Better input boxes for AVR for .NET Windows apps

Although .NET includes a masked input control, its behavior is a little finicky and its most effective with fixed-size inputs (eg, a US social security number) It doesn't work well for entering numeric values of varying lengths.

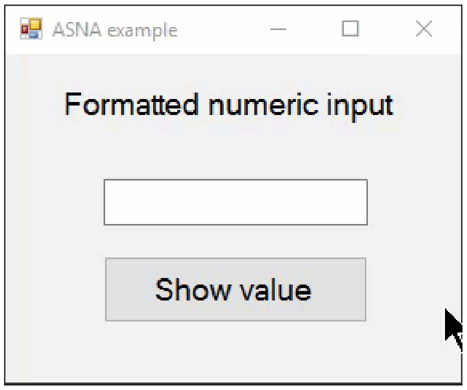
This article provides three subroutines you can add to your Windows form projects that pumps up the numeric input capabilities of the standard System.Windows.Forms.TextBox.

The features this code adds are:

- Numeric-only input (no letters are allowed.)
- A single minus sign (-) may be entered, but it must be at the start of the number.
- A single decimal point is allowed.
- Special keyboard control keystrokes (such as the tab and backspace key) are allowed.
- The decimal length is set explicitly, and enforced, for each TextBox.

> The TextBox control has a TextAlign prroperty. Setting this property to `Right` for numeric input works well with this technique.

The GIF below shows this numeric formatting in action.



Hooking things up

To register a TextBox to this code, use the provided `RegisterTextBoxForFormattedInput` method like this:

```
RegisterTextBoxForFormattedInput(textBox1, 2)
```

where the first argument is the name of the TextBox to register and the second argument is the number of decimal places.

Add these two `Using` statments at the top of your class:

```
Using System.Globalization
Using System.Text.RegularExpressions

<small>Figure x.y</small>
```

And add the three subroutines below:

```
BegSr RegisterTextBoxForFormattedInput
    DclSrParm tb Type(System.Windows.Forms.TextBox)
    DclSrParm MaxDecimalPlaces Type(*Integer4)

    tb.Tag =  MaxDecimalPlaces

    AddHandler SourceObject(tb) +
               SourceEvent(keypress) +
               HandlerSr(AllowOnlyNumericAndDecimalInput)

    AddHandler SourceObject(tb) +
               SourceEvent(leave) +
               HandlerSr(FormatNumericOnLostFocus)
EndSr
```

Figure 1a. This routine registers TextBox for formatted input.

```
BegSr AllowOnlyNumericAndDecimalInput Access(*Private) DclSrParm sender Type(*Object)
 DclSrParm e Type(System.Windows.Forms.KeyPressEventArgs)

    DclFld nfi Type(NumberFormatInfo)
    DclFld DecimalSeparator Type(*String)
    DclFld AllowableCharacters Type(*String)
    DclFld DecimalDigitsPattern Type(*String)
    DclFld MaxDecimalPlaces Type(*Integer4) Inz(2)

    // Get the decimal separator for the current culture.
    nfi = CultureInfo.CurrentCulture.NumberFormat
    // Or if you need to get the decimal separater for
    // a specific culture:
    DecimalSeparator = nfi.NumberDecimalSeparator

    // Define allowable input characters.
    AllowableCharacters = '0123456789-' + DecimalSeparator

    //DecimalDigitsPattern = '\.\d{2}$'
    If (sender *As TextBox).Tag <> *Nothing
        MaxDecimalPlaces = (sender *As TextBox).Tag.ToString()
    EndIf
    DecimalDigitsPattern = String.Format('\{0}\d{{{1}}}$', +
                                     DecimalSeparator, +
                                     MaxDecimalPlaces)

    // Allow only control characters (eg backspace)
    // and ALLOWABLE_CHARACTERS.
    If NOT System.Char.IsControl(e.KeyChar) AND +
       NOT AllowableCharacters.Contains(e.KeyChar.ToString())
        e.Handled = *True
    EndIf

    // Allow only one decimal point and one minus sign.
    If e.KeyChar = DecimalSeparator  AND +
       (sender *As TextBox).Text.Contains(DecimalSeparator) OR +
       (e.KeyChar = '-' AND (sender *As TextBox).Text.Contains('-'))
        e.Handled = *True
    EndIf

    // If minus sign is present make sure it is at the
    // beginning of the input.
    If e.KeyChar.ToString() = '-' AND +
       NOT (sender *As TextBox).Text = String.Empty
       e.Handled = *True
    EndIf

    // Ensure that the maximum number of decimal places isn't
    // exceeded.
    If char.IsDigit(e.KeyChar) AND +
       (sender *As TextBox).Text.Contains('.') AND +
       Regex.Match((sender *As TextBox).Text, +
                   DecimalDigitsPattern).Success
        e.Handled = *True
    EndIf
EndSr
```

Figure 1b. This routine filters input characters for the TextBox.

```
BegSr FormatNumericOnLostFocus Access(*Private)
 DclSrParm sender Type(*Object)
 DclSrParm e Type(System.EventArgs)

    DclFld ValueEntered Type(*Packed) Len(16,12)
    DclFld MaxDecimalPlaces Type(*Integer4) Inz(2)
    DclFld NumericFormattingString Type(*String)

    If (sender *As TextBox).Tag <> *Nothing
        MaxDecimalPlaces = (sender *As TextBox).Tag.ToString()
    EndIf

    // The "magic" Fx (where x is the number of decimal places)
    // used in the ToString() method below is .NET numeric
    // formatting parlance for fixed-decimal with MaxDecimalPlaces
    // places.
    // Click this Microsoft link to learn more about numeric formatting.
    // https://t.ly/e9jkr
    // If MaxDecimalPlaces = 2 this resolves to
    // NumericFormattingString = 'F2'

    NumericFormattingString = String.Format('F{0}', MaxDecimalPlaces)

    ValueEntered = (sender *As TextBox).Text

    (sender *As TextBox).Text = +
          ValueEntered.ToString(NumericFormattingString, +
                                CultureInfo.InvariantCulture)
EndSr
```

Figure 1c. This routine sets the value displayed in the TextBox when it loses focus.