Name of Experiment: Display the Unicode for the key-board characters.	**Exp No:**WP4
Background: Student should have a basic knowledge of C#.	
Summary: After going through this experiment, the student is aware of the fact that each character on the alpha-numeric virtual keyboard has a unique constant value (i.e. ASCII) associated with it.	
Learning Objective: To explain the Unicode value associated with each alpha-numeric character in the keyboard.	
Target Platforms: This application is tested on Windows Phone Emulator and Windows Phone (Lumia 800).	

Procedure:

Step1: Please repeat steps [1-6] as in experiment no WP1.

Step2: Go to Stack Panel->Text Block->Name= “Experiment Title” and Text=“Exp No: WP4” and set Text Alignment as “RIGHT”.[Refer source code section]

Step3: Do the similar task, go to Stack Panel->Text Block->Name = “Application Title and Text= “Display Unicode Character”.[Refer source code section]

Step4: Now, Drag and drop the Textbox, Button and Text block from the Toolbox.

Step5: Align all control elements as the UI given in fig. no 1. [Refer fig. no1]

Step6: Initialise bool hint= “true”, string hint_text= “Enter the character” before the MainPage constructor.

Step7: Check for empty navigation, i.e. a message should be popped up if the user will click on the button without entering any value to the given textbox.

Step8: After that we will call the method named Print_Unicode(str1,u16LE), where str1 is a character array having the user’s entered character and an Encoding scheme object for 16bit Unicode character.[Refer source code section]

Step9: Print_Unicode method will converts the encoding stream into array of bytes.

Step10: Each byte of the array is printed by the method named PrintHexBytes().

Step11: Now, save all the changes made to the project.

Step12: Hit the F5 button from the keyboard, and your project is running on the Windows emulator for the debugging purpose.[Refer the fig. no. 2-8]

Source Code	Comments

MainPage.xaml
 	
`<!--LayoutRoot is the root grid where all page content is placed-->`
`<Grid x:Name="LayoutRoot" Background="Transparent">`
`<Grid.RowDefinitions>`
`<RowDefinition Height="Auto"/>`
`<RowDefinition Height="*"/>`
`</Grid.RowDefinitions>`
 	
`<!--TitlePanel contains the name of the application and page title-->`
`<StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">`
`<TextBlock x:Name="ExperimentTitle" Text="Exp No:WP4" TextAlignment="Right" Style="{StaticResource PhoneTextNormalStyle}"/>`
`<TextBlock x:Name="ApplicationTitle" Text="Display Unicode Character" Style="{StaticResource PhoneTextNormalStyle}"/>`
`<TextBlock x:Name="PageTitle" Text="Home Page" Margin="9,-7,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>`
`</StackPanel>`
 	
`<!--ContentPanel - place additional content here-->`
`<Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">`
`<TextBox Height="72" HorizontalAlignment="Left" GotFocus="textBox1_GotFocus" LostFocus="textBox1_LostFocus" Margin="44,91,0,0" Name="textBox1" Text="TextBox" VerticalAlignment="Top" Width="338" />`
`<Button Content="Get Unicode" Height="72" HorizontalAlignment="Left" Margin="110,198,0,0" Name="button1" VerticalAlignment="Top" Width="230" Click="button1_Click" />`
`<TextBlock Height="64" HorizontalAlignment="Left" Margin="133,326,0,0" Name="mytext" VerticalAlignment="Top" Width="189" />`
`</Grid>`
`</Grid>`
 	
MainPage.xaml.cs
 	
`using System;`
`using System.Windows;`
`using Microsoft.Phone.Controls;`
`using System.Text;`
	←Name= “ExperimentTitle” Text= “Exp No:WP4” TextAlignment= “Right”.
	
	←Name= “ApplicationTitle” Text= “Display Unicode Character”.
	
	←Name= “PageTitle” Text= “Home Page”.
	
	← Content= “Get Unicode”
	
	← For Encoding class

```
namespace Unicode
{

    public partial class MainPage :
PhoneApplicationPage
    {

        bool hint = true;
        string hint_text = "Enter the
character";
        //char[] mychars;

    // Constructor
        public MainPage()
        {
            InitializeComponent();
            textBox1.Text = hint_text;
        }

        private void button1_Click(object
sender, RoutedEventArgs e)
        {
            char[] str1;
if (textBox1.Text.Equals(hint_text) ||
textBox1.Text == string.Empty)
            {
                MessageBox.Show("Please
enter some character");
            }
            else
            {
                string str = textBox1.Text;
                str1 = str.ToCharArray();
       Encoding u16LE = Encoding.Unicode;

                Print_Unicode(str1, u16LE);
            }
        }
        public  void Print_Unicode(char []
st, Encoding en)
        {
            byte[] bytes = en.GetBytes(st);
            PrintHexBytes(bytes);
        }
        public  void PrintHexBytes(byte[]
bytes)
        {
            if ((bytes == null) ||
(bytes.Length == 0))
                Console.WriteLine("<none>");
            else
            {
         mytext.Text = string.Empty;

     mytext.Text = bytes[0].ToString();
    }
       }
                }
```

← Text display as a Hint

← Click event for Get Unicode Button

← check for empty textbox

← Message for the user.

← Encoding object for Unicode 16

← call for Print_Unicode method.

← body for Print_Unicode method.

← call for PrintHexBytes method

← body for PrintHexBytes method

← assigning Unicode to the textblock for display.

```
 private void textBox1_GotFocus(object
sender, RoutedEventArgs e)
        {
            if (hint) {
                textBox1.Text =
string.Empty;
                hint = false;
            }
        }

 private void textBox1_LostFocus(object
sender, RoutedEventArgs e)
        {
            if (!hint &&
textBox1.Text.Length==0) {
                textBox1.Text = hint_text;
                hint = true;
            }
        }
    }
}
```

← GotFocus method for entering the value in the textbox.

← LostFocus method for displaying the hint.

Screenshots

Fig no. 1 UI for the Home Page

Fig no. 2 Output Screen

Fig no. 3 Entering the character

Fig no.4 Enter the character ‘a’.

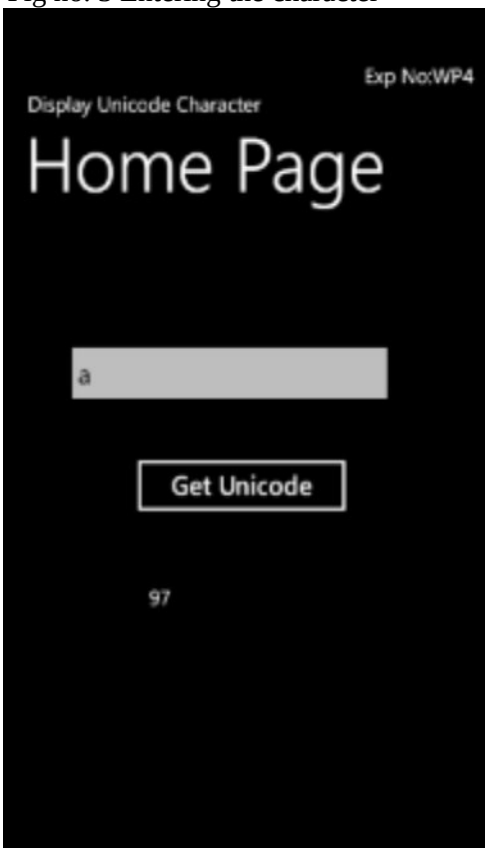

Fig no. 5 Showing the Unicode for the input

Fig no.6 Message alert on clicking empty textbox

Fig no. 7 Enter the next character as ‘b’.

Fig no 8 Showing the output for the input character

Observations: It is observed that each character in the virtual keyboard has been associated with some Unicode through which it is identified by the hardware.