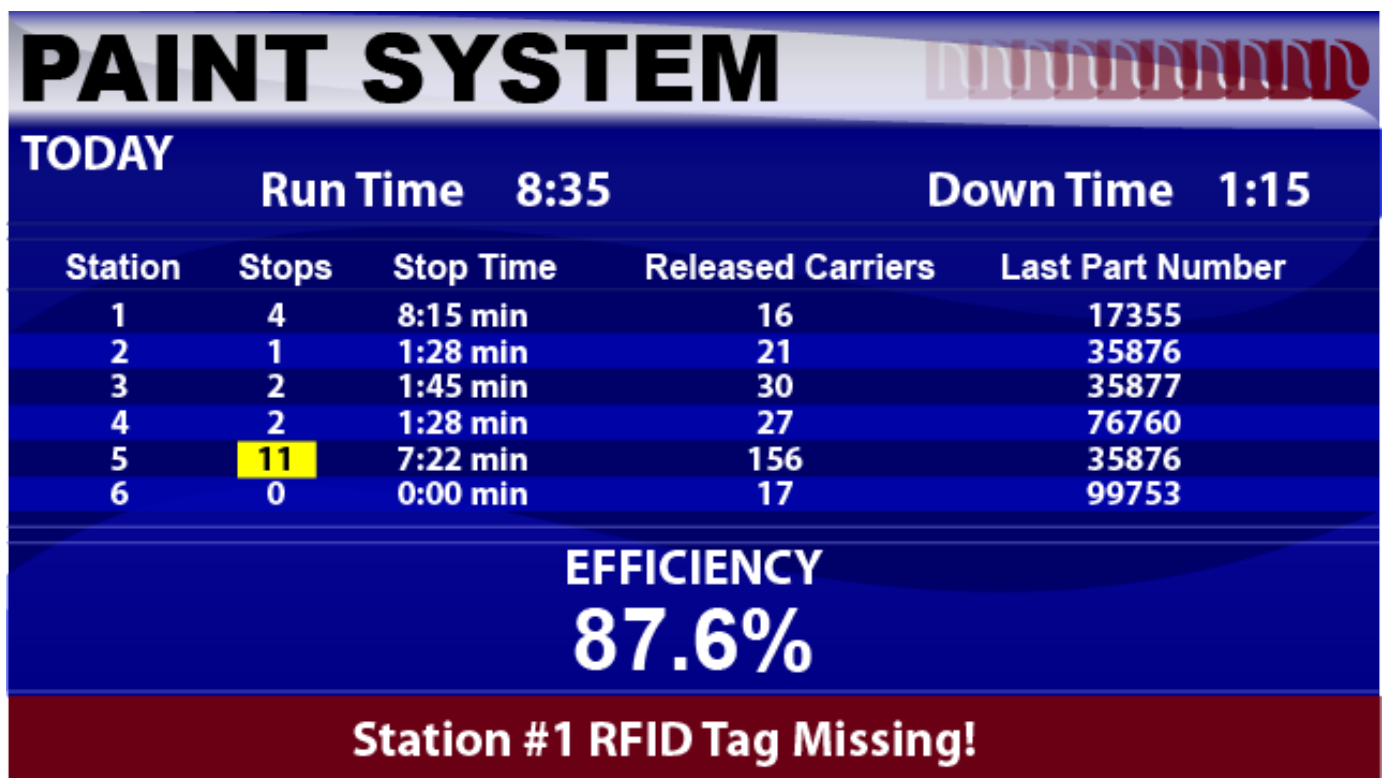


# Basic Label



*User Manual*

## Introduction

Although referred to as a Basic control, the BasicLabel is one of the most flexible and powerful components in AdvancedHMI's toolbox. This manual will guide you through the many properties and also give some tips and tricks that will let you get the most out of the BasicLabel.

The screen shot on the previous page shows an example of an HMI screen developed with only a background image and BasicLabels. With a full understanding and some creativity, the BasicLabel can be used to make very nice and useful HMI screens.

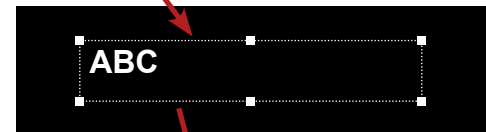
## Autosize and AutoEllipsis Properties



By default the Autosize property is set to True causing the size of the BasicLabel to change based on the content of the Text Property. Changing this to False will change many characteristics of the BasicLabel as we will demonstrate.

Properties	
AutoEllipsis	False
AutoSize	False
Text	ABC
TextAlign	TopLeft

When Autosize is set to False, the width and height of the BasicLabel stays the size set in design mode.



Now let's see what happens when there is more text than can fit into the BasicLabel.

Properties	
AutoEllipsis	False
AutoSize	False
Text	ABCDEFGHIJKLMNOPQRSTUVWXYZ
TextAlign	TopLeft



The Text property was changed to the complete alphabet, but only through "O" is shown. This is because the BasicLabel size is too small to display all of the text.

The AutoEllipsis property can be used to give an indication the text has been truncated.

Properties	
AutoEllipsis	True
AutoSize	False
Text	ABCDEFGHIJKLMNOPQRSTUVWXYZ
TextAlign	TopLeft

With Autosize set to False and Autoellipsis set to True, and the size of the BasicLabel is too small to display all of the text, an ellipsis (3 dots) will be automatically added to give an indication that all of the text cannot fit in the BasicLabel.



Now let's see how things react when we increase the vertical size of the label.

Properties	
AutoEllipsis	True
AutoSize	False
Size	190,40
Text	ABCDEFGHIJKLMNOPQRSTUVWXYZ
TextAlign	TopLeft

Increasing the height of the BasicLabel to give enough room for 2 lines of text to fit allows all of the contents of the Text property to be displayed and ellipsis is no longer added.

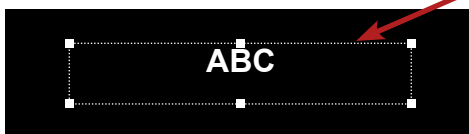


## TextAlign Property

In the previous samples the TextAlign property was left to default of TopLeft. With the AutoSize property set to False, the TextAlign is used to position the contents of Text within the size of the BasicLabel bounds.



Properties	
AutoEllipsis	False
AutoSize	False
Text	ABC
TextAlign	TopCenter



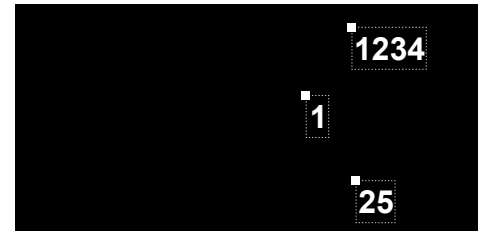
TextAlign has been set to TopCenter in this example.

## Tips & Tricks for Autosize & TextAlign

### Alignment

A well layed out form is important in giving your application a professionally done appearance. Making use of the AutoSize, TextAlign, Location, and Size properties help ensure a nice even layout. GroupBoxes are also a nice tool for grouping related items.

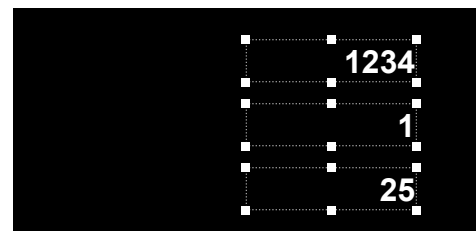
Let's start by adding 3 BasicLabels to the form in random locations. Also put in some numerical values of different lengths. Set the properties to the values shown below.



Properties	
AutoSize	False
Location	200, ??
Size	190,40
TextAlign	MiddleRight

Set the X coordinates of all 3 BasicLabels to the same value. The Y coordinates should increment by the same. For example, the 3 BasicLabels Y values can be 100, 150, and 200.

We can now see the 3 BasicLabels are all nicely right justified no matter how many characters in the box. The width will have to be adjusted to accomodate the longest text expected.

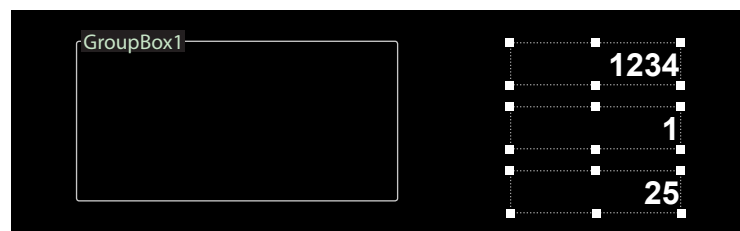


*Note : Visual Studio has a tool for helping to align and space controls. Try selecting all 3 of the BasicLabels, then go to the Format menu and select Vertical Spacing->Make Equal*

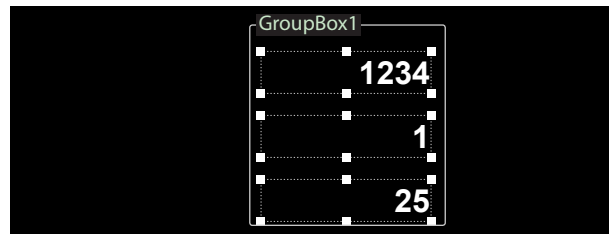
### GroupBox

We now have 3 BasicLabels that are nicely aligned, but they still have an appearance of unrelated values floating on the form. This can be improved upon by using a GroupBox. Let's now add a GroupBox to the form.

*Note : By default the ForeColor property (which is the text color) of the GroupBox is black, so you will need to change this in order to see the text.*



Increase the size of the GroupBox so all 3 BasicLabels can fit into the box. Now select all 3 BasicLabels simultaneously and drag them into the GroupBox. Be sure to drag all 3 simultaneously in order to preserve the spacing and alignment.

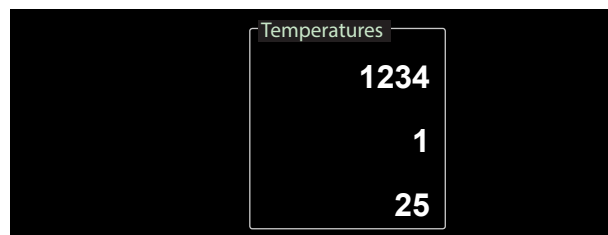


*Tip : In Visual Studio multiple items can be selected by either drawing a window around them or clicking one at a time while holding down the Shift key.*

### GroupBox1

Properties	
ForeColor	<input type="checkbox"/> White
Text	Temperatures

A few settings to the properties of the GroupBox give us our nice refined group of numbers.

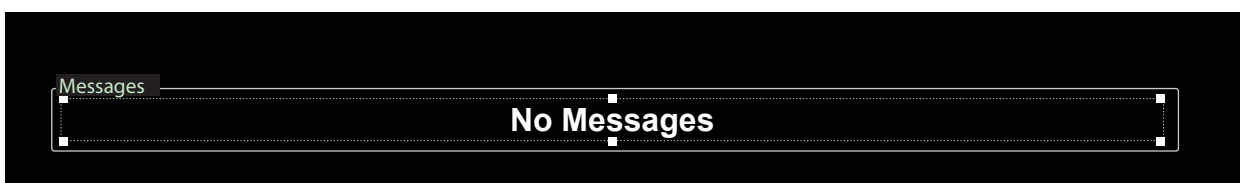


Another common use for the GroupBox is a message display bar. We will add a group box that is very wide, but only slightly taller than our text font we will use. We then add a BasicLabel in the GroupBox, set AutoSize to False, and make it the full width of the GroupBox. Finally set the TextAlign to MiddleCenter.

### BasicLabel1

Properties	
AutoSize	False
Location	10,20
Size	600,40
TextAlign	MiddleRight

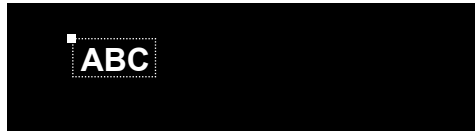
Notice the coordinates of the location is relative to its container. Which in this case the BasicLabel's container is the GroupBox.



The final result is a center justified message with a border and title.

## HighlightColor and HighlightKeyCharacter Properties

These properties can be used to make the background of the label change colors based on the contents of the Text property. For example, if you were using the BasicLabel to display alarms kept in a string in the PLC, the background color can change if a character is found in the string that matches the HighlightKeyCharacter.



Properties	
HighlightColor	<span style="color: red;">■</span> Red
HighlightKeyCharacter	!
Text	ABC

Leaving the property values to default and entering ABC into the Text property, we see nothing significant occurs.

If an exclamation mark is added to the Text property, it triggers the highlight function. We see the effect is the changing of the Background color to red (HighlightColor)

Properties	
HighlightColor	<span style="color: red;">■</span> Red
HighlightKeyCharacter	!
Text	ABC!



## NumericFormat and ScaleFactor Properties

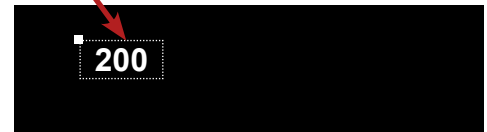
These properties play a role in the display of numeric values only.



By default the ScaleFactor property is set to 1. The scale factor is used to scale a value from the PLC before displaying. The value from the PLC is multiplied by the scale factor prior to being displayed.

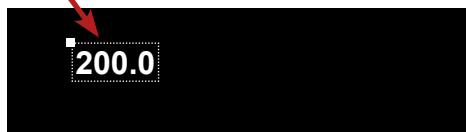
Properties	
NumericFormat	False
ScaleFactor	2
Text	100

By setting the ScaleFactor to 2, the value that goes into the Text property gets multiplied by the ScaleFactor



Properties	
NumericFormat	0.0
ScaleFactor	2
Text	100

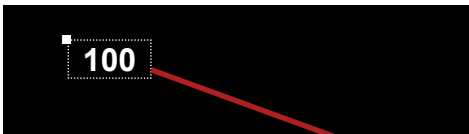
NumericFormat is used to set the number of digits to the right of the decimal. If a value of "0.0" is put in NumericFormat, then we can see our number becomes 200.0





## TextPrefix and TextSuffix Properties

In many cases you will want to show units of the value or an indication of what it represents. This is where a prefix and/or suffix can help.



Starting out with nothing in the TextPrefix and TextSuffix, the value shown is what's in the Text property. When we add something in the TextSuffix property, we can see that it gets appended to the Text value.

Properties	
TextPrefix	
TextSuffix	Lbs
Text	100

The TextSuffix is appended after the value put in Text.



Properties	
TextPrefix	Weight:
TextSuffix	Lbs
Text	100

The TextPrefix will insert before Text





### PLC Related Properties

The BasicLabel has 4 properties that are related to connections to the PLC through a communication driver. These properties are shown below.

Properties	
CommComponent	EthernetIPforPLCSLCMicroComm1
PLCAddressEntry	N7:0
PLCAddressText	N7:0
PLCAddressVisibility	B3/0

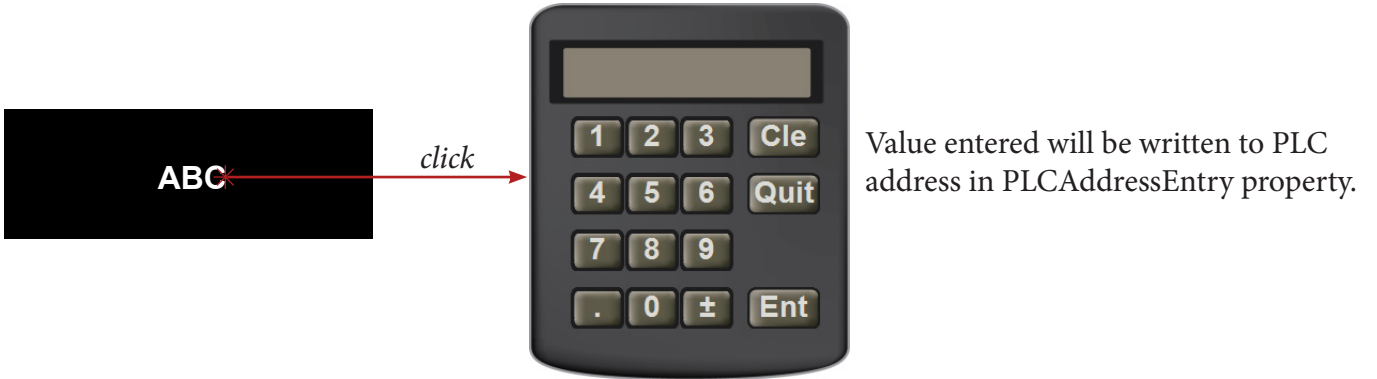
CommComponent tells the BasicLabel which driver instance to use therefore which PLC to obtain the values from. AdvancedHMI allows connections to multiple PLCs in a single application. This is achieved by adding multiple instances of drivers to a form and setting the connection properties accordingly. This will then give a selection option in the CommComponent property. Below shows a component tray with two driver instances.

 EthernetIPforPLCSLCMicroComm1

 EthernetIPforPLCSLCMicroComm2

PLCAddressEntry is used to allow values to be written back to the PLC by popping up a keypad when the BasicLabel is clicked on. The keypad will only popup if a PLC address is put in this property.

Properties	
PLCAddressEntry	N7:0



There are a range of properties that allow you to customize the look and behavior of the keypad. KeypadFont, KeypadForeColor, and KeypadText are used to display text at the top of the keypad. This is mainly used to give the user an indication of what the data being keyed in is related to.

Properties	
KeypadFont	<b>Arial, 8pt</b>
KeypadFontColor	<b>N7:0</b>
KeypadMaxValue	<b>1000</b>
KeypadMinValue	<b>0</b>
KeypadText	<b>Enter a New Value</b>
KeypadWidth	<b>300</b>

KeypadMaxValue and KeypadMinValue limit the allowable range that is to be written to the PLC. If a value is entered outside of the range, then a message box will pop up to notify the user.

KeypadWidth alllows you to set the size of the popup keypad. This may be necessary to accomodate differet screen resolutions.

PLCAddressText is the main property used for showing values from the PLC. By putting a valid PLC address in this property, the value read from the PLC will be put in the Text property therefore displaying the value.

PLCAddressVisibility is used to control whether the BasicLabel is visible during runtime based on a value from the PLC. If a valid address is put on this property, then the BasicLabel will only show on the form when the value from the PLC is true. This property has a unique feature to allow you to reverse the logic. By putting the keyword “NOT” in front of the PLCAddress will make it visible when the value in the PLC is False.

Properties	
PLCAddressVisibility	<b>NOT B3/0</b>

Inserting the keyword “NOT” before the PLC address will invert the logic of the PLCAddressVisibility property.

## Events and Manipulation With Visual Basic Code

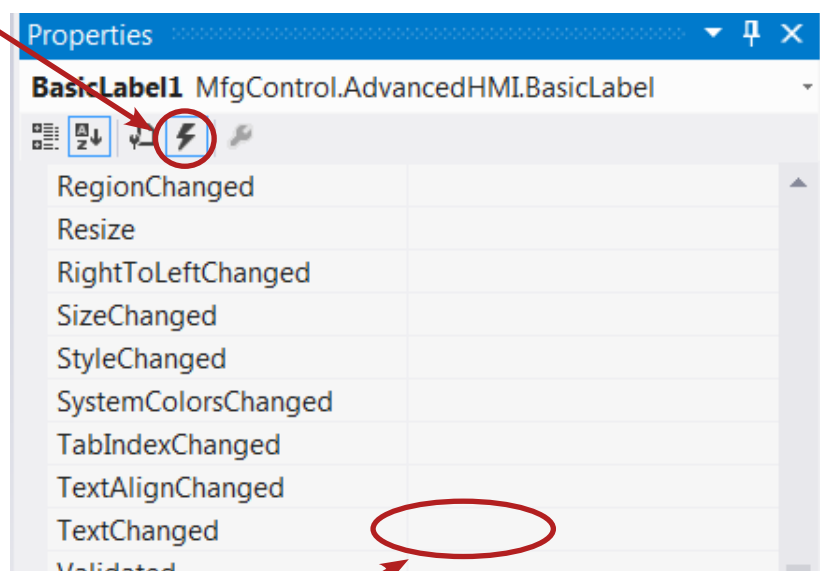
Nearly every control has some events associated with it that trigger when a certain condition occurs. Handling events does require writing some code. When it comes to writing code, the topic can easily span over a 1000 page book. Since this manual is not intended to be a code writing manual, we are just going to cover a commonly use event to give some a real example of how events can be used.

### TextChanged Event

This particular event may be the one event that gets used the most. As the name implies, whenever the contents of the Text property change, this event fires. An example of when to use this event is to check the value of the BasicLabel and change the Forecolor based on the value. For example, if the value exceeds 50 then you can change the color to yellow. If it them exceeds 100, you can change it to red. So now let's look at the process of handling the event to perform this.

Add a BasicLabel to your MainForm and be sure it is selected. Now go to the properties window and near the top of the windows you will see some icons and one of them will look like a lightening bolt. Click on the lightening bolt to see all of the possible events.

Select your events and find the  
TextChanged Event



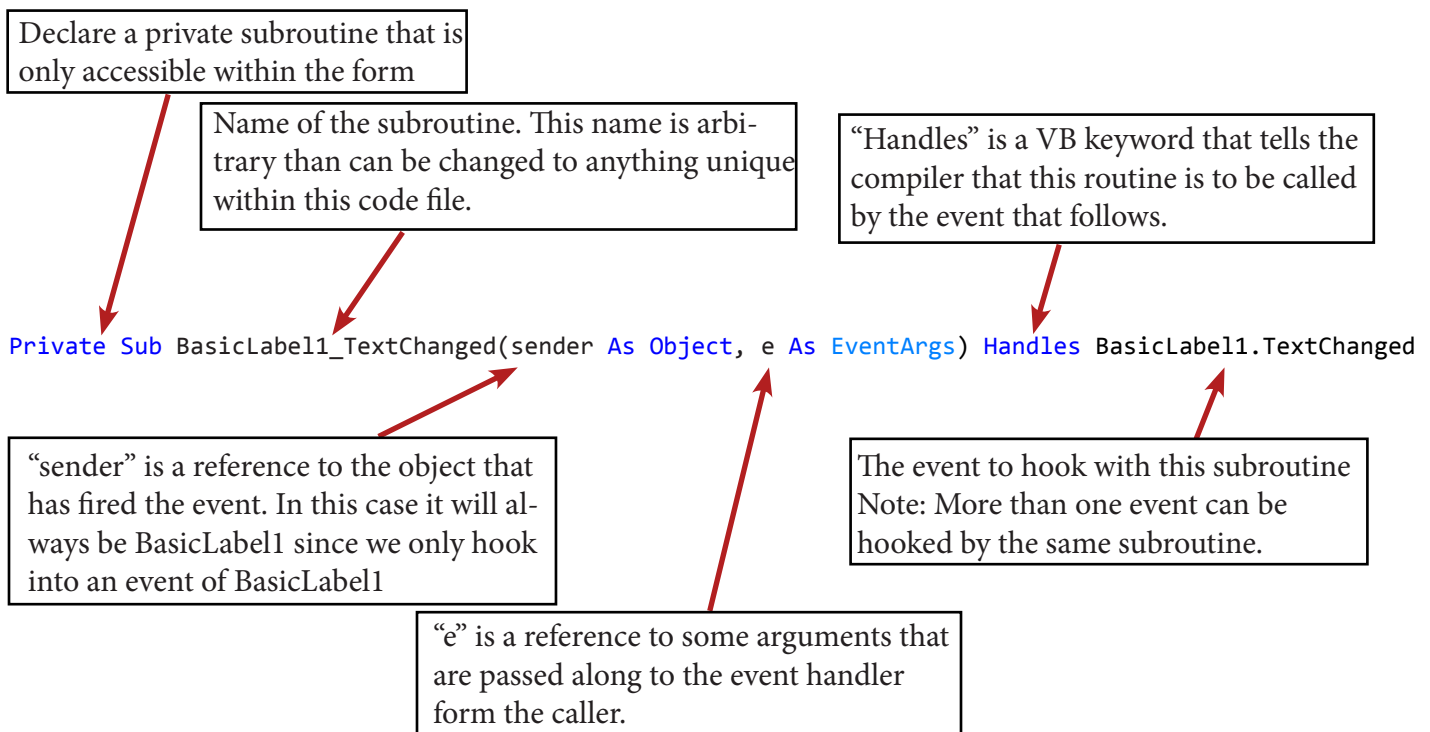
Double click in this area to the right  
of TextChanged

After double-clicking, you will be taken to the code window. This is where you will type in the VB code to perform the required action. In this example, we will change the font color (ForeColor) based on the value coming from the PLC.

Shown below is the code that will change the text to red when it is greater than 100:

```
Private Sub BasicLabel1_TextChanged(sender As Object, e As EventArgs) Handles BasicLabel1.TextChanged
    If BasicLabel1.Text > 100 Then
        BasicLabel1.ForeColor = Color.Red
    Else
        BasicLabel1.ForeColor = Color.White
    End If
End Sub
```

Let's dissect the code to help understand everything that is happening.



Conditional statement keyword “If” used in VB frequently to check conditions. After the condition, the keyword “Then” must follow

`If BasicLabel1.Text > 100 Then`

Condition to check for. In this case we refer to the Text property within our BasicLabel1 instance. Since Text is a type of string, VB will implicitly attempt to do a conversion to type integer and check if it is greater than 100

If the condition is True, then the first action will execute:

`BasicLabel1.ForeColor = Color.Red`

We reference our instance BasicLabel1 and the property “ForeColor”

“Color” is a predefined object in VB that holds constant values of many commonly used colors. In this case we are assigning the values for Red to the ForeColor property of our BasicLabel1 instance.

“Else” is a VB keyword used to indicate what to execute in case the “If” condition is False

`Else`

Just as above, we are assigning a color to the ForeColor property of our BasicLabel1 instance. Except in this case it will be assigned white when the comparison is False

`BasicLabel1.ForeColor = Color.White`

And finally we close out If-Then block using the keywords “End If”

`End If`