BERGtools help file

You can use these in VB.NET, C Sharp.NET and C++.NET.

To use these tools, after you have copied the DLL onto your system, you will need to add a reference to your project. Add the reference by right clicking on Reference >> Add Reference… A dialog box will show, select the Browse tab, browse to where you installed the software and select the BERGtools.DLL.

Thermometer:

This object shows the look and feel of a glass thermometer. It will adjust itself to whatever size and shape you make of the associated picture box.

Place a picture box on your form in the rectangular shape you want the thermometer to have.

Place this syntax in the Public class of your form or similar location

```
Private TempMeter As BergTools.Thermometer
```

Where TempMeter is the name of the object you will call throughout the program.
you can instantiate this as an array if you wish

Here is the initialization routine you would put in the form load of your application:

```
Object = New Thermometer()
```

Instantiate the thermometer as TempMeter
Example: `TempMeter = New Thermometer()`

Initialize:

```
TempMeter.Initialize(PictureboxName)
```

select the picturebox to use as a thermometer,
Default settings are:

```
BackgroundColor = Cornsilk
TemperatureColor = Red
FontColor = Black
ScaleColor = Black
```

`YScale` to minimum = 0, maximum = 100, and ticks = 10.
Example: `TempMeter.Initialize(pbTMeter)`

TemperatureColor:

```
TempMeter.TemperatureColor(System.Drawing.Color)
```

Select the color of the temperature data , use any system drawing color.
Example: `TempMeter.TemperatureColor(Color.Red)`

FontColor:

```
TempMeter.FontColor(System.Drawing.Color)
```

Select the color of the text, use any system drawing color.
Example: `TempMeter.FontColor(Color.Black)`

BackgroundColor

`TempMeter.BackgroundColor(System.Drawing.Color)`
Set the background color of the meter, use any system drawing color.
`Dim mycolor As System.Drawing.Color = Me.BackColor`
this line reads the form color (for transparent to form)
Example: `TempMeter.BackgroundColor(System.Drawing.Color)`

ScaleColor:
`TempMeter.ScaleColor(System.Drawing.Color)`
select scale and tick mark color, use any system drawing color.
Example: `TempMeter.ScaleColor(Color.Black)`

YScale:
`TempMeter.YScale(min, max, ticks)`
enter the min, max, and number of ticks for the Y Scale.

min
Minimum Scale value.
max
Maximim scale value.
ticks
Number of tick marks to show
Example: `TempMeter.YScale(0, 100, 10)`

DrawBackground:
`TempMeter.DrawBackground()`
Draw the generated bitmap into the picture box as background, only shows when the app is running.

updateTMeter:
`TempMeter.updateTMeter(NewTemperature)`

Call this function when you want to send a new value to the thermometer

NewTemperature
Data value to send to the thermometer. It can be Single, Double, Decimal, Integer or Long.
Example: `TempMeter.updateTMeter(32)`

Analog Meter

This object offers the look and feel of an analog meter on your form. It can be set to uni-polar or bi-polar scaling, and take whatever size you make the related picture box.

Place a picture box on your form in the rectangular shape you want the analog meter to have.

Place this syntax in the Public class of your form or similar location

```
Private Ameter As BergTools.AnalogMeter
```

Where AMeter is the name of the object you will call throughout the program.
You can instantiate this as an array if you wish:
Example: `Private AMeter(1) As AnalogMeter`

Here is the initialization routine you would put in the form load of your application:

```
Object = New Thermometer()
```

Instantiate the AnalogMeter
Example: `AMeter = New AnalogMeter()`

Initialize:
`AMeter.Initialize(PictureboxName)`
Select the picturebox to use as analog Meter
Default settings are:
SelectPointerColor = Blue
SelectScaleColor = Blue
MeterBackgroundColor = Beige
SelectFontColor = Blue
Scale to minimum = -5, maximum = 5, and ticks = 6.
Example: `AMeter.Initialize(pbMeter1)`

SelectPointerColor:
`AMeter.SelectPointerColor(System.Drawing.Color)`
Select the color of the data pointer, use any system drawing color.
Example: `AMeter.SelectPointerColor(Color.Black)`

SelectScaleColor:
`AMeter.SelectScaleColor(System.Drawing.Color)`
Select the scale and tick color, use any system drawing color.
Example: `AMeter.SelectScaleColor(Color.Crimson)`

MeterBackgroundColor:
`AMeter.MeterBackgroundColor(System.Drawing.Color)`
Select the meter background color, use any system drawing color.
Example: `AMeter.MeterBackgroundColor(Color.Beige)`

SelectFontColor:
`AMeter.SelectFontColor(System.Drawing.Color)`
Select the color of the text, use any system drawing color.

Example: `AMeter.SelectFontColor(Color.Chocolate)`

Scale:
`AMeter.Scale(Min, Max, Ticks)`
Enter the min, max, and number of ticks for the Scale. You can have a unipolar or bipolar scale.

min
Minimum Scale value.
max
Maximim scale value.
ticks
Number of tick marks to show.
Example: `AMeter.Scale(0, 100, 3)`

DrawBackground:
`AMeter.DrawBackground()`
Draw the generated bitmap into the picture box as background, only shows when the app is running.

UpdateValue:
`AMeter.UpdateValue(NewDatapoint)`
Call this function when you want to send a new value to the meter.

NewDatapoint
NewDatapoint is the new data to add to meter, as a single precision number.
Example: `AMeter.UpdateValue(8.235)`

StripChart

This Stripchart will plot 1 channel of continuously updating data, 1 point at a time. The graph is based on the Cartesian coordinate plane, quadrant 1, or quadrants 1 and 4. The X Axis can be set from a Timer's Interval property.

Place a picture box on your form in the rectangular shape you want the stripchart to have.

Place this syntax in the Public class of your form or similar location

```
Private Charter As BergTools.StripChart
```

Where AMeter is the name of the object you will call throughout the program.
You can instantiate this as an array if you wish:
Example: `Private Charter(1) As StripChart`

Here is the initialization routine you would put in the form load of your application:

```
Object = New StripChart()
```

Instantiate the Stripchart
Example: `Charter = New StripChart()`

Initialize:

```
Charter.Initialize(Picturebox)
```

Select the picturebox to use as stripchart
Default settings are:

SelectPenColor to Blue,
BackgroundColor to Beige,
FontColor to dark gray,
ScaleColor to black,
YStartPoint to Bottom
XScale = Interval = 100 and Ticks = 5)
EnableAutoscale = True
YScale to minimum = 0, maximum = 100, and ticks = 10.

Example: `Charter.Initialize(pbStripChart1)`

SelectPenColor:

```
Charter.SelectPenColor(System.Drawing.Color)
```

Select the color of the data trace line, use any system drawing color.
Example: `Charter.SelectPenColor(Color.Blue)`

FontColor:

```
scope.FontColor(System.Drawing.Color)
```

Select the color of the text, use any system drawing color.
Example: `scope.FontColor(Color.Beige)`

BackgroundColor:

```
scope.BackgroundColor(System.Drawing.Color)
```

Select the background color, use any system drawing color.
Example: `scope.BackgroundColor(Color.DarkGreen)`

ScaleColor:
`scope.ScaleColor(System.Drawing.Color)`
Select scale and tick mark color, use any system drawing color.
Example: `scope.ScaleColor(Color.DarkKhaki)`

YStartPoint:
`Charter.YStartPoint(BERGtools.YStartingPoint)`
Select the Y axis start reference point for data that is unipolar or bipolar.
Bottom or unipolar, the Y Axis origin is at the bottom left corner of the graph.
Middle or bipolar, the origin of the Y axis is in the middle of the graph along the left side.
Example: `Charter.YStartPoint(BERGtools.YStartingPoint.Bottom)`

YScale:
`Charter.YScale(Min, Max, Ticks)`
Enter the min, max, and number of ticks for the Y Scale.

min
Minimum Scale value.
max
Maximim scale value.
ticks
Number of tick marks to show.
Example: `Charter.YScale(0, 200, 3)`

XScale:
`Charter.XScale(timetick, ticks)`
enter the timer interval and number of ticks for X scale.
timetick
The timer interval used to calculate the values of the X axis ticks. This can be best set to the value of the timer objects Interval property.
ticks
Number of tick marks to show.
Example: `Charter.XScale(tmrAtoD.Interval, 5)`

EnableAutoscale:
`Charter.EnableAutoscale(Boolean)`
enable/disable autoscaling of y axis.
True enables Y axis autoscaling
False disables Y axis autoscaling
Example: `Charter.EnableAutoscale(True)`

DrawBackground:
`Charter.DrawBackground()`
Draw the generated bitmap into the picture box as background, only shows when the app is running.

DrawStripChart:
```
Charter.DrawStripChart(ByRef NewDatapoint As Double)
```
Call this method when you want to send a new value to the stripchart.

NewDatapoint

NewDataPoint is the new data to add to meter, as a double precision number.

Example: `Charter.DrawStripChart(8.235)`

Reset:

`charter.Reset()`

To reset and clear the stripchart.

Oscilloscope

This oscilloscope will plot up to 8 channels of data on a Cartesian coordinate plane.

Place a picture box on your form in the rectangular shape you want the oscilloscope to have.

Place this syntax in the Public class of your form or similar location

```
Private scope As BergTools.Oscilloscope
```

Where AMeter is the name of the object you will call throughout the program.
You can instantiate this as an array if you wish:
Example: `Private scope(1) As Oscilloscope`

Here is the initialization routine you would put in the form load of your application:

```
Object = New Oscilloscope()
```

Instantiate the oscilloscope
Example: `scope = New Oscilloscope()`

Initialize:
`scope.Initialize(Picturebox)`
Select the picturebox to use as oscilloscope, and initialize all settings to defaults.
Default settings are:

All TraceColors (0-7) to white,
BackgroundColor to Blue,
FontColor to Gray,
ScaleColor to white,
YAxisScale to bipolar
YScale to minimum = -10, maximum = 10, and ticks = 0.

Example: `scope.Initialize(pbScope)`

TraceColor:
`scope.TraceColor(TraceNumber, System.Drawing.Color)`
Select the color of the data trace for a channel,
TraceNumber

The number or index of the trace to configure. An integer value between 0 and 7.

color

Select the color of the data trace line, use any system drawing color.

Example: `scope.TraceColor(0,Color.White)`

FontColor:
`scope.FontColor(System.Drawing.Color)`
Select the color of the text, use any system drawing color.
Example: `scope.FontColor(Color.Beige)`

BackgroundColor:
`scope.BackgroundColor(System.Drawing.Color)`
Select the background color, use any system drawing color.
Example: `scope.BackgroundColor(Color.DarkGreen)`

ScaleColor:

```
scope.ScaleColor(System.Drawing.Color)
```

Select scale and tick mark color, use any system drawing color.

Example: `scope.ScaleColor(Color.DarkKhaki)`

YAxisScale:

```
scope.YAxisScale(BERGtools.YAxisScaleVal)
```

Select the Y axis start reference point for data that is unipolar or bipolar.

Unipolar, the Y Axis shows only positive values.

Bipolar, the Y axis shows a range of negative to positive values.

Example: `scope.YAxisScale(BERGtools.YAxisScaleVal.Bipolar)`

YScale:

```
scope.YScale(Min, Max, Ticks)
```

Enter the min, max, and number of ticks for the Y Scale.

min

Minimum Scale value.

max

Maximim scale value.

ticks

Number of tick marks to show.

Example: `scope.YScale(-10, 10, 8)`

DrawBackground:

```
scope.DrawBackground()
```

Draw the generated bitmap into the picture box as background, only shows when the app is running.

UpdateScope:

```
Scope.updateScope(ByRef AddedDataArray(,) As Double)
Scope.updateScope(ByRef AddedDataArray(,) As Single)
Scope.updateScope(ByRef AddedDataArray(,) As Integer)
Scope.updateScope(ByRef AddedDataArray(,) As Long)
```

call this function when you want to send a new array of data. The array must be a two dimensional array. This function is overloaded to accept arrays of Singles, Doubles, Integers or Longs.

Example:

```
Dim twoDData(2,49) As Double
scope.updateScope(twoDData)
```

Function Generator

Place this syntax in the Public class of your form or similar location

```
Private FuncGen As BergTools.FunctionGenerator
```

Here is the initialization routine you would put in the form load of your application:

```
Object = New FunctionGenerator
```

Instantiate the AnalogMeter as AMeter

Example: `FuncGen = New FunctionGenerator`

Initialize:

```
FuncGen.Initialize(ByVal NumChannels As Integer)
```

Enter number of waveform channel. Recommended to not exceed the number of analog outputs on your device.

Example: `FuncGen.Initialize(1)`

WaveSelect:

```
FuncGen.WaveSelect(ByVal BERGtools.WaveType, ByVal CurrentChannel As Integer)
```

Select a predetermined wavetype.

MyWavetype

The waveform pattern to output. Choices are:
Square, Ramp, Noise, Triangle, Sine

CurrentChannel

The indexed number of the channel you want to select the waveform type. Not to exceed the number of channels selected above.

If there are multiple channels of varying numbers of periods, the function generator will calculate the least common multiple of waveforms needed so the data is syncronous.

Example: `FuncGen.WaveSelect(intCurrentChannel, BERGtools.WaveType.Sine)`

PointsPerPeriod:

```
FuncGen.PointsPerPeriod(PointsPerWavePeriod As Integer)
FuncGen.PointsPerPeriod(PointsPerWavePeriod As Integer, ChannelNumber As
Integer)
```

Number of points per period of waveform

PointsPerWavePeriod

The number of points that make up one period of the waveform.

ChannelNumber

The indexed number of the channel you want to select the waveform type. Not to exceed the number of channels selected above. This parameter is options if only using FunctionGenerator as 1 channel, else it is required.

Example: `FuncGen.PointsPerPeriod(360, 0)`

Amplitude:

```
FuncGen.amplitude(ByVal WaveAmplitude As Double)
FuncGen.amplitude(ByVal WaveAmplitude As Double, ByVal ChannelNumber As
Integer)
```

Peak voltage amplitude of waveform

WaveAmplitude

The number of points that make up one period of the waveform.

ChannelNumber

The indexed number of the channel you want to select the waveform type. Not to exceed the number of channels selected above. This parameter is options if only using FunctionGenerator as 1 channel, else it is required.

Example: `FuncGen.amplitude(2.9, 0)`

Offset:

```
FuncGen.offset(ByVal waveoffset As Double)
FuncGen.offset(ByVal waveoffset As Double, ByVal ChannelNumber As Integer)
```

Voltage offset of waveform

Offset

The DC offset of the waveform.

ChannelNumber

The indexed number of the channel you want to select the waveform type. Not to exceed the number of channels selected above. This parameter is options if only using FunctionGenerator as 1 channel, else it is required.

Example: `FuncGen.offset(0.5, 0)`

DutyCycle:

```
FuncGen.DutyCycle(ByVal WaveDutyCycle)
FuncGen.DutyCycle(ByVal WaveDutyCycle As Double, ByVal ChannelNumber As
Integer)
```

Duty Cycle - only for Squarewave, in percent, such as, 50 for 50% duty cycle.

Example: `FuncGen.DutyCycle (35, 0)`

CreateWave:

```
FuncGen.CreateWave()as single(,)
```

Call this function to create a new array of data

Output is an array of new data created by the function generator structured as a 2D array of Singles.

Example:

```
Dim AOScanData(,) As Single
AOScanData = FuncGen.CreateWave()
```

PID Control

Place this syntax in the Public class of your form or similar location

```
Private PID As BergTools.PIDControl
```

you can instantiate this as an array if you wish

```
Private PID(1) As PIDControl
```

Here is the initialization routine you would put in the form load of your application:

```
Object = New PIDControl
```

Instantiate the PIDControl

Example: `PID = New PIDControl()`

Proportional:

```
PID.Proportional(ByVal MyProportional As Double)
```

Set the Proportional value(gain)

Example: `PID.Proportional(5)`

Integral:

```
PID.Integral(ByVal MyIntegral As Double)
```

Set the Integral value (Reset)

Example: `PID.Integral(10)`

Derivative:

```
PID.Derivative(ByVal MyDerivative As Double)
```

Set the Derivative value

Example: `PID.Derivative(0)`

SetPoint:

```
PID.SetPoint(ByVal MySetPoint As Double)
```

Set the SetPoint value note that this can be set and reset from anywhere.

Example: `PID.SetPoint(80)`

PIDControl:

```
PIDControl(ByVal ProcessVariable As Double) As Double
```

Call this function when you want to add a new data point (from a timer event most likely).

Process variable

The new data to add to the pID calculation

OUTP

The newly calculated output from the algorythm.

Example: `OutP = PID.PIDControl(ProcessVariable)`

FFT

The FFT requires an array of either single or double precision data points. The array must be sized base in a power of 2, ie 512, 1024, 2048 for example. Should an array not of these lengths be provided, the algorythm will adjust the provided array to conform by truncating it to the desired size.

Place this syntax in the Public class of your form or similar location

```
Private FFT As BergTools.FFT
```

you can instantiate this as an array if you wish

```
Private FFT(1) As BergTools.FFT
```

Here is the initialization routine you would put in the form load of your application:

```
Object = New BergTools.FFT
```

Instantiate the FFT object

Example: `FFT = New BergTools.FFT()`

Initialize:

```
FFT.Initialize()
```

Call this function to initialize the FFT.
Default settings are:

SelectTransform(Transform.PowerSpectrum)

Example: `FFT.Initialize()`

SelectTransform:

```
FFT.SelectTransform(BERGtools.YStartingPoint)
```

Select the type of transform you want to perform, choices are:

Real
Imaginary
Magnitude
Phase
PowerSpectrum

Example: `Charter.YStartPoint(BERGtools.Transform.PowerSpectrum)`

SamplingRate:

```
FFT.SamplingRate(ByVal Rate As Integer)
Enter the speed at which the data was collected.  To achieve the best
accuracy of frequency, it is best to use a sampling rate that is a power of 2
such as 102400 instead of 100000 for example.
```

Example: `FFT.SamplingRate(102400)`

FFT:

```
FFT.FFT(ByRef FFTData(,) As Single) As Single(,)
FFT.FFT(byref FFTData(,) As Double) As Double(,)
```

Performs the Fast Fourier Transform of the type requested, on an array of provided Single or Double precision data, and returns an array, one half the original size of the same data type.

Example: `fftdata = FFT.FFT(ADDataV)`

FindMaxFreq:

```
FFT.FindMaxFreq() As Single
```

Returns the max frequency of the FFT performed data. The frequency is really the array element of the maximum data point, regardless of FFT type performed.

Example: `Label3.Text = FFT.FindMaxFreq.ToString`

IncludeDCContent:

```
IncludeDCContent(ByVal DC As Boolean)
```

Specify if you want to include the DC content of the original waveform in the FFT. Enter True for yes, and False for no. If you do want it included, then the DC content will be located in the 0 element of the output array. Also the FindMaxFreq function will not work correctly.

Example: `FFT.IncludeDCContent(False)`

DCContent:

```
DCContent() As Single
```

Returns the DC content of the original waveform or array.

Example: `Label4.Text = "DC Content = " + FFT.DCContent.ToString`