

Visual Basic - System Information Viewer

Welcome to our special tutorial of visual basic. In this tutorial we will use Microsoft visual studio 2010 version. You can download it for free from their website. It will also work on other visual studio version if you have one.

We are going to use Visual Basic programming language. The main reason I've created this application in Visual Basic is to demonstrate how much is possible with it and also Its fun to make useful application.

We will create a simple system information viewer such as **CPUZ** if you aren't familiar with it you can look at their website and learn more from there. <http://www.cpubid.com/>. It's a simple application which allows you to see what processor, graphics card, operating system hard drive you are using in your system. It's a very useful application and the developers have done a great job.

Check out the demo video of this application so you know what you will be making in this tutorial.

Note: This will be a long tutorial so don't haste in your coding, take your time and understand what you are inputting. Happy day after that.

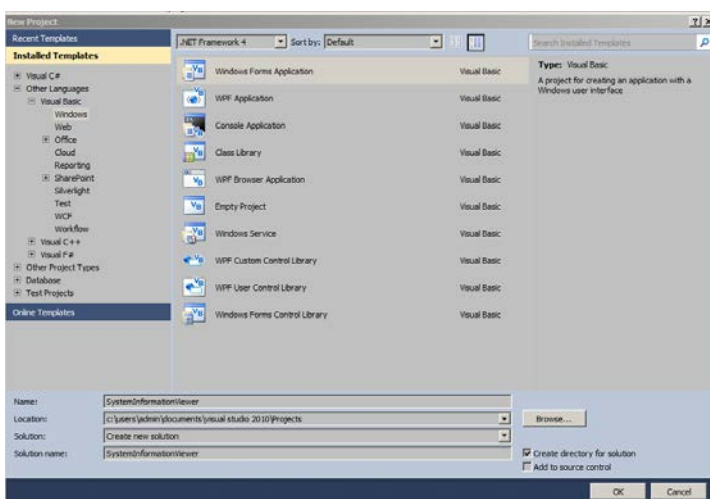
Application requirements

This System Information Viewer is required to do the following:

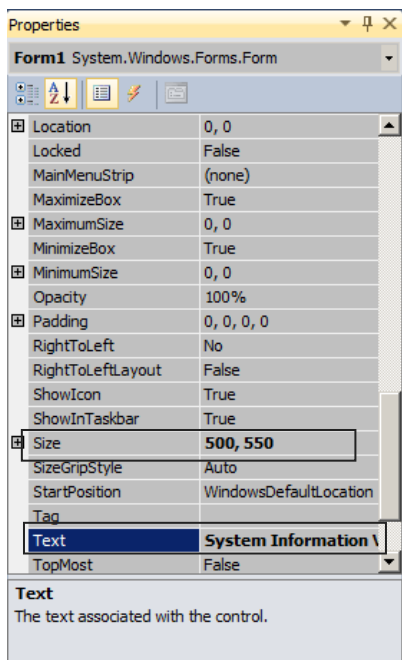
1. Show CPU (Central processing Unit) information such as name and speed of it
2. Show GPU (Graphical processing unit) information such as name, speed, RAM etc
3. Show operating system and user information
4. Show RAM(Random Access Memory) information. How much is available to the system
5. Show storage information such as hard drive, DVD or any connected USB or network drives
6. User should be able to save the information to Text file
7. User should be able to print that information

Look again that is what we will be making. Yep

Now then. Open Visual Studio and start a new project. Name this project **SystemInformationViewer**.



To start with click on the empty Form and Now look at the properties window. If you cannot see it then right click on the form and click on properties.



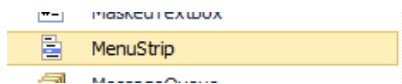
Change the size to 500, 550 and change the Text to System Information Viewer.

The text will show up on the title of the Form so it's more informative. We are changing the height and width of the form so it can fit all the elements inside the form and its visually available to the user.

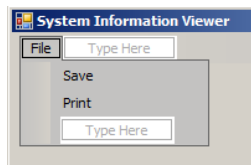
Small warning we will be using loads of text boxes and labels so we won't be giving them any unique names just try to keep track of them while following this tutorial.

Now lets complete the user interface and then we can start entering the code.

First lets add a menu strip to the form

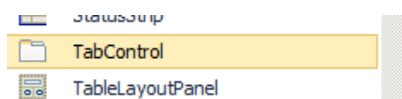


You will find it in the toolbox.

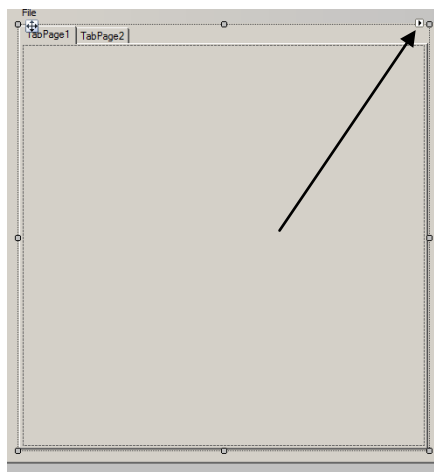


When you click on it, there is option to enter your own menu. So for now we called the menu File. Under file we have entered Save and Print option.

Now lets the add the second component Tab Control

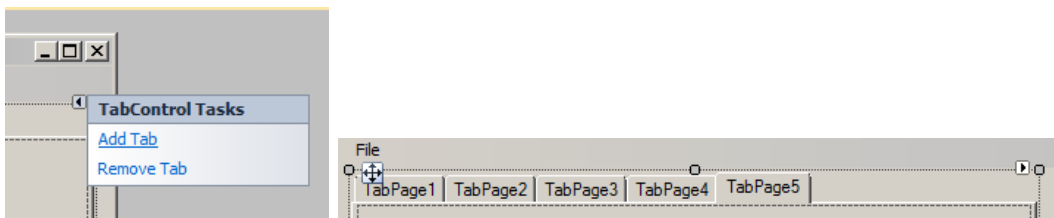


With this we can create an app with multiple tabs. Such as when you use in Firefox or Google Chrome.



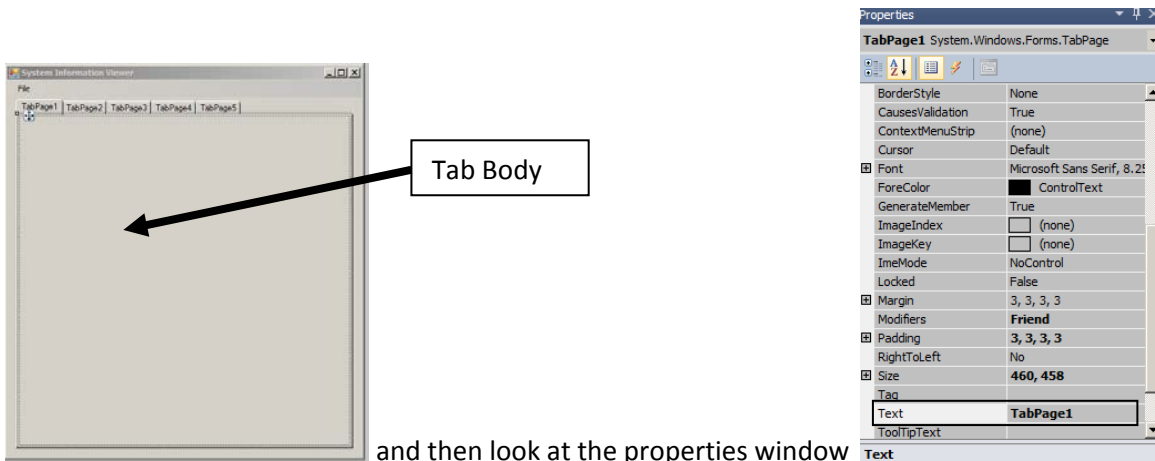
Now make the tab control fit around the form. We will define each of our tabs so the user knows where to go. Remember user interface can make or break your application.

For this application we will need 5 tabs. When click on the tab there is a small triangle shape on the top right corner. Click on that and then click on Add Tab.



Now there is 5 tabs. Time to change the titles of the tab.

Click on the TAB BODY



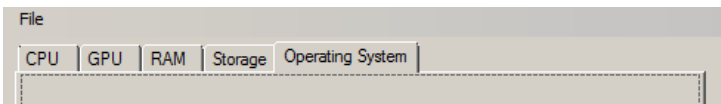
and then look at the properties window

Each tab has its own properties and can be changed. For now we want to change the text. Do the following

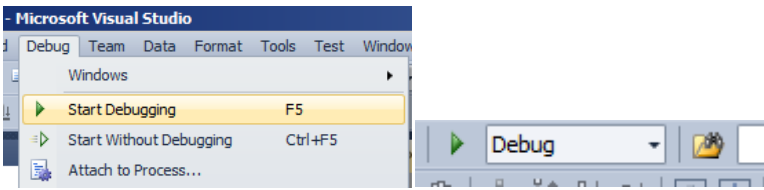
Components	Text Property
TabPage1	CPU
TabPage2	GPU
TabPage3	RAM
TabPage4	Storage
TabPage5	Operating System

Final view

more tutorials on www.mooict.com



Now let's run the program.



Either do from the debug menu or click on the play button next to the debug drop down menu it will do the same.



Yep it works. Well done

We will go through each Tab and start adding the components needed to view the information.

CPU - Central Processing Unit Tab

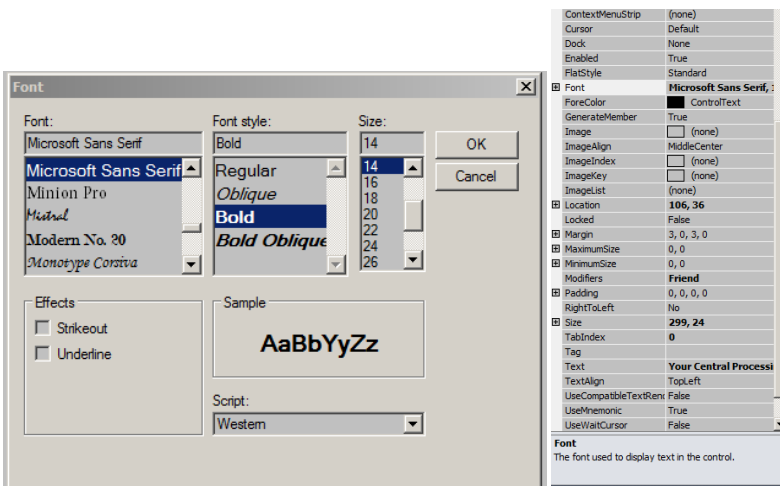
In this part we will be 2 labels and 1 button



We can change the properties of the label in the properties window.

Click on Label 1

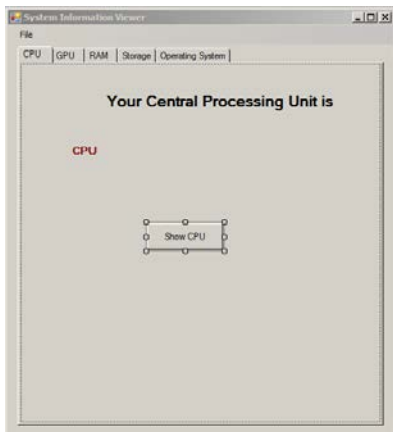
Change the Label 1 text to Your Central Processing Unit is: and change the font to size 14 and Bold



more tutorials on www.mooc.com

For Label 2 change the text to CPU. This label will change once the user clicks on the button. So just for a place holder show the letters CPU. For the font in this one you can change the fore colour to red, font to bold and size 10.

Change the button text to Show CPU



This is the final view of the CPU Tab.

GPU - Graphical Processing Unit Tab

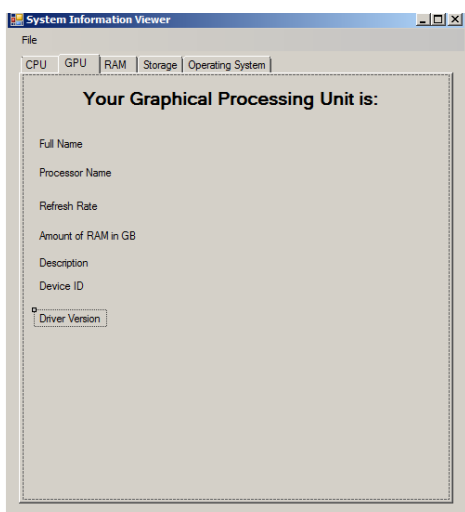
This tab to push out a lot of information about the GPU so we need to have enough labels and text boxes to show those information. For this tab we need 8 labels, 7 text boxes and 1 button.

Label 3 is our title label. So this one as we have done before for the CPU we can change the font and make it bold. The text for label 3 should be "Your Graphical Processing Unit is: "

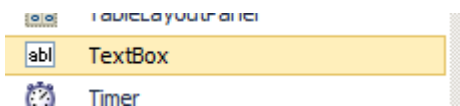
Follow the table below to change the texts of the labels

Components	Text Property
Label4	Full Name
Label5	Processor Name
Label6	Refresh Rate
Label7	Amount of RAM in GB
Label8	Description
Label9	Device ID
Label10	Driver Version

Here are the labels added and text changed

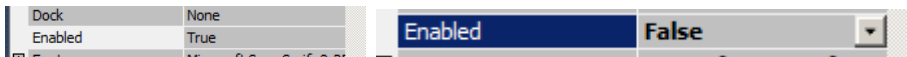


Now it's time to add the text boxes

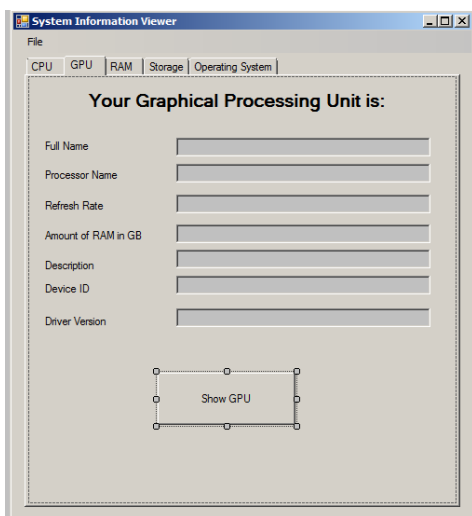


Here is the text box in the tools box menu. Drag it to the form.

Once you have added the text box to the form now select it and look at the properties window, find the option that says Enabled and change the value from true to false.



This will stop the user from changing the text in the box. We want the graphic card information to be seen in the text and be selectable but not to be edited or deleted.



Here is the final version of our GPU tab. I've added the button and changed the text to Show GPU.

Textbox 1 to Textbox 7 are used in the GPU tab.

Label 3 to Label 10 are used in the GPU Tab.

Ram, Random Access Memory Tab

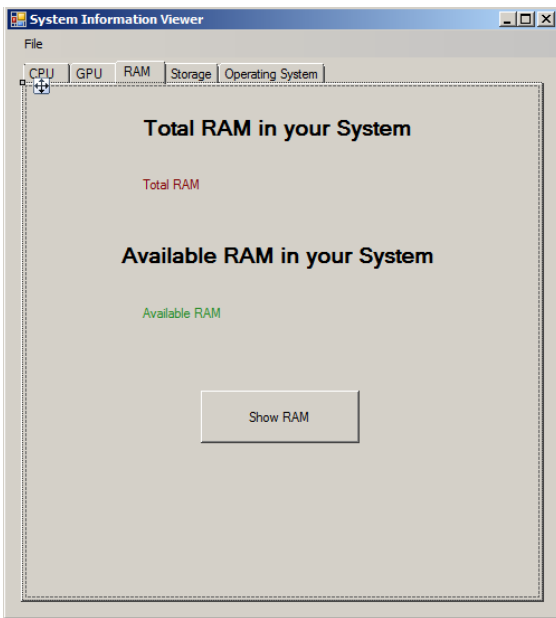
In this tab we will need 4 labels and 1 button.

Change the text of the labels according to the table below.

Components	Text Property
Label11	Total RAM in your System
Label12	Total RAM
Label13	Available RAM in your System
Label14	Available RAM

Label 12 and 14 will be change when the user clicks the button so it can simply say total and available RAM on them. You can also change the fore colour of the text for Label 12 to Maroon and Label 14 to Green.

Now change the button text to Show RAM.



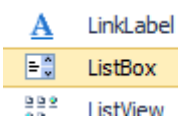
Here is the final view of the RAM tab.

I think this is a good time run the App and get a feel for what you are making. Go on run it and gloat.

Back to reality. We have two more tabs to fill before going in to coding.

Storage Tab

In this tab we will need 6 labels, 1 list box, 5 text box and 1 button. Quite a shopping list.



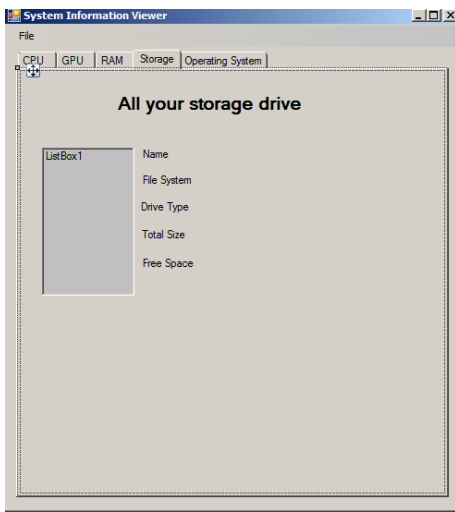
Drag the list box to the form

Now add the 6 labels and arrange them.

Change the texts of the labels to the table below:

Components	Text Property
Label15	All your storage drives
Label16	Name
Label17	File System
Label18	Drive Type
Label19	Total Size
Label20	Free Space

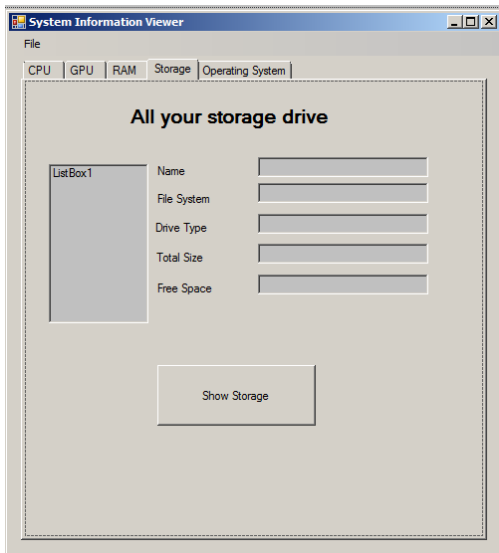
(if you have been following the tutorial so far we are starting at label 15 now. If yours is different check again)



Here we have added all of our labels. Notice Label 15 is the title of the tab so make it bigger and bolder than the rest.

Now add the 5 text boxes. As before we need to change the Enabled properties for each text box from True to False.

Also add the button to the form and change the text to "Show Storage"



This is the final view of the tab.

If you are wondering why are we using a list box to show the storage of a computer. Well most of the times now we have multiple hard drives and network drives connected to a computer at any one point. There is no way to determine whether the drive will have C: D: or event E: so why guess the drive right. We will use some clever code to list all the available drives inside that list box and once we select them the information will dynamically be changed according to the drive itself.

Now for the final Operating systems Tab

In the operating system tab we will need 7 Labels 6 text boxes and 1 button.

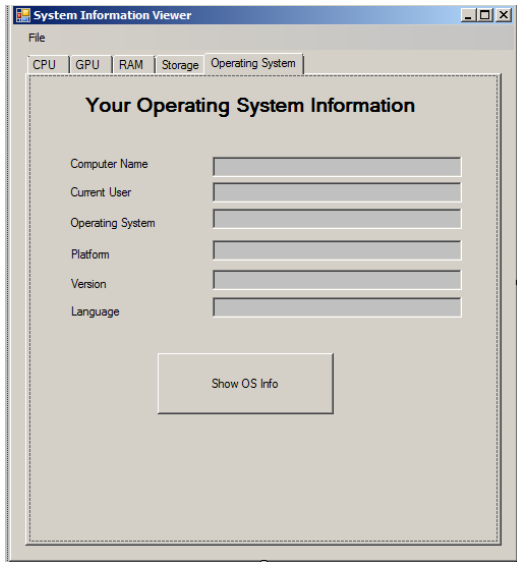
It's the same thing as before for the text boxes change the Enabled from True to False and follow the table below to see what to change the Label texts to:

Components	Text Property
Label 21	Your Operating System Information
Label 22	Computer Name
Label 23	Current User
Label 24	Operating System
Label 25	Platform
Label 26	Version
Label 27	Language

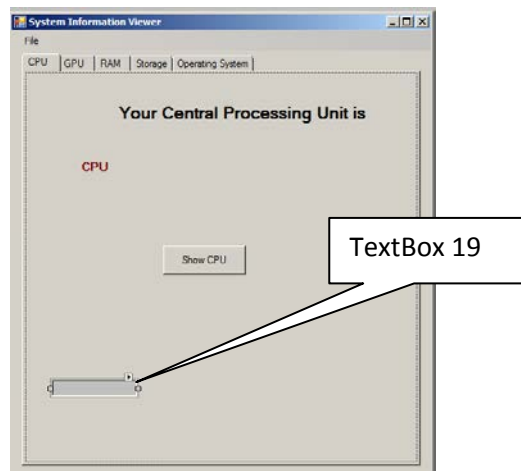
Label 21 is our title label for this tab. So make the font bigger and bold.

For the button change its text to "**Show OS Info**"

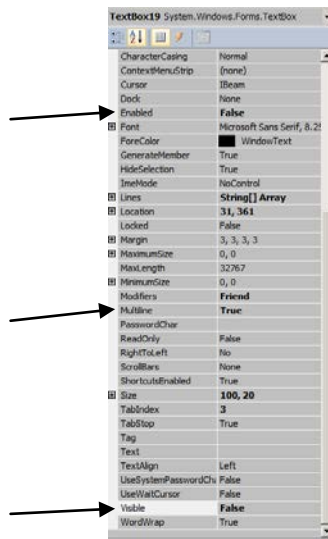
Here is the final user interface for the operating system Tab.



Remember where we planned to print the information from this application. We now need to add one more Text Box on the form. We will load all the information to the form and then we will print them.

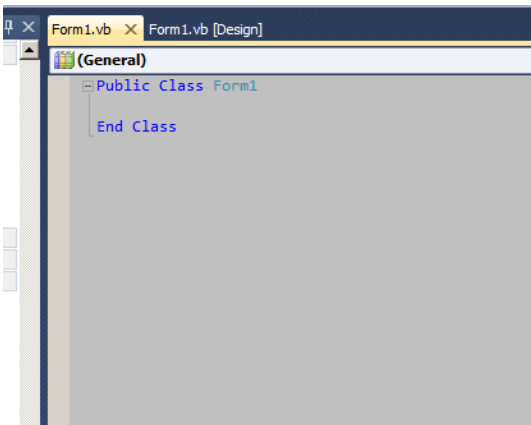


In the properties for the text box change the following



Change enabled to FALSE and multiline to TRUE and Visible to FALSE. We don't need to see this text only need to load the information it so we can print them all. Now if you run the program the last text box will not be visible on the form any more.

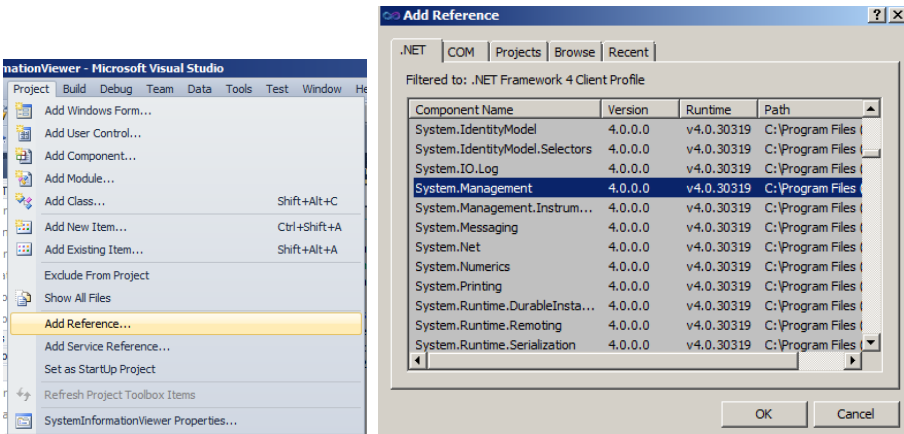
Now run the application to ensure you are happy with how it looks. Next comes the fun part all the coding yaay.



This is what our code view looks like at the moment.

Before we get started on pulling the data from the system we need to add some references to our project which will make the world load a lot easier.

First click on the project option and then add reference ->



Click on System.Management

and click ok

more tutorials on www.mooict.com

Now add the following lines before the Class Form1

```
Imports System.Management
Imports System
Imports System.IO
Imports System.Drawing.Printing
Imports System.Runtime.InteropServices
```

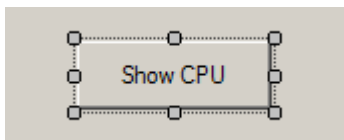
```
Imports System.Management
Imports System
Imports System.IO
Imports System.Drawing.Printing
Imports System.Runtime.InteropServices

Public Class Form1
```

These 5 lines will import all the functionalities we need from Windows thus making it easier for us to pull data and also to print them in the future task.

Central Processing Unit Information

Double click on the Button1 which states Show CPU



```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
```

```
End Sub
```

Visual studio will automatically add the code above and link it to the button. We need to add our own actions before the End Sub line.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
```

```
' shows the processor name and speed of the computer
```

```
Button1.Enabled = False
```

```
Dim MyOBJ As Object
```

```
Dim cpu As Object
```

```
MyOBJ = GetObject("WinMgmts:").instancesof("Win32_Processor")
```

```
For Each cpu In MyOBJ
```

```
Label2.Text = cpu.Name.ToString + " " + cpu.CurrentClockSpeed.ToString + " Mhz"
```

```
Next
```

```
End Sub
```

Lets look at the code above in a logical way. First when the button is clicked we are disabling the button because we don't want the user to click on it multiple times. We created a MyOBJ variable which will hold object types in the memory. Since the CPU information stored in the computer are counted as objects we can put those information in this variable. Then we created a CPU variable which is an object data type. The reason we have created two different object variable is because we will put the whole cpu information in one and then loop through that first one with the second object and collect the information we need.

```
MyOBJ = GetObject("WinMgmts:").instancesof("Win32_Processor")
```

In this line we are giving the value GetObject("WinMgmts") get the information from windows management .Instancesof("Win32_Processor") only the processor information.

We need to loop through the MyOBJ because there are multiple core to a single CPU now so we need accurate information.

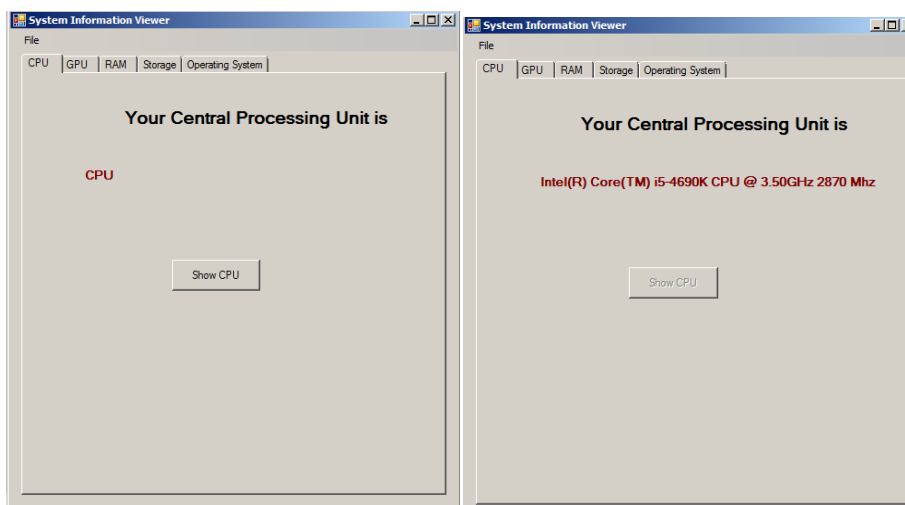
```
For Each cpu In MyOBJ
```

```
Label2.Text = cpu.Name.ToString + " " + cpu.CurrentClockSpeed.ToString + " Mhz"
```

```
Next
```

For each is a loop where we will loop through each object stored inside of MyOBJ variable. Once we find them we will update the **Label2.Text** component with the string information of CPU name and clock speed.

Now lets run the application and click on the button.

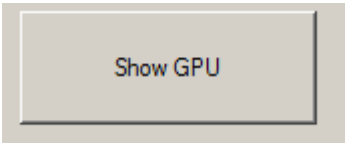


Okay this worked Phew. On to the GPU information now.

Graphical Processing Unit Information.

We all love a good GPU am I right, remember that reference stuff we had to add to the project in the beginning we will be using some of those instances in this tutorial. Its very simple kind of similar to the CPU stuff.

Firstly once again double click on the Show GPU button. In our case its called Button2.



```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
```

```
End Sub
```

The code above will added automatically once you double clicked on the button. Lets add our own

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
```

```
Button2.Enabled = False
```

```
Dim objectQuery As New ObjectQuery("SELECT * FROM Win32_VideoController")
```

```
Dim objectSearch As New ManagementObjectSearcher(objectQuery)
```

```
For Each memObj As ManagementObject In objectSearch.Get
```

```
    TextBox1.Text = memObj("Name")
```

```
    TextBox2.Text = memObj("VideoProcessor")
```

```
    TextBox3.Text = Convert.ToInt64(memObj("MaxRefreshRate")).ToString & "Hz"
```

```
    TextBox4.Text = Convert.ToInt64(memObj("AdapterRAM") / 1048576 / 1024).ToString & "GB"
```

```
    TextBox5.Text = memObj("Description")
```

```
    TextBox6.Text = memObj("DeviceID")
```

```
    TextBox7.Text = memObj("DriverVersion")
```

```
Next
```

```
End Sub
```

Firstly we will disable the button once its been clicked.

We are creating a Object query variable. Since there can be multiple GPU's in a system windows has its own database to store the information in. We will need to query or search that database for a specific component in this case the GPU.

Then we are creating another variable in this one we will give it a type of Management Object Searcher. Nothing we are using this line **Dim objectSearch As New ManagementObjectSearcher(objectQuery)** here we are putting the first object query variable inside it. We are doing this because we want the object search variable find those GPU information inside the windows database.

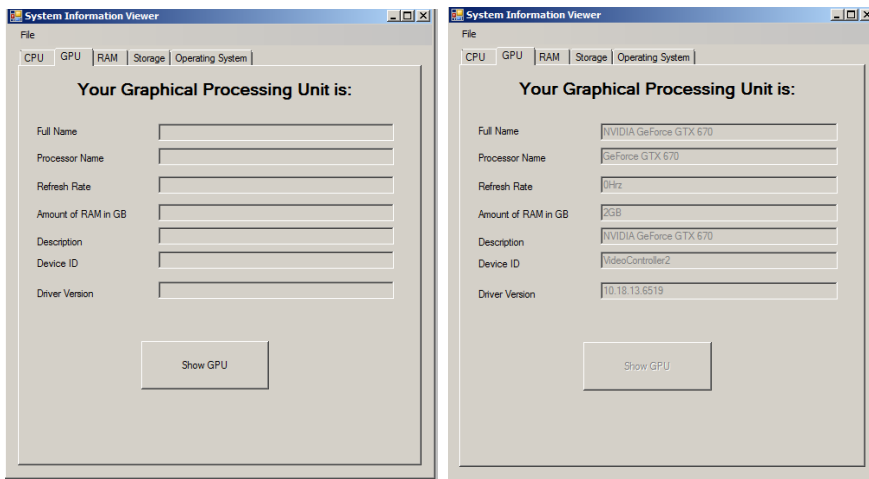
Now we have 7 text boxes in this tab. Textbox 1 - textbox 7 we will need to populate those with the given information. So we will run a loop and grab those information and put them all in the appropriate boxes.

```
TextBox4.Text = Convert.ToInt64(memObj("AdapterRAM") / 1048576 / 1024).ToString & "GB"
```

This line above might get you curious. Windows displays its RAM in byte mode which means we have to convert to mega byte and then to Gigabyte. Hence which we are diving the initial number of VRAM by **1048576** and then divide again by **1024** which will give us an approximate number of the total ram.

Lets test this out now shall we.

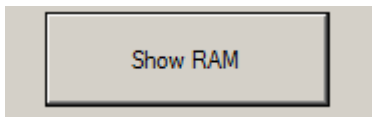
more tutorials on www.mooint.com



Neat. Moving on to RAM now.

RAM - Random Access Memory

Lets get started on this one now. Double click on the Show Ram button. In our case its Button3.



Same as before visual studio will add some code to link the button and an event.

Lets add the following code to the new button.

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button3.Click

    Button3.Enabled = False

    Dim ramAmount As Double

    ramAmount = (My.Computer.Info.TotalPhysicalMemory / 1048576) / 1024

    Label12.Text = "RAM: " & ramAmount.ToString("N") & "GB"

    Dim ramLeft As Double = (My.Computer.Info.TotalPhysicalMemory -
My.Computer.Info.AvailablePhysicalMemory) / 1048576 / 1024

    Label14.Text = "RAM: " & ramLeft.ToString("N") & "GB"

End Sub
```

First of all we will disable the button. create a ramAmount variable and give it a data type of double.

Inside the ram amount variable we will pull the system information for the physical ram and covert it GB as we did with the GPU memory.

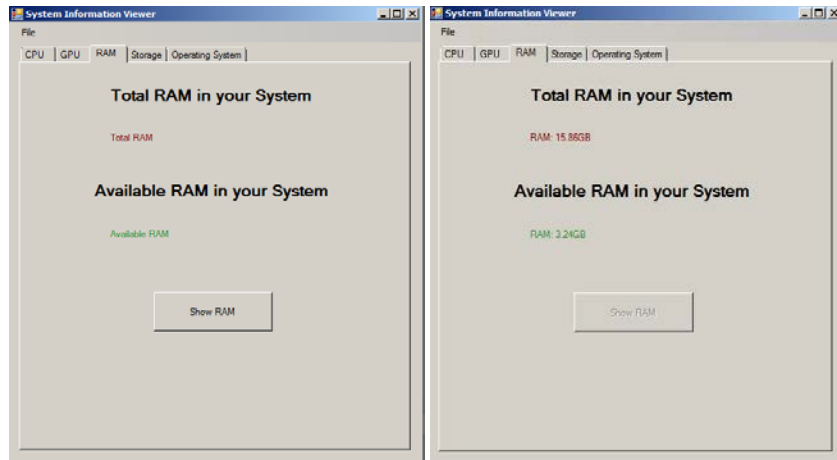
Label12 will show the final RAM amount on the screen. Hence we are changing its text value by Label12.Text = ""

more tutorials on www.mooict.com

Create another double variable called ramLeft in this variable we will deduct the total physical memory minus the total available memory. Then we convert that final number to a GB or gigabyte.

Lastly label 14 will show the total available ram information.

Lets test this out now.



Done.

Storage - Hard drive, CD,DVD or USB

We have made our way to the storage information section now.

Before this we have only dealt with buttons, text boxes and labels which is fine however we have new component in this section. What is it? its a list box. The idea for this section is once we click that button we will scan the whole computer and populate that list box with the drives available. So we can click on each drive it will give us the relative information about it.

Firstly double click on the show storage button.



Now add the following code inside the button event

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button4.Click

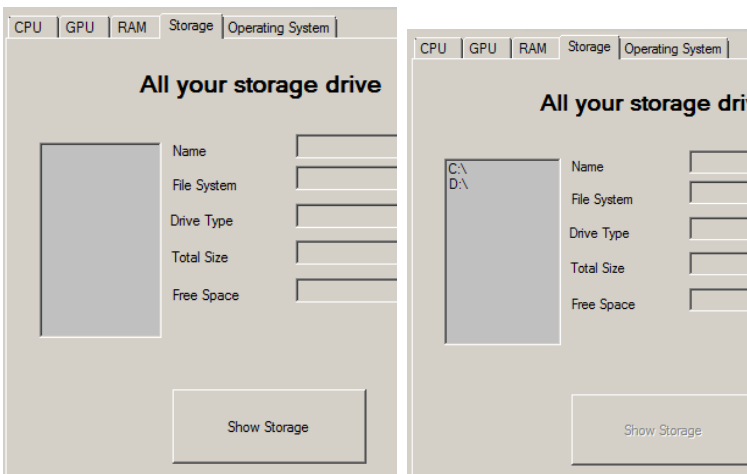
    Button4.Enabled = False

    For Each Drv As IO.DriveInfo In IO.DriveInfo.GetDrives
        ListBox1.Items.Add(Drv)
    Next
End Sub
```

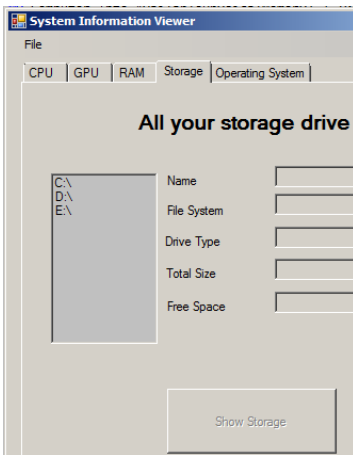
This button will disable itself once clicks and then populate the drives information in the list box.

Lets test it out, run the program now.

more tutorials on www.mooict.com



By clicking on the button its gives me the C and D drive which is correct. Now let me connect a USB to the computer and run it again see if it picks up.



Yes it did. Well done

We can click on the items inside the list box but we cannot see any information about them. We need to do the same to list box as we have done with buttons. We need to link an event to it. Now double click on the list box.



Visual studio will automatically add some code to link it with an event.

```
Private Sub ListBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ListBox1.SelectedIndexChanged
End Sub
```

So each time the selected index of this box has been changed it will run the code inside this function above.

Add the code below inside that ListBox1 function.

```
Dim selectedDrive As String = ListBox1.SelectedItem.ToString
Dim driveReady As String = My.Computer.FileSystem.GetDriveInfo(selectedDrive).IsReady
If driveReady = True Then
    TextBox8.Text = My.Computer.FileSystem.GetDriveInfo(selectedDrive).VolumeLabel
    TextBox9.Text = My.Computer.FileSystem.GetDriveInfo(selectedDrive).DriveFormat
End If
```

more tutorials on www.mooict.com


```

Dim driveTypeNum As Integer = My.Computer.FileSystem.GetDriveInfo(selectedDrive).DriveType

If driveTypeNum = 0 Then
    TextBox10.Text = "Unknown"
ElseIf driveTypeNum = 1 Then
    TextBox10.Text = "CD"
ElseIf driveTypeNum = 2 Then
    TextBox10.Text = "Removeable Media"
ElseIf driveTypeNum = 3 Then
    TextBox10.Text = "Hard Drive"
ElseIf driveTypeNum = 4 Then
    TextBox10.Text = "Network Drive or Cloud Drive"
ElseIf driveTypeNum = 5 Then
    TextBox10.Text = "Network Drive or Cloud Drive"
ElseIf driveTypeNum = 6 Then
    TextBox10.Text = "Removeable Media"
End If

Dim totalMemory As Double = My.Computer.FileSystem.GetDriveInfo(selectedDrive).TotalSize / 1048576 / 1024

TextBox11.Text = Format(totalMemory, "0.00") & " GB"

Dim freeSpace As Double = My.Computer.FileSystem.GetDriveInfo(selectedDrive).AvailableFreeSpace / 1048576 / 1024
TextBox12.Text = Format(freeSpace, "0.00") & " GB"
Else
    MessageBox.Show(selectedDrive & " - Is not ready")
End If

```

first we created a select drive variable this will store which drive is selected from the list and then we can take appropriate action accordingly.

Drive ready variable will get the information from windows whether the drive is ready or not.

we can run an if condition when the drive is ready

Text box 8 will show the volume label or name of the drive

Text box 9 will show the drive format type whether its a FAT, FAT32, NTFS etc.

Since there various types of storage devices we have broken it down and created our own inside an if statement which will collaborate with a number and represent a type of storage unit.

for example if the device type num variable returns a 0 it will be seen as an unknown device, 1 will be seen as a CD or DVD drive, 2 will be seen as a removable media, 3 will be seen as Hard Drive, 4/5 will be seen as network drive or cloud drive, 6 will be seen a removable drive as well. All of these information will be updated on the text box 10.

total memory is a double variable which will calculate the total memory of the device. Same as we done with the RAM and GPU before we are converting the byte to gigabyte and storing it inside the double.

We need to show the information is a two decimal number such as 67.98GB or 1.34GB etc we don't want a long 56.78765675675 GB because it will not make sense to the end user.

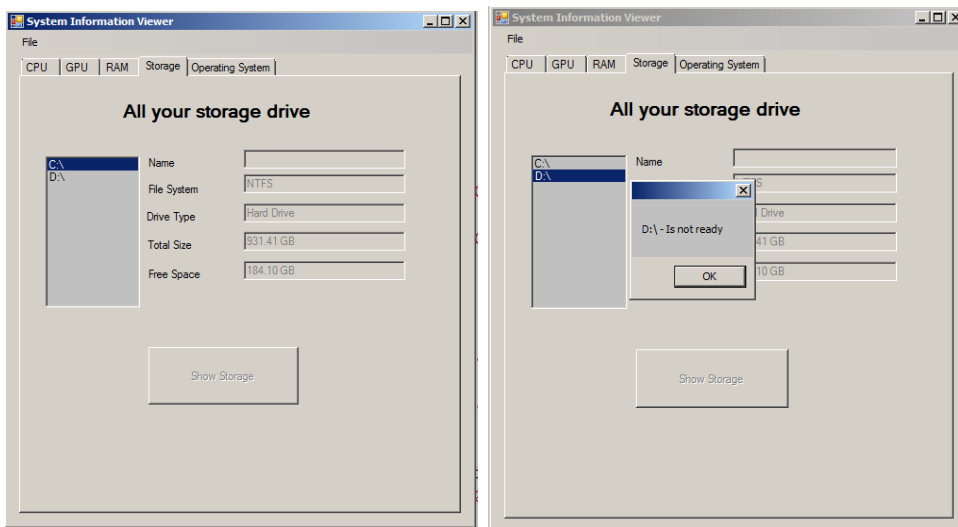
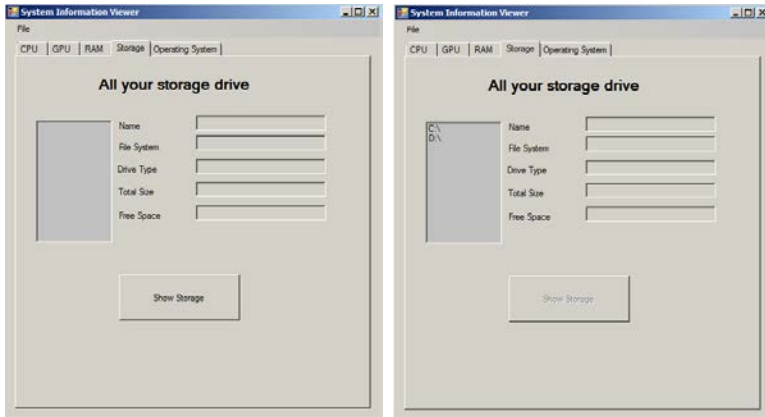
Inside the Text box 11 we will format its outs put by using the Format(totalMemory, "0.00") & "GB". Basically we are reducing the total memory numbers to a two decimal point and then showing it on the text box 11.

Free space variable is also a double data type this one will calculate the free space we have each drives and convert them to gigabytes as we have done few times before.

Textbox 12 will also be formatted to show two decimal numbers and we will use free space variable in that to show the correct information.

Finally, if the drives are not ready then we will show an error message stating the drive is not ready.

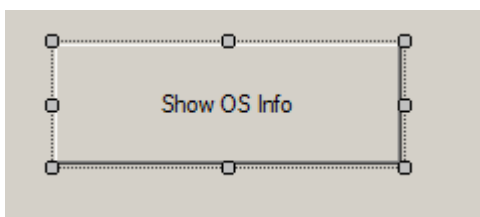
So on this computer I am working right now I have two drives one hard drive and one DVD drive now the hard drive information can be pulled without any problem however I want to see if the program gives an error message for the DVD drive because its empty.



That worked flawlessly.

OS - Operating system Information

Now double click on the Show OS Info button. In our case its button 5.



Inside the button function add the following code.

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click  
  
    Button5.Enabled = False  
  
    TextBox13.Text = System.Environment.MachineName  
    TextBox14.Text = System.Environment.UserName  
    TextBox15.Text = My.Computer.Info.OSFullName
```

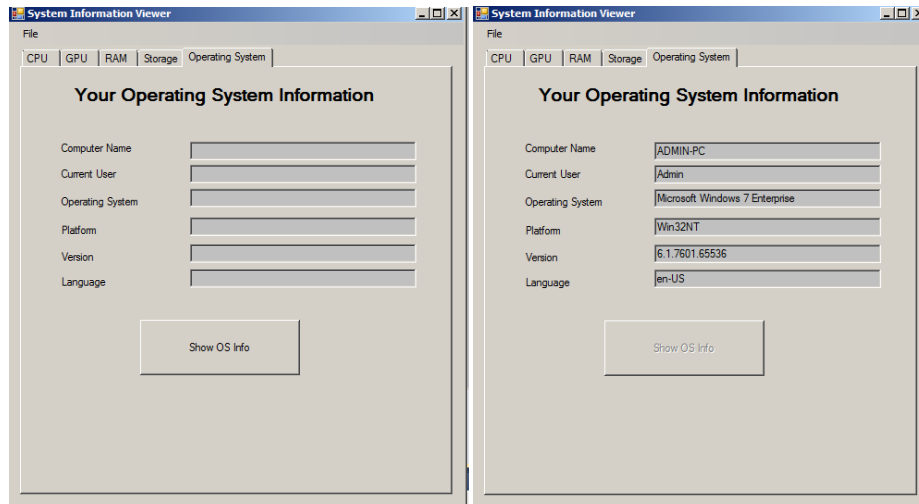
more tutorials on www.mooict.com

```
TextBox16.Text = My.Computer.Info.OSPlatform  
TextBox17.Text = My.Computer.Info.OSVersion  
TextBox18.Text = My.Computer.Info.InstalledUICulture.ToString
```

End Sub

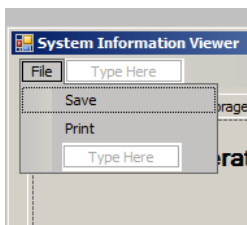
This function is very straight forward because we are not creating any variables for this. We are simply taking the information from my computer and displaying on the screen.

Test the program



Saving information to text file

Now we will code the functionality where this program can save all the information into a text file.



Click on file and then double click on the save option.

```
Private Sub SaveInformationToolStripMenuItem_Click (sender As System.Object, e As System.EventArgs) Handles  
SaveInformationToolStripMenuItem.Click
```

```
saveThis()
```

```
End Sub
```

Visual basic will automatically add an event to the menu strip inside it add saveThis()

SaveThis() is a function we will create shortly. In that function we can have all the text information we want to add to the file. Each time this menu strip will be clicked it will trigger the saveThis() function.

Lets start making our saveThis function. We start with the usual function declaration as you can see below. Its an empty function which we will populate.

```
Private Sub saveThis()
```

more tutorials on www.mooict.com

End Sub

We start adding with our first variable file. Now this will be a stream writer type variable which means its allowed to be written in and it will use the system to write to a specific file.

```
Dim file As System.IO.StreamWriter
```

Once the variable has been declared we now tell it where to save the text file to. In this case the file will saved where the EXE or compiled file is. It's usually in the debug folder which we will show you later on. In this variable you notice we are using the open text file write then the text file name inside the quotations. After the question there is a Boolean which we set to false. Now if it was set to true it will append of edit an older text file so we set it false in case we already created a spec file before and when we create a new one we want to over write it completely. Otherwise it will continue to edit the spec file and you will have loads of different specifications inside one text file.

```
file = My.Computer.FileSystem.OpenTextWriter("spec.txt", False)
```

Since the file variable is an stream write object we can call the write variable to enable writing the file we stated above.

```
file.Write("")
```

Write your first line.

```
file.WriteLine("Welcome to my new text file")
```

Important to note that we opened the text writer before now we need to close it so it can finalise the text file and save it. This needs to be at the bottom of the function always.

```
file.Close()
```

Lets see how that works now.

```
End Sub

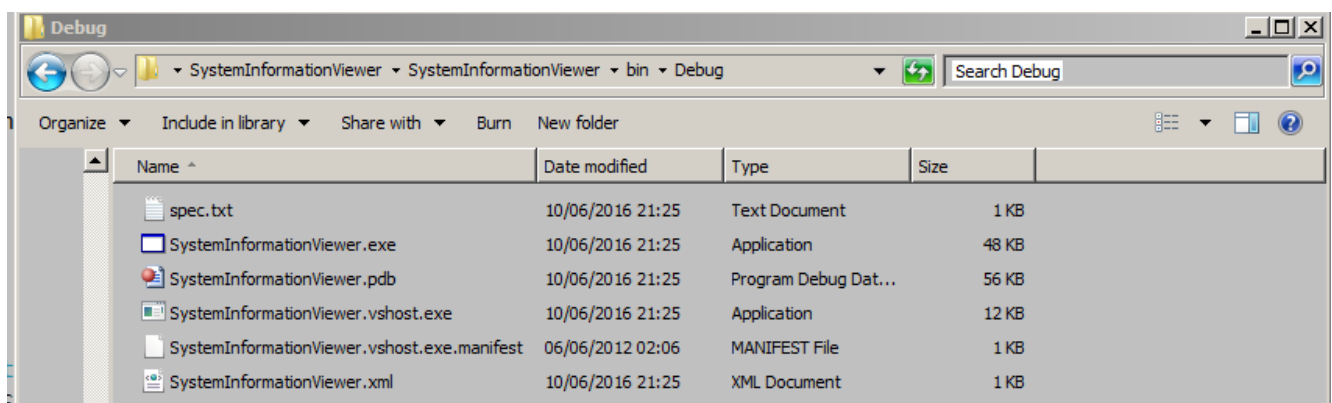
Private Sub saveThis()
    Dim file As System.IO.StreamWriter
    file = My.Computer.FileSystem.OpenTextWriter("spec.txt", False)
    file.Write("")
    file.WriteLine("Welcome to my new text file")

    file.Close()
End Sub

End Class
```

Here is the code.

Now run the program and lets click on save.

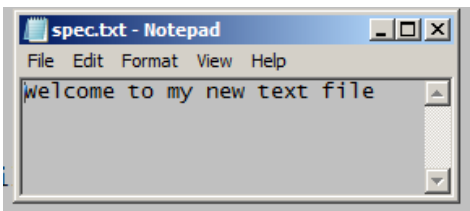


The project folder is inside your documents folder. in my case its inside

\\Documents\\Visual Studio 2010\\Projects\\SystemInformationViewer\\SystemInformationViewer\\bin\\Debug

You can see its created a spec.txt file right next to the EXE file there.

more tutorials on www.mooict.com



Cool right. That's the general idea right now, we click on save its saved the spec.txt file in the back ground and the user can access it.

We need to add the codes for the save this function.

Logic is we collaborate everything in this whole program inside this function. We bring in the CPU, GPU, RAM, OS not the hard drive information. Since the hard drive needs to be individually selected there is no point in saving that information for this exercise.

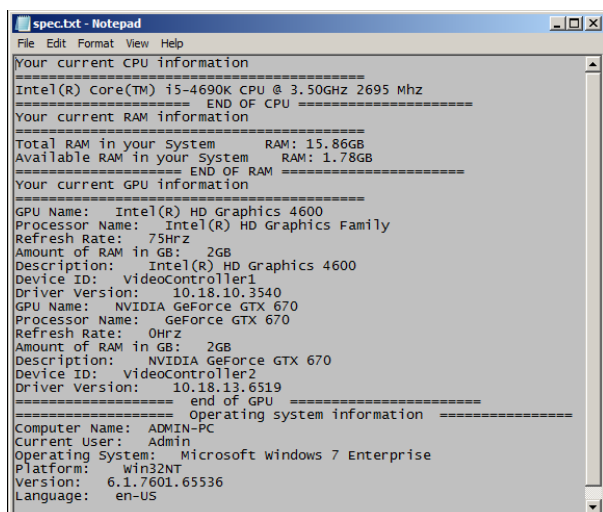
Note - just because I am not covering in this tutorial you can add that later on if you wish.

<pre>Private Sub saveThis() Dim file As System.IO.StreamWriter file = My.Computer.FileSystem.OpenTextFileWriter("spec.txt", False) file.Write("") file.WriteLine("Your current CPU information") file.WriteLine("=====")</pre>	<p>We start the function here as we did before in the exercise. Now we are writing the line to say where the cpu information starts and then we added the equals sign = to state a division.</p>
<pre>'cupU info Dim MyOBJ As Object Dim cpu As Object MyOBJ = GetObject("WinMgmts:").instancesof("Win32_Processor") For Each cpu In MyOBJ file.WriteLine(cpu.Name.ToString + " " + cpu.CurrentClockSpeed.ToString + " Mhz") Next file.WriteLine("===== END OF CPU =====")</pre>	<p>You will notice its similar to the cpu button codes except the file.writeLine part. Instead of showing the result on screen we are writing the information to the spec.txt file this time.</p>
<pre>file.WriteLine("Your current RAM information") file.WriteLine("=====") Dim ramAmount As Double ramAmount = (My.Computer.Info.TotalPhysicalMemory / 1048576) / 1024 file.WriteLine("Total RAM in your System " & "RAM: " & ramAmount.ToString("N") & "GB") Dim ramLeft As Double = (My.Computer.Info.TotalPhysicalMemory - My.Computer.Info.AvailablePhysicalMemory) / 1048576 / 1024 file.WriteLine("Available RAM in your System " & "RAM: " & ramLeft.ToString("N") & "GB") file.WriteLine("===== END OF RAM =====")</pre>	<p>This section starts with the RAM information. Its similar to the RAM button codes except the file.writeline codes. We are writing the total for the system and the available ram for the system in the text file.</p>
<pre>file.WriteLine("Your current GPU information") file.WriteLine("=====") Dim objectQuery As New ObjectQuery("SELECT * FROM Win32_VideoController") Dim objectSearch As New ManagementObjectSearcher(objectQuery)</pre>	<p>This section is for the GPU of the section. Its similar to the GPU information button code and it will simply write the information to the file instead of showing in on the</p>

<pre> For Each memObj As ManagementObject In objectSearch.Get file.WriteLine("GPU Name: " & memObj("Name")) file.WriteLine("Processor Name: " & memObj("VideoProcessor")) file.WriteLine("Refresh Rate: " & Convert.ToInt64(memObj("MaxRefreshRate")).ToString & "Hz") file.WriteLine("Amount of RAM in GB: " & Convert.ToInt64(memObj("AdapterRAM") / 1048576 / 1024).ToString & "GB") file.WriteLine("Description: " & memObj("Description")) file.WriteLine("Device ID: " & memObj("DeviceID")) file.WriteLine("Driver Version: " & memObj("DriverVersion")) Next file.WriteLine("===== end of GPU =====") </pre>	<p>screen. You will notice that we running the same loop as before got the GPU name, processor name, refresh rate, amount of VRAM, Device ID etc.</p> <p>If you look closely we are using an IF statement in this section of the code. Because we want print about all of the GPU present in the system.</p>
<pre> file.WriteLine("===== Operating system information =====") file.WriteLine("Computer Name: " & System.Environment.MachineName) file.WriteLine("Current User: " & System.Environment.UserName) file.WriteLine("Operating System: " & My.Computer.Info.OSFullName) file.WriteLine("Platform: " & My.Computer.Info.OSPlatform) file.WriteLine("Version: " & My.Computer.Info.OSVersion) file.WriteLine("Language: " & My.Computer.Info.InstalledUICulture.ToString) file.WriteLine("End of Operating system Information") </pre>	<p>This section will write the code for OS information. Its the same as OS button code but its writing on the text file.</p>
<pre> file.WriteLine("Thank you for using the system information viewer 1.0") </pre>	<p>Showing gratitude to the user for using this app. Since we are really nice right.</p>
<pre> file.Close() End Sub </pre>	<p>This part we are closing the file before ending the function.</p> <p>Note - do not do any action relating to writing on file after the close command. It will return an error.</p>

All of the code above are for save this function. It's a long code however you need to read it carefully to ensure you are not making any mistakes.

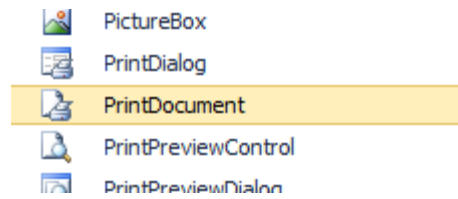
Lets run the program again and click on save see what happens.



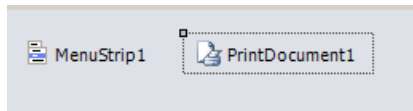
Here is the final spec.txt file which was compiled by the application. Pretty neat.

Print the information

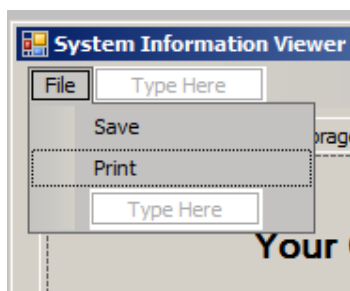
Lets get started with our printing set up. We need to go back to the design view.



drag and drop the print document component from the toolbox to the form.



This is the printDocument1 component added to our form.

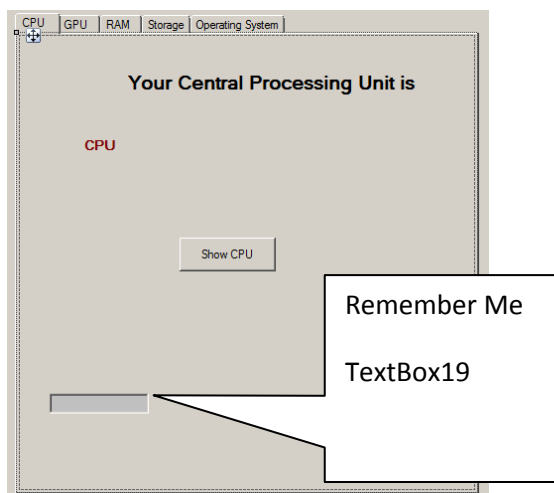


Double click on the print button from the menu strip

```
Private Sub PrintToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles PrintToolStripMenuItem.Click  
  
End Sub
```

Visual studio will add the following code once you have doubled clicked on the button.

Now we will need to run the save this function inside the function and load it into our text box which we hidden earlier. If you do not remember look at the last part of the GUI section.



Lets go back to the code and add the following inside the function we created earlier for the menu strip print button.

```
Private Sub PrintToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
```

more tutorials on www.mooict.com

```

System.EventArgs) Handles PrintToolStripMenuItem.Click
    saveThis()
    For Each s As String In System.IO.File.ReadAllLines("spec.txt")
        TextBox19.AppendText(s + vbNewLine)
    Next
    PrintDocument1.Print()
End Sub

```

First we are running the save this function. This will create the spec.txt file with all the required information.

Then we are running a for loop which will read all the lines for the spec.txt file and load them into the **TextBox19** which is shown above in the screen shot.

In logical terms for each line exists in the spec.txt file we will add it to the text box 19 by using append text method. vbNewLine is a method which will add a new line right after it finds a new line in the text box. For example if we don't use vbNewLine it will like this -

```

Your current CPU information=====Intel(R) Core(TM) i5-4690K
CPU @ 3.50GHz 3501 Mhz===== END OF CPU =====

```

We don't want that we want the information to look like this

```

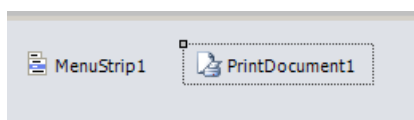
Your current CPU information
=====
Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz 3501 Mhz
===== END OF CPU =====

```

As programmers its our duty to make sure that all of the input and output of an application we create if professional and easy to use / understand by the end user. If the program is not easy to use no one will use it no matter how great the software is, always remember that.

Last we are running the printDocument1.print() function. Which will run the printing action.

Lets get back to the design view and double click on the printDocument1 component.



Yep double click on this one.

```

Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage

End Sub

```

Visual studio will insert the code above automatically once you double clicked on the printDocument1 component.

Add the following code inside the function

```

Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
    Dim font1 As New Font("Arial", 10, FontStyle.Regular)
    e.Graphics.DrawString(TextBox19.Text, font1, Brushes.Black, 100, 100)
End Sub

```


This section is important because we are telling Windows how we want our text file printed.

First we are declaring a new font variable called font1 and we are giving it a setting of Ariel font size 10 and font size regular.

Secondly We will call the graphics draw string to pull the text box 19's content with the font1 setting we created before and giving it a colour black. 100, 100 stands for the height and width of the text file we want. We don't want each line to go over the line of that.

Now try the program once again and click on print. Remember if you have a printer set up on your system it will send the request to print straight away.

Here is the printed copy below - I've used on a different system than mines.

```
Your current CPU information
=====
Intel(R) Core(TM) i5-3340 CPU @ 3.10GHz 2077 Mhz
===== END OF CPU =====
Your current RAM information
=====
Total RAM in your System   RAM: 3.70GB
Available RAM in your System  RAM: 2.28GB
===== END OF RAM =====
Your current GPU information
=====
GPU Name: Intel(R) HD Graphics
Processor Name: Intel(R) HD Graphics Family
Refresh Rate: 120Hz
Amount of RAM in GB: 2GB
Description: Intel(R) HD Graphics
Device ID: VideoController1
Driver Version: 10.18.10.3277
===== end of GPU =====
===== Operating system information =====
Computer Name: SC21373
Current User: aali
Operating System: Microsoft Windows 7 Enterprise
Platform: Win32NT
Version: 6.1.7601.65536
Language: en-US
```

End

OK, you have made to the end of this tutorial. Its been a long one and it took me a while to put it together. If you have any questions do contact me through <http://www.mooict.com/contact/> and I will try my best to help you through. Remember to check out more tutorials on the site.