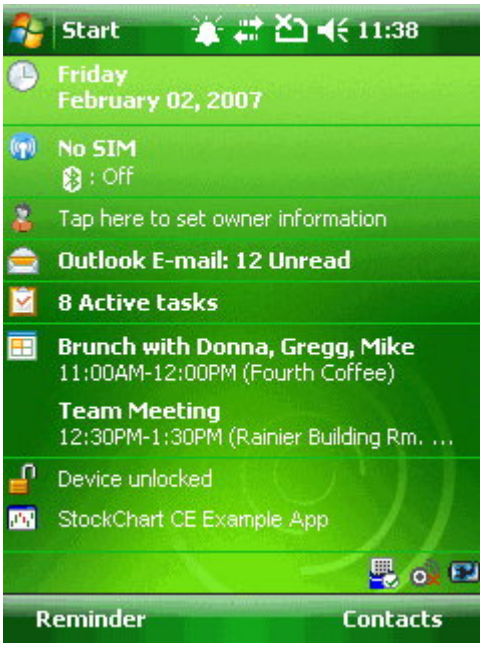
Service Disclaimer

Overview



StockChart CE was designed to take advantage of the latest features of Microsoft's newest Compact Edition of Windows.

StockChart CE targets all popular SmartPhones and PDAs that run on Windows Mobile 6.

Windows Mobile 6 allows users to check e-mail, keep track of their schedule and contacts, browse the Internet and manage business documents using mobile versions of Microsoft applications like Outlook, Office, and Windows Live™.

StockChart CE has been tested to work on all popular SmartPhones including the new Sprint MOGUL, T-Mobile Wing, T-Mobile DASH, MOTO Q 9m and many others…

Professional Design

Developing a mobile charting application is extremely challenging due to the compact nature of small handheld devices. The memory constraints and operating system feature limitations alone make developing a full-featured charting application extremely difficult and tedious.

We are proud to offer this exciting new edition of our financial charting products for the Windows Mobile 6 platform.

Features

StockChart CE features over 60 popular technical indicators that may be customized by the end user, complete navigational control (zooming and scrolling), extended line styles and color properties, user-drawn trend lines, advanced stylus events that return logical x and y values from the stylus, plus much more. StockChart CE is an extremely powerful and lightweight financial charting component for the Windows Mobile 6 platform.

Prerequisits

StockChart CE is a lightweight financial charting component designed for Windows Mobile 6.0 and later. This charting component is compatible with any SmartPhone or PDA running Windows Mobile 6 with a minimum of 1.8MB of free RAM. A working knowledge of VB.NET 2005 or later is required to use this product.

Introduction

StockChart CE is designed for Visual Studio 2005 and later. When you load the StockChart CE solution into Visual Studio, you will see four or six projects in the Solution Explorer, depending if you purchased the source code.

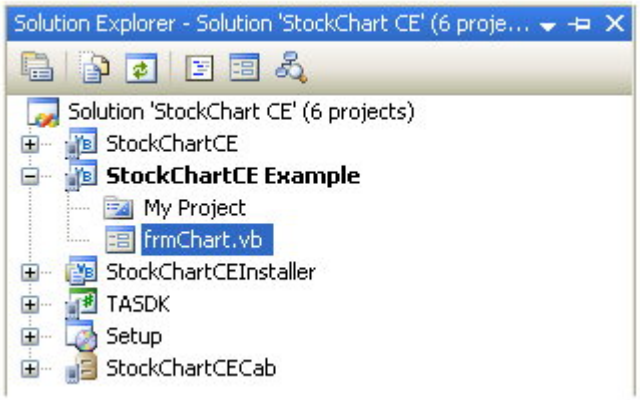
If you purchased the source code for StockChart CE, you will see two projects named StockChartCE and TASDK.

TASDK contains the source code for all technical indicators used by the charting component and is written in C#.

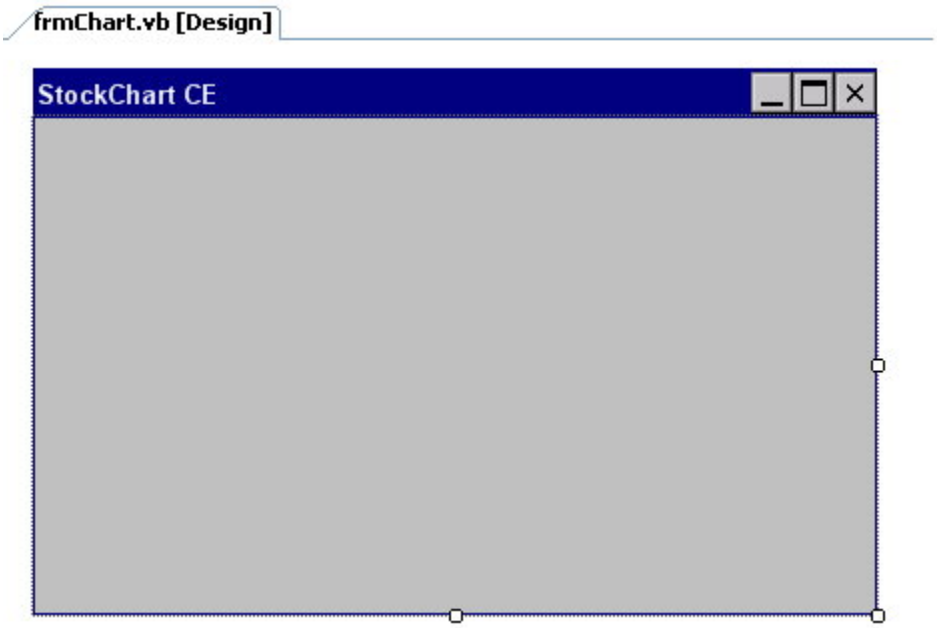
StockChartCE contains the actual stock chart source code for painting the charts plus property dialogs for chart color settings, indicator parameters, etc.

No matter if you have the StockChartCE source code or not, you will see the **StockChartCE Example** project in the solution explorer.

The StockChart CE Example project is a VB.NET project that is used as a container to draw the chart on a Windows CE form.



When you expand the **StockChart CE Example** project, you will see the frmChart.vb Windows CE form. This form is the container control for the StockChart CE library.



The form appears blank with no design time controls. This is because StockChart CE is created within the form's code in this example project. Optionally, you can place the control on a form at design time.

All supported features are shown in frmChart.vb

The code behind frmChart.vb shows every supported feature of StockChart CE, which is extremely simplified by design. The **LoadTestData** function of frmChart.vb shows commented source code for all features including setting up the chart, adding the license string, loading test data, setting bar properties, adding technical indicators and so forth.

```
Inherits System.Windows.Forms.Form

Private WithEvents Chart1 As New StockChartCE.Chart

'Loads test data
Private Sub LoadTestData()

    'Setup the control:
    Chart1.LicenseKey = "{0000000-0000-0000-0000-00000000000}" 'DO NOT DISTRIBUTE THIS KEY
    Me.Controls.Add(Chart1)
    Chart1.Dock = DockStyle.Fill
    Chart1.Visible = True

    'The symbol must be set before the control can be used:
    Chart1.Symbol = "MSFT"

    'Optionally, set the context menu:
    Chart1.ContextMenu = Me.ContextMenu1

    'Get the panel that will contain the candles:
    Dim p As StockChartCE.ChartPanel = Chart1.DefaultPanel()

    'Insert some the bar data:
    p.AddBar("9/8/2007", 23.25, 23.4, 21.49, 21.5, 84062500)
    p.AddBar("9/9/2007", 20.88, 23.18, 20.7, 23.11, 101409500)
    p.AddBar("9/10/2007", 22.73, 22.86, 21.22, 21.42, 81719500)
```

The StockChart CE Example project is the main project in this solution. Unless you need to modify the technical indicators or chart painting features, you will not need to edit the **StockChartCE** or **TASDK** projects (these are included with the source code license only).

Setup and Deployment
StockChart CE is provided as two DLL files: **StockChartCE.dll** and **TASDK.dll** These two files must be referenced from within your Windows Mobile 6 project and installed on the Windows Mobile 6 target device.

If you will be deploying your project over a network, you must sign your files using a signing company such as Verisign. Please note that a standard code-signing certificate is inadequate. A special certificate for Windows Mobile 6 must be purchased and is licensed per "signing event". Please contact your certificate authority for details.

The StockChart CE solution includes three setup projects.

The **StockChartCECab** project is a Windows CE cab builder. This project takes the output of **StockChartCE Example** and includes **StockChartCE.dll** plus **TASDK.dll** to create a single cab file, which is to be installed on the mobile target device.

Note that if you will be using a code-signing certificate as mentioned above, you may right click on the **StockChartCECab** project and choose **Properties**, then click the **Authenticode Signature** check box and load your certificate by clicking the **Browse** button. When you compile the **StockChartCECab** project, the cab file will be signed automatically. If you have any difficulties signing your project you must contact your code signing authority for technical support with your certificate, as Modulus cannot support code signing certificate technical support inquiries.

StockChartCEInstaller is a Windows executable that is run from the end user's host desktop machine that is used to install the Windows Mobile 6 application via Microsoft ActiveSync (the software used to communicate between the mobile device and Windows XP, Vista, etc.).

StockChartCEInstaller is designed to initiate the installation process by telling ActiveSync to copy the StockChart CE cab file from the ActiveSync directory and onto the mobile device.

Setup
The **Setup** project located in the Solution Explorer is designed to copy the output of the **StockChartCECab** project plus the output of the **StockChartCEInstaller** into the ActiveSync directory. **Setup** initiates the ActiveSync installation process via a "Custom Action", which may be found in the **Setup** project's **Custom Actions** folder. This Custom Action calls the **StockChartCEInstaller.dll** registration function, which in turn initiates the ActiveSync installation process. Please note that we do not provide technical support for modifying the setup projects or debugging problems associated with setups.

Solution Build Order

The StockChart CE solution must be built in a particular order because each project output is dependant upon another project output.

Always start by choosing “Clean Solution” from the Visual Studio .NET Build menu.

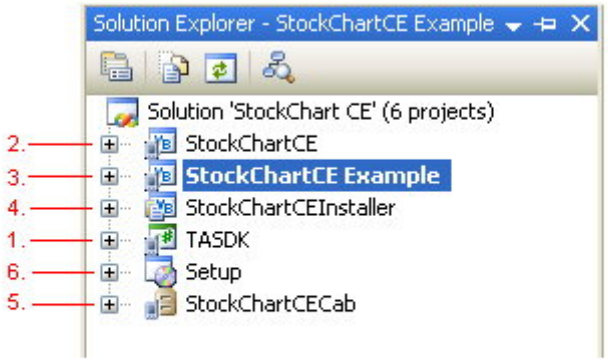
1. If you have purchased source code for StockChart CE, the first project that must be built is **TASDK**, followed by **StockChartCE**. This is required only if you have purchased source code for the charting component.

2. The next step is to build the **StockChartCE Example** project. This is the project that contains source code that will be deployed on the mobile device.

3. Next, build the **StockChartCECab** project. This project packages the executable output of the **StockChartCE Example** project and also packages the StockChartCE.dll and TASDK.dll files into a cab file.

4. Now build the **StockChartCEInstaller** project. This project copies the cab file from the ActiveSync directory onto the mobile device when initiated from the **Setup** project.

5. Finally, build the **Setup** project. The setup project is responsible for copying the cab file to the ActiveSync directory and initiating the mobile installation process.



Build Order

StockChart CE Library Documentation

The StockChart CE library is extremely simple by design, yet powerful and customizable. Only a handful of properties and functions are required to build full-featured stock, commodity and forex charts using historic or real-time market data.

Please note: This documentation is considered complete as of this writing. If a property or function is not mentioned herein, that particular property or function is unsupported and you should avoid using it. All supported features are shown in the example form.

 **Chart Class**

The Chart is a user control that can be dropped on any Windows CE form at design time, or created at run time. This is the only visual object that should be added to your form.

Chart Properties

 **Chart.LicenseKey As String**
The LicenseKey property is a 37-character GUID string that is provided on the Downloads & Updates page of our web site. The string may look similar to this:
`{1111111-2222-3333-4444-555555555555}`
You must set this property before showing the StockChart CE form, otherwise the chart will not paint!

 **Chart.Symbol As String**
This property represents the stock, futures or forex symbol and must contain a string between 1 and 50 characters in length. You must set this property before showing a chart.

 **Chart.ContextMenu As ContextMenu**
The ContextMenu property may be set to an instance of an existing ContextMenu control. This allows the application to display a context menu when the user clicks and holds the stylus over the chart. The context menu may allow the user to view chart properties, technical analysis, draw trend lines, etc. as shown in the example project. You must handle the events of the ContextMenu in your client application as shown in the example.

 **Chart.DefaultPanel As ChartPanel**
StockChart CE allows you to display one or more “panels” on a chart. A panel is a horizontal area that is separate from the rest of the chart. Panels are stacked on top of each other to form a chart. For example, the price panel (DefaultPanel) is always on top. This is where candles or bars are shown. The volume panel (VolumePanel) is always on bottom. This is where volume is shown. Panels may be added in-between the price and volume panels to show technical analysis. The DefaultPanel property returns the instance of the top price panel, which always exists and cannot be removed.

Chart.VolumePanel As ChartPanel
Returns the handle to the volume panel if it exists. The volume panel may be added via the AddVolumePanel method and may be removed by the RemoveVolumePanel method.

Chart.VolumePostFix As String
This string allows you to set the postfix letter of the volume. For example if volume is divided by millions, set the VolumePostFix property to “M” for millions, or “K” for thousands if it is divided by thousands, etc.

Chart.Panel(Name As String) As ChartPanel
Accepts a string identifier for a chart panel and returns the ChartPanel handle if found.

Chart.CurrentPanel As ChartPanel
Returns the instance to the most “current” panel, where the last known StylusMove event was sent from.

Chart.ChartPeriodicity As Periodicity
Sets the x-scale periodicity to real time or daily, which changes the format of the x-scale.

Chart.ChartBackcolor As Color
Sets or returns the back color of the chart (for all visible panels).

Chart.ScaleColor As Color
Sets or returns the x and y scale back color.

Chart.DownColor As Color
Sets or returns the down-candle color where the close is less than the open.

Chart.UpColor As Color
Sets or returns the up-candle color where the close is greater than the open.

Chart Methods

Chart.RemovePanel(Name As String)
Accepts a string identifier for a chart panel and removes the ChartPanel and all series associated with that panel if the panel is found.

Chart.AddVolumePanel
Adds the volume panel to the bottom of the chart.

Chart.RemoveVolumePanel
Removes the volume panel from the bottom of the chart.

Chart.AddChartPanel(Name As String) As ChartPanel
Adds an empty ChartPanel to the Chart and returns a handle to the newly added ChartPanel. You must add a technical indicator to this new panel before the chart is painted, otherwise the panel will be set to null.

Chart.ShowChartProperties
Displays the chart properties dialog box, which allows the user to modify chart colors.

Chart.ShowTechnicalAnalysis
Displays the technical analysis properties dialog box, which allows the user to add and remove technical indicators.

Chart.AddTechnicalIndicator(TheIndicator As Indicator.IndicatorType, IndicatorName As String, PanelName As String) As Indicator
This method adds a technical indicator to the specified chart panel and returns a handle to that new Indicator object.

Chart.ZoomIn
Removes one bar from the visible record count, thereby zooming in on the chart.

Chart.ZoomOut
Adds one bar to the visible record count, thereby zooming out of the chart.

Chart.ResetZoom
Resets the visible record count of the chart to equal the record count of all records, thereby zooming out 100%.

Chart.ScrollLeft
Scrolls the chart to the left by one bar. Note that the chart must be zoomed in by at least one bar for this to work.

Chart.ScrollRight
Scrolls the chart to the right by one bar. Note that the chart must be zoomed in by at least one bar for this to work.

Chart Events

OnStylusMove(ByVal Name As String, ByVal SeriesShortName As String, ByVal X As Single, ByVal Y As Single, ByVal TradeDate As Date, ByVal Value As Double)
Fires when the stylus is moved across any chart panel. Returns the panel Name, SeriesShortName for that panel (the short description of the series name, e.g. “SMA” for “Simple Moving Average”), the X and Y pixels, the TradeDate, and the logical Value.

OnStylusMouseDown(ByVal Name As String, ByVal SeriesShortName As String, ByVal X As Single, ByVal Y As Single, ByVal TradeDate As Date, ByVal Value As Double)
Fires when the stylus is pressed down on any chart panel. Returns the panel Name, SeriesShortName for that panel (the short description of the series name, e.g. “SMA” for “Simple Moving Average”), the X and Y pixels, the TradeDate, and the logical Value.

OnStylusMouseUp(ByVal Name As String, ByVal SeriesShortName As String, ByVal X As Single, ByVal Y As Single, ByVal TradeDate As Date, ByVal Value As Double)
Fires when the stylus is released from any chart panel. Returns the panel Name, SeriesShortName for that panel (the short description of the series name, e.g. “SMA” for “Simple Moving Average”), the X and Y pixels, the TradeDate, and the logical Value.

ControlBoxVisibleChanged(ByVal Visible As Boolean)
Fires whenever the control box is shown or hidden. The user may show the control box by moving the stylus across the top of the chart, or hide the control box by clicking lower down on the chart.

ControlEnabled(ByVal Type As ChartPanel.ControlType, ByVal Enabled As Boolean)
Fires whenever a control is enabled or disabled on the control box.

ControlFocus(ByVal Type As ChartPanel.ControlType)
Fires whenever a control receives focus in the control box.

ChartPanel Class

A ChartPanel is a container for price data, technical indicators and line drawings. One ChartPanel is loaded by default for price data (Chart.DefaultPanel). Other panels may be added or removed automatically by the Chart object after the user makes technical analysis selections via the technical analysis property dialog.

Important Note: Never create an instance of this class yourself. The Chart object is responsible for the creation and destruction of ChartPanel objects.

The ChartPanel class manages data within the BarData and SeriesData classes:

ChartPanel.SeriesData – stores series data.

Value As Double
Properties As BarProperties

ChartPanel.BarData – stores bar data and extended series data.

TradeDate As Date
OpenPrice As Double
HighPrice As Double
LowPrice As Double
ClosePrice As Double
Volume As Long
Properties As BarProperties
Series As List(Of SeriesData)

Note that each bar may have an infinite number of series (i.e. indicator values).

ChartPanel.BarProperties – stores visual property settings.

Name As String
Color As Color
Style As Drawing2D.DashStyle
Type As SeriesType

ChartPanel.SeriesType – visual properties for bar styles.

CandleChart = 0
StockChart = 1
LineSeries = 2
Histogram = 3

ChartPanel.LineStudyType – line studies.

TrendLine = 0

ChartPanel.NullValue = -987654321

ChartPanel.AddBar
Adds a bar to the panel. A bar consists of a trade date, open, high, low, close, volume and optionally, extra series data such as custom technical indicators. Several unique versions of AddBar are available. Technical indicators will be recalculated after calling AddBar.

ChartPanel.RemoveBar
Removes a bar and its associated extra series from the chart panel.

IMPORTANT NOTE: When you call AddBar or RemoveBar, YOU MUST ENSURE THAT ALL SERIES ARE THE SAME LENGTH before the chart paints. If one series is shorter than the other series, you must insert the NullValue constant to make the series the same length, and the chart will avoid painting in this area.

ChartPanel.EditBar
Edits a bar and its associated series.

ChartPanel.GetBar(TradeDate As Date Or Index As Integer) As BarData
Returns a BarData instance based on the Trade Date or Index.

ChartPanel.GetBarData() As List(Of BarData)
Returns a list of all BarData instances and their associated extra series in the chart panel.

ChartPanel.BarCount As Integer
Returns the number of records in the chart panel.

ChartPanel.DrawLineStudy(LineStudy As ChartPanel.LineStudyType, Key As String)
Sets the ChartPanel in a state that allows the user to draw a line study using the stylus. The LineStudyCount property will be incremented by one after the user has finished drawing the line study.

ChartPanel.RemoveLineStudy(Key As String)
Finds a line study based on the supplied key and removes it from the chart panel.

ChartPanel.ClearLineStudies()
Removes all line studies from the chart panel.

ChartPanel.LineStudyCount() As Integer
Returns the number of line studies in the chart panel.

ChartPanel.LineStudyType(Key As String) As ChartPanel.LineStudyType
Sets or returns the line study type based on the specified line study key.

ChartPanel.LineStudyColor(Key As String) As Color
Sets or returns the line study color based on the specified line study key.

ChartPanel.RemoveIndicator(Key As String)
Finds a technical indicator based on the specified key and removes it from the chart panel if it is found.

ChartPanel.Indicators As List(Of Indicators)
Returns a list of technical indicators loaded in the chart panel.

Indicator Class

Indicator.Color1 As Color
Sets or returns the color for the first series in the indicator list. Some indicators return more than one series, e.g. MACD, Aroon, etc.

Indicator.Color2 As Color
Sets or returns the color for the second series in the indicator list. Some indicators return more than one series, e.g. MACD, Aroon, etc.

Indicator.Color3 As Color
Sets or returns the color for the third series in the indicator list. Some indicators return more than one series, e.g. MACD, Aroon, etc.

Indicator.IsOverlay
Returns True if the indicator is an overlay series. Overlay series consist of bands, moving averages, etc. which are overlaid on the price panel (DefaultPanel).

Indicator.FriendlyName
Returns the "friendly" name for an indicator e.g. for indSimpleMovingAverage, FriendlyName returns "Simple Moving Average".

Indicator.ShortName
Returns a short name for an indicator, less than 10 characters in length e.g. Simple Moving Average = "SMA", Chande Moment Oscillator = "CMO", etc.

`Indicator.IndicatorType - list of technical indicators.`

```
indSimpleMovingAverage = 0
indExponentialMovingAverage = 1
indTimeSeriesMovingAverage = 2
indTriangularMovingAverage = 3
indVariableMovingAverage = 4
indVIDYA = 5
indWellesWilderSmoothing = 6
indWeightedMovingAverage = 7
indWilliamsPctR = 8
indWilliamsAccumulationDistribution = 9
indVolumeOscillator = 10
```

```
indVerticalHorizontalFilter = 11
indUltimateOscillator = 12
indTrueRange = 12
indTRIX = 14
indRainbowOscillator = 15
indPriceOscillator = 16
indParabolicSAR = 17
indMomentumOscillator = 18
indMACD = 19
indEaseOfMovement = 20
indDirectionalMovementSystem = 21
indDetrendedPriceOscillator = 22
indChandeMomentumOscillator = 23
indChaikinVolatility = 24
indAroon = 25
indAroonOscillator = 26
indLinearRegressionRSquared = 27
indLinearRegressionForecast = 28
indLinearRegressionSlope = 29
indLinearRegressionIntercept = 30
indPriceVolumeTrend = 31
indPerformanceIndex = 32
indCommodityChannelIndex = 33
indChaikinMoneyFlow = 34
indWeightedClose = 35
indVolumeROC = 36
indTypicalPrice = 37
indStandardDeviation = 38
indPriceROC = 39
indMedian = 40
indHighMinusLow = 41
indBollingerBands = 42
indFractalChaosBands = 43
indHighLowBands = 44
indMovingAverageEnvelope = 45
indSwingIndex = 46
indAccumulativeSwingIndex = 47
indComparativeRelativeStrength = 48
indMassIndex = 49
indMoneyFlowIndex = 50
indNegativeVolumeIndex = 51
indOnBalanceVolume = 52
indPositiveVolumeIndex = 53
indRelativeStrengthIndex = 54
indTradeVolumeIndex = 55
indStochasticOscillator = 56
indStochasticMomentumIndex = 57
indFractalChaosOscillator = 58
indPrimeNumberOscillator = 59
indPrimeNumberBands = 60
indHistoricalVolatility = 61
indMACDHistogram = 62
```