ThisSymbol As String, _
            ByVal LongCycle As Integer, _
            ByVal ShortCycle As Integer, _
            ByVal EMACDColour As Long, _
```

```
            ByVal EMACDWidth As Long, _
            ByVal SignalPeriods As Integer, _
            ByVal SignalColour As Long, _
            ByVal SignalWidth As Long)
    '  Details on describing what these next parameters mean are described
    '  in the paragraphs below this example.

    Dim IndicatorName As String
    IndicatorName = "EMACD " & CStr(LongCycle) & ":" & CStr(ShortCycle)

  With StockChartX1
    '  Set StockChartX so that it will not bomb out when comparing series
that have different lengths
     .IgnoreSeriesLengthErrors = True

     .AddIndicatorSeries indMACD, IndicatorName, PanelNumber, True
     '  This Indicator's Property String number 1 is the Data Source, which
     '  in this case is the internal close prices (as a string)
    .IndPropStr(IndicatorName, 1) = ThisSymbol
     '  This Indicator's Property String number 2 is the Default Periods,
     '  which in this case is the EMA Periods (as an Integer) is 15
    .IndPropInt(IndicatorName, 2) = LongCycle
    .IndPropInt(IndicatorName, 3) = ShortCycle
    .Update

    .IndPropInt(IndicatorName, 4) = SignalPeriods
    '  This Indicator's Colour is set to vbBlue (which is a Long Integer)
    .Update

    .SeriesColor(IndicatorName) = EMACDColour
    '  This Indicator's Colour is set to vbBlue (which is a Long Integer)
    .SeriesWeight(IndicatorName) = EMACDWidth
    '  This Indicator's Weight is set to 2 (which is a Long Integer)
    .SeriesColor(IndicatorName & " SIGNAL") = SignalColour
    '  This Indicator's Weight is set to 2 (which is a Long Integer)
    .SeriesWeight(IndicatorName & " SIGNAL") = SignalWidth
    '  Show the Indicator Dialogue Input Box - to make changes
    .Update

  End With
End Sub
```

This indicator is slightly more complicated because it is necessary to firstly add the indicator series names and settings and then create the series, then add the EMA time constant property for the trailing signal in the specific code:

```
     .IndPropInt(IndicatorName, 4) = SignalPeriods
     .Update
```

The code after these two lines of code is the cosmetics for the EMACD and trailing Signal. These cosmetic entries have no effect on the development of the EMACD indicator and trailing signal and these cosmetic entries form the first part of the overall presentation process.

A highly functional EMACD is included in the Copy sand Paste area at the back of this document.

More About Indicator Properties
From working with a few indicators it becomes apparent that there are a number of property types that need a little bit of explanation. Fundamentally, these property types (pt) are internal functions that process various parts of a sub-indicator to produce the necessary data for creating an indicator value.

Considering the EMA, the source for the data is under strSource (1), and the periods is under intPeriods (2)

12/21/2006 8:41:08 PM

White candle with black border = close above open & close above close yesterday.
White candle with red border = close above open & close below close yesterday.
Solid black candle = close below open and close above close yesterday.
Solid red candle = close below open and close below close yesterday

On inserting the **`".AddIndicatorSeries ="`** command, the OCX will ask for four parameters. The first is the name of the "***Indicator Type***" and in this case the Exponential Moving Average has **`indExponentialMovingAverage`** as the Type (note it starts with 'ind' for Indicator). Every Standard Indicator has its own name and this can be seen in the TA.CHM file (Technical Analysis Compacted HTML file). This very sketchy reference is mandatory.

The second parameter is the "***Key as a String***". What this really means is that every indicator that is put on the charts must have a unique name (or 'key'), and it is simply not safe to give a generic key of EMA because when you put in a second EMA indicator, that key will have been already used by the first EMA. One suggestion is to either have an incremented Public variable (long integer) and use this in addition with the series abbreviation, to make a unique key for example: **`("EMA" & Chart.lIndCount)`** or to use the EMA periods as the defining unique key so for example a 15 period EMA could have the Key as **`("EMA15")`** and either of these approaches should ensure a unique naming Key every time.

The third parameter is "***Panel as Long***". This tells StockChart which panel the indicator is going to go into; so somehow you have to know, or give implicit directions. The panel count starts at 0 for the price chart at the top of the screen and proceeds to 1 for the Volume Chart (at the bottom of the screen). If another panel is added, then the Volume chart moves into the new panel at the bottom of the screen and this leaves the older (lower-numbered) panel ready to take new indicator data.

What really happens is that the indicator is assigned to a panel, which can be changed at a later time, and note that if the indicator is **`.Visible = False`**, then the values of that indicator (or series) play no part in setting the charting scale parameters (until it is made visible in the presentation phase).

The fourth parameter is "***User Parameters as Boolean***" (True/False). What this is really referring to is a "User Panel" where the parameters can be changed from what are programmed into the indicator. This little panel about half the size of a playing card will pop up every time that the indicator is used. Once this situation has been worked out it is advisable to make the "True" parameter into a "False" and the User Panel will not come up!

Indicator Properties

Every Indicator has a few properties that it has to have and use, so that it can provide the necessary indication. In the associated help file (TA.CHM) the commands for associating properties of an indicator with that indicator are somewhat cryptic, and they need a little explanation, but typically as an example, they start as this:

```
.IndPropStr( Key, Number)=
```

and this:

```
.IndPropInt( Key, Number)=
```

In looking a little closer, the difference is that the first refers to a **string** entry while the second refers to a **long integer** entry. The "Key" indicator is the 'family' for the indicator, which is common for all the indicators on the chart, but the following number makes the property entry unique. These Indicator Properties are numbered from 1 (not from zero) and the basics of these properties are provided in TA.CHM for each indicator. For example in TA.CHM the properties for the EMA are:

```
Parameters:
str Source
int Periods
```

This terminology needs a bit of interpretation to bring into understandable code. The Key is the indicator family code for this chart, and referring to the line of code below; the Key is a string called "EMA150". This key is then associated with the number 1 to make it a unique first parameter, and it relates to a string that holds the data source. In this case, the data source is another field in the StockChartX flat file – the Close price for that period.

```
StockChartX1.IndPropStr("EMA150", 1) = StockChartX1.Symbol & ".Close"
```

This means that in this case StockchartX's Indicator Property String (number 1) will be called EMA150, and the source data will come as a string from inside the StockchartX object. Say for example the Symbol was CBA (for Commonwealth Bank of Australia) then the source of the numerical data would come from the internal string series called **`CBA.Close`**.

The second parameter associated with the EMA is the number of periods. This property is a long Integer, the Key in this case is (**`"EMA150"`**), and the parameter count is number 2, and this parameter is **`150`** (periods) in this case, and this time the code is for an integer as follows:

```
StockChartX1.IndPropInt("EMA150", 2) = 150
```

If you follow this general process, then virtually any indicator's properties can be loaded into StockChartX and the sequence is important.

In this code above, a Standard Indicator (that is already created in software in StockChartX) is added in the first line. The Standard Indicators all have names starting with "ind" and the next area "Naming the Indicators" provides a list of some of the most common Standard Indicator names. Note that the Standard Indicator coding that starts with 'indXXXX' is part of the StockChartX objects' enumerated internal code and it is not a string variable – even though it can be treated as one for the Standard Indicators. StockChartX interprets this standard indicator word as a number – not a word string.

With Custom Indicators, the code indCustomIndicator needs to be directly suffixed without "talkies", and StockChartX will directly interpret that code into an internal number.

Following the coding for the (Standard) Indicator Type, the local name for this indicator is to be provided inside "talkies". In this case the name was "SMA" and following that a specific panel number is provided to tell StockChartX as to where the new indicator series will be located, and (in this case) the specific panel number (1) is the panel under the top panel. The "True" means that the User Parameters (data) provided in the lines below this top line will be source for the structure of the Simple moving Average.

As the structure of the SMA parameters are already specified in StockChartX as it is a standard indicator, then the parameters need to be loaded directly – hence the next two lines for the indicator property string to ask for the Source for the data, and the number of periods for the SMA as an indicator parameter integer; which in this case is an integer (ShortSMA) that will have the number referred in it.

The next three lines are interesting as they specify the presentation of the indicator for the colour, the line width and to make the SMA indicator invisible at the moment. Note that when an indicator series is invisible, it plays no part in the specification of the vertical scale.