Interfacing to Visual Basic in Visual Studio 2010

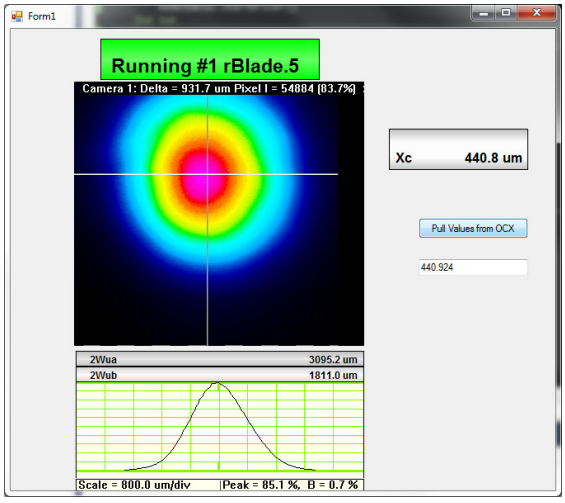
Covers: Interfacing DataRay Camera and Slit Scan Profilers to Visual Basic in Visual Studio 2010 using the DataRay OCX.

Start in the standard software:

- *As Administrator*, install the DataRay software which came with your product.
- Attach the profiler product. Allow the drivers to install.
- Open the DataRay software and select your profiler in the **Device** pull-down menu.
- *Learn to use your product in the DataRay software*. Then close the software.

Add Visual Studio 2010: We do not claim to be VB or Visual Studio 'experts', however we are able to create new VB projects in Visual Studio that can control DataRay products. Install Visual Studio 2010 on your computer (earlier versions should work, but exact details will change). Download the example from the DataRay website:

- **Cameras**: Download & unzip: http://www.dataray.com/UserFiles/file/DataRayInterfaceToVBVS2010.zip
- **BeamMap2**, **Beam'R2**, **ColliMate**: TBA



Build and run example: This example should build and run with no errors. Not working? Email support@dataray.com or call (303)543-8235 with:

- Device name and serial number
- DataRay, Windows & Visual Studio versions which you are using. Only Visual Studio 2010 and later are fully supported. The DataRay OCX still works in VS2006 and VS2008, but we are only able to provide limited support.

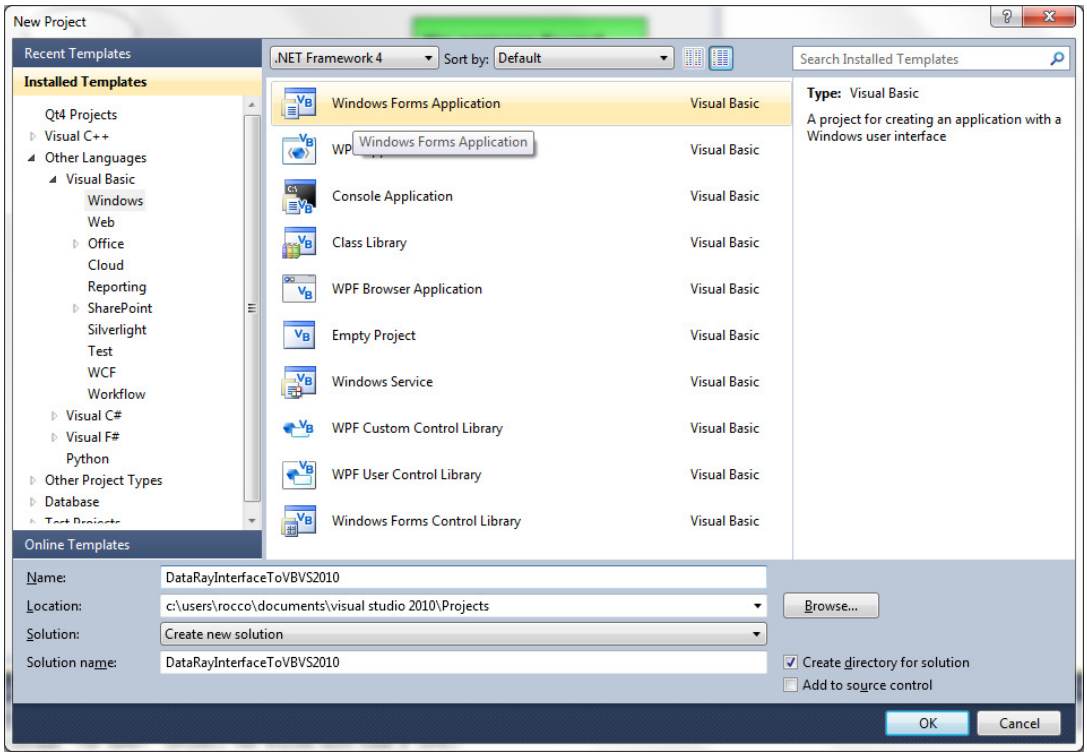
Overview of OCX: Your interfacing code communicates with DataRay products through the DataRay OCX. The OCX is an ActiveX component that can be accessed from a variety of Windows based environments. The OCX is automatically generated and registered with the Windows operating system upon installing the DataRay software. Once initialized, the OCX is always running. This means that the camera is still running, even while editing GUI elements in Visual Studio. Do not be alarmed if DataRay OCX GUI elements are active while your program is not running. This is the expected behavior.

Some important notes:

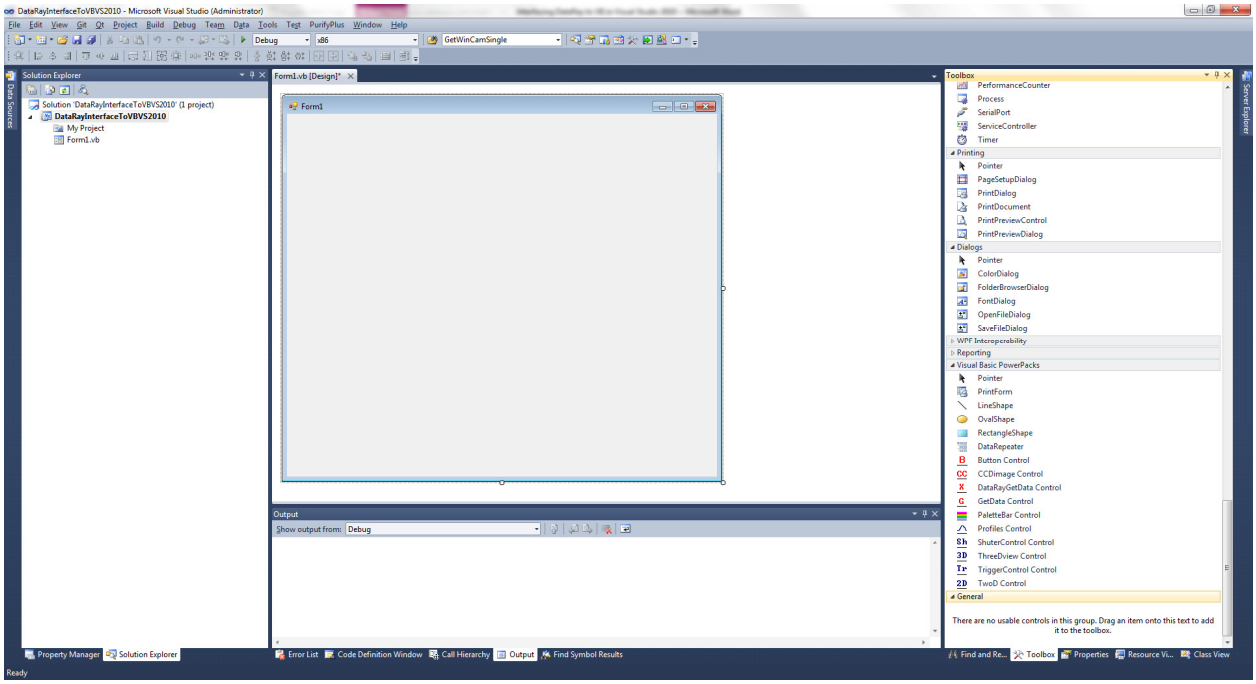
- Read through this entire document.
- Some prior experience with Visual Basic, Windows MFC programming, and Visual Studio is required.
- The OCX is only functional as part of a GUI-based program.
- In the **Design View** of VS2010, elements may appear as white boxes or as the actual GUI element they represent.

Tutorial:

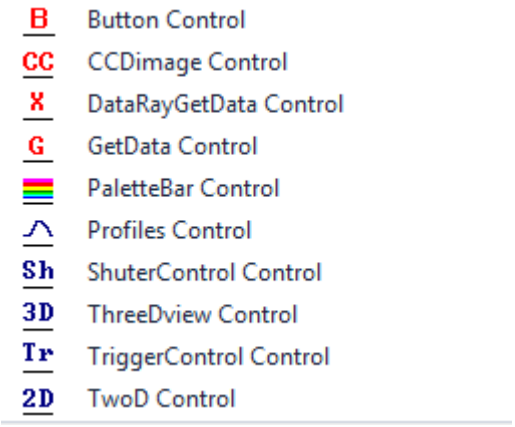
We will show you step-by-step how the example program was created. First, create a new **Windows Forms Application** in VS2010:



The empty project should look like this:



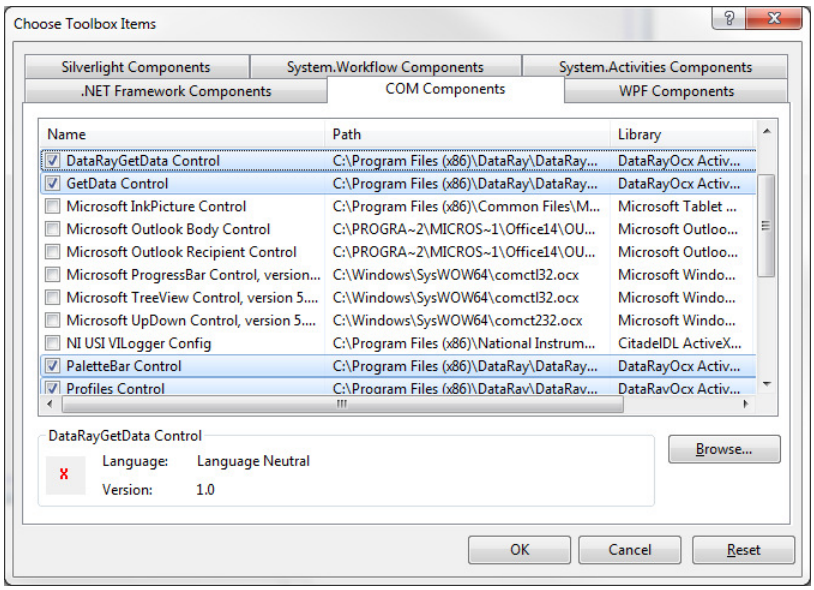
Open the **Toolbox**. You should see the following DataRay components:



If these components aren't visible, complete the following steps:

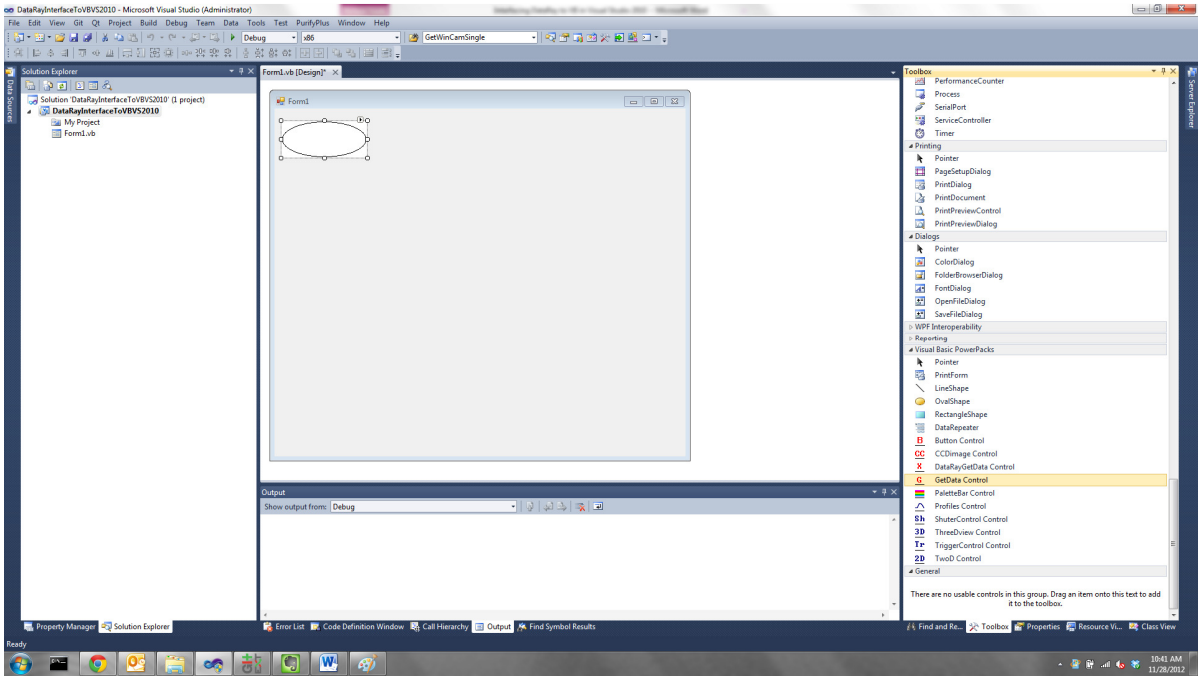
1) Select **Tools->Choose Toolbox Items**
2) Select **COM Components** tab
3) Select **Browse…**
4) Navigate to the your DataRay install directory
5) Select **DataRayOcx.ocx**

The **Choose Toolbox Items** should now show the DataRay components:

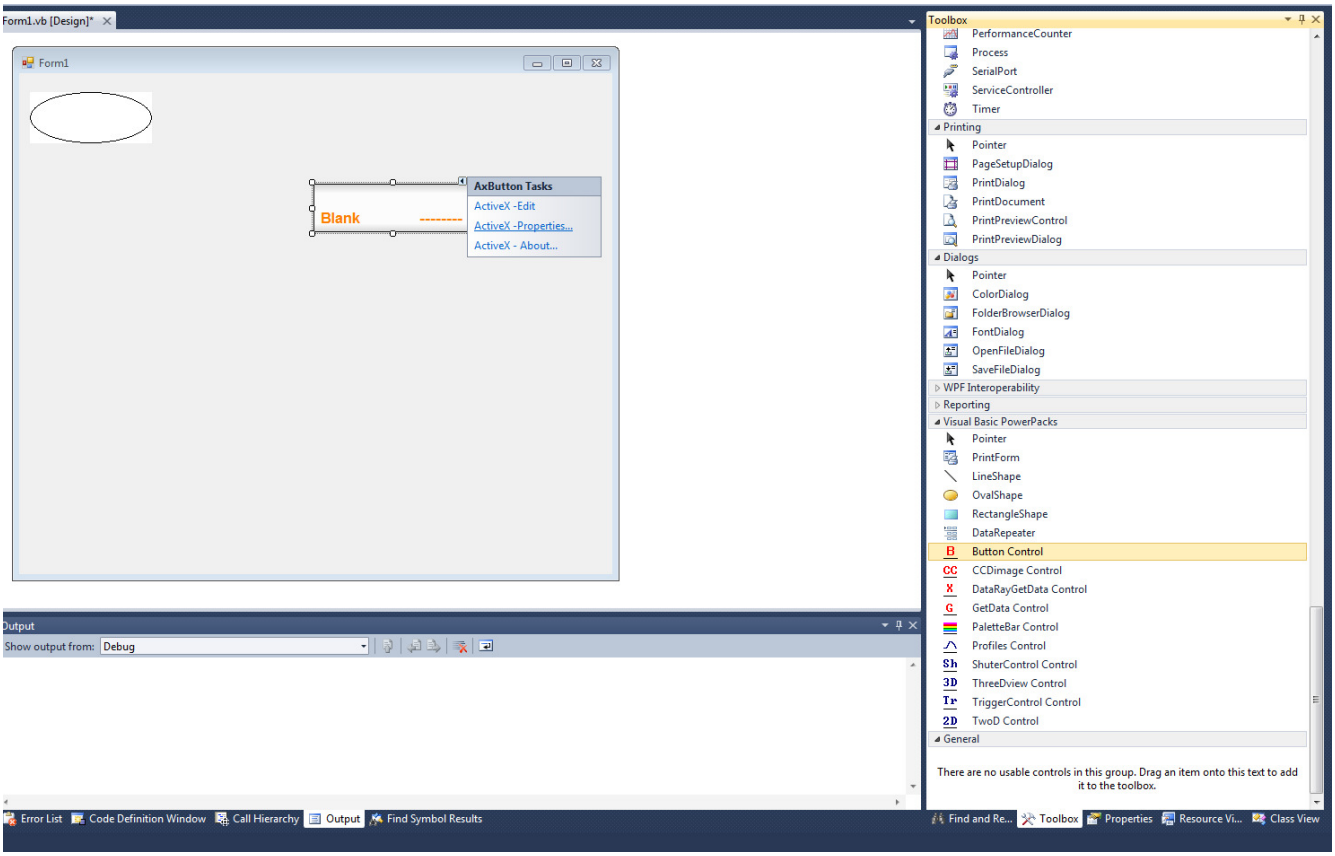


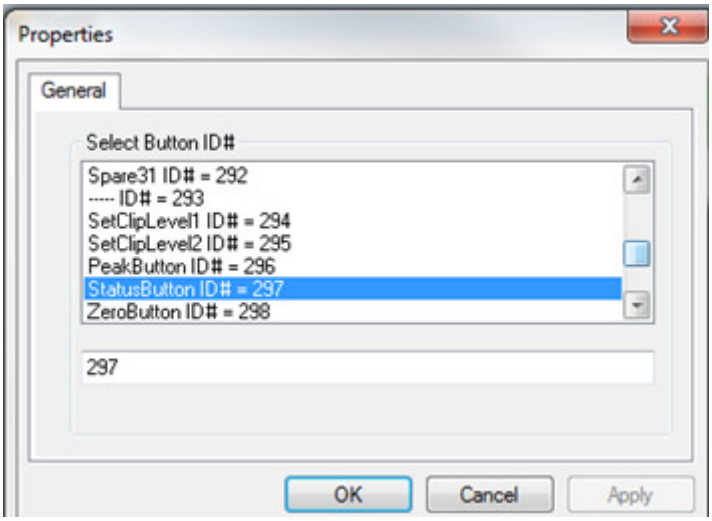
Your toolbox should now be populated with the DataRay Controls.

Now we can begin building the actual program. First drag a **GetData Control** (**not DataRayGetData Control**) onto the dialog box. This is the only OCX control class required for interfacing to DataRay cameras.

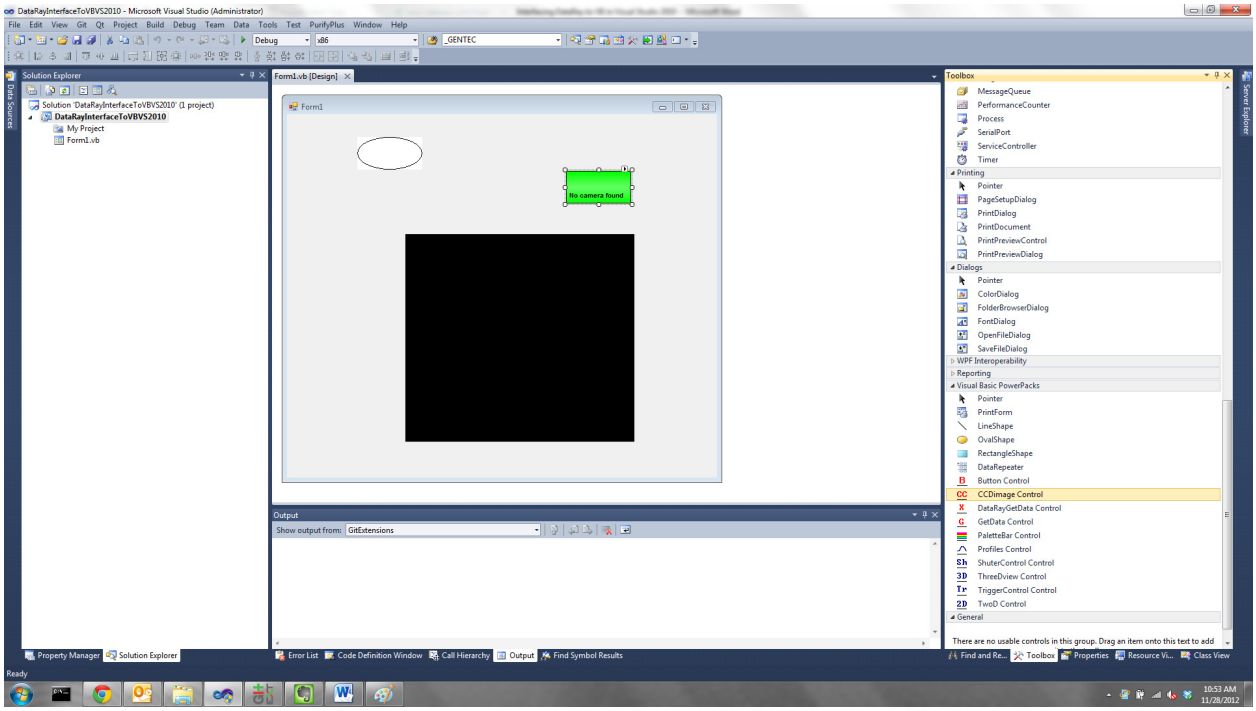


We'll also create a **Ready** button and a display for the two-dimensional camera display (known as a **CCDImage**). Drag a **Button Control** to the dialog box. Click on the small rightwards arrow on the top of button. Select **ActiveX-Properties…** Change the **Button ID#** to **297**.



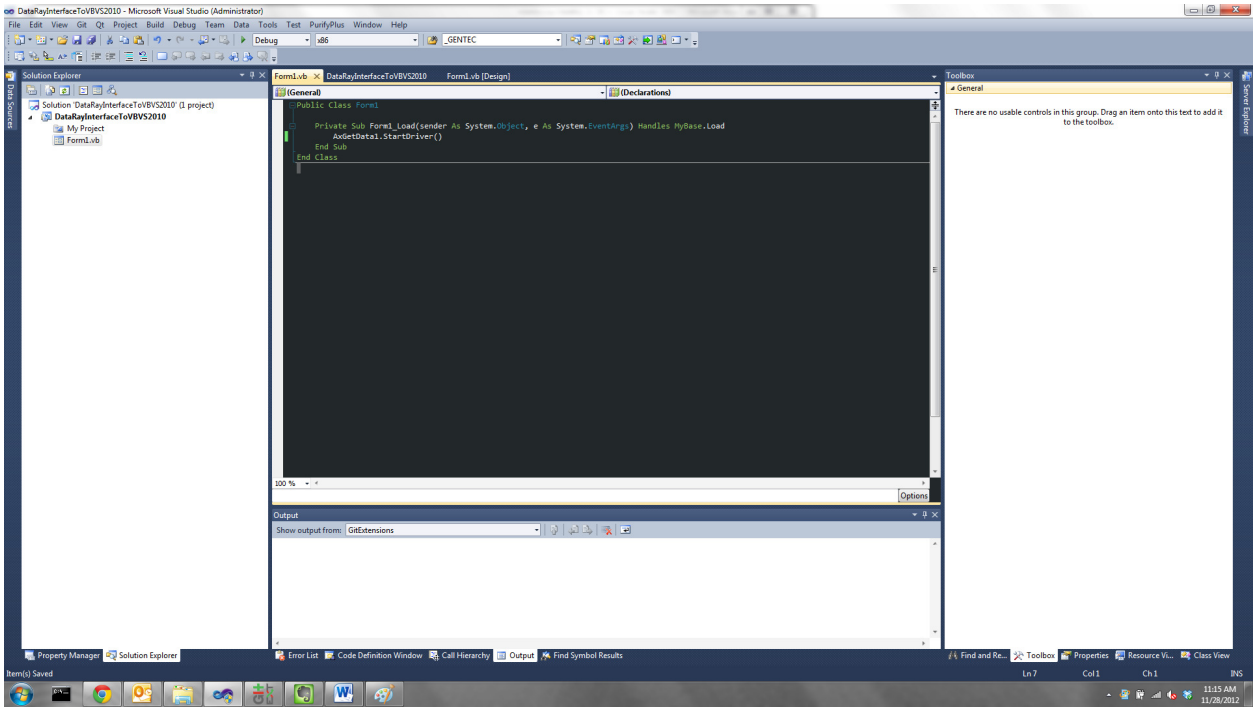


Now drag a **CCDimage Control** onto the form. This completes the basic layout of the dialog box.

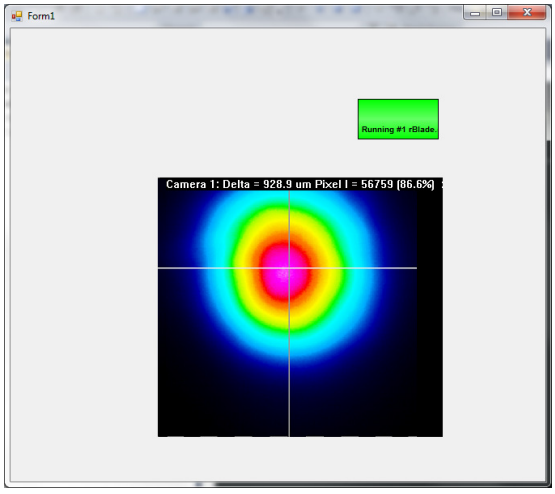


Now we need to add some code to the template. Open up the code for your form. Add the following line to the Load function of your form:

```
AxGetData1.StartDriver()
```

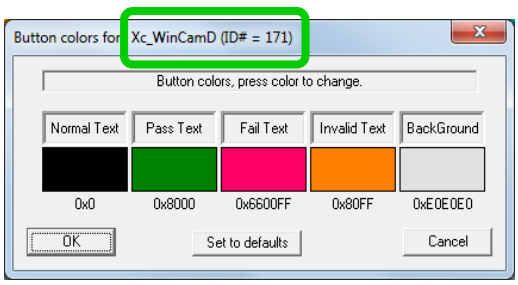


Now you are ready to build the project. Build and run your project. The green button is exactly the same **Ready** button as in the DataRay software. Click on the button to begin running your camera. You should see something similar to this (depending on your laser source):



Congratulations, you are now interfacing with your DataRay device.

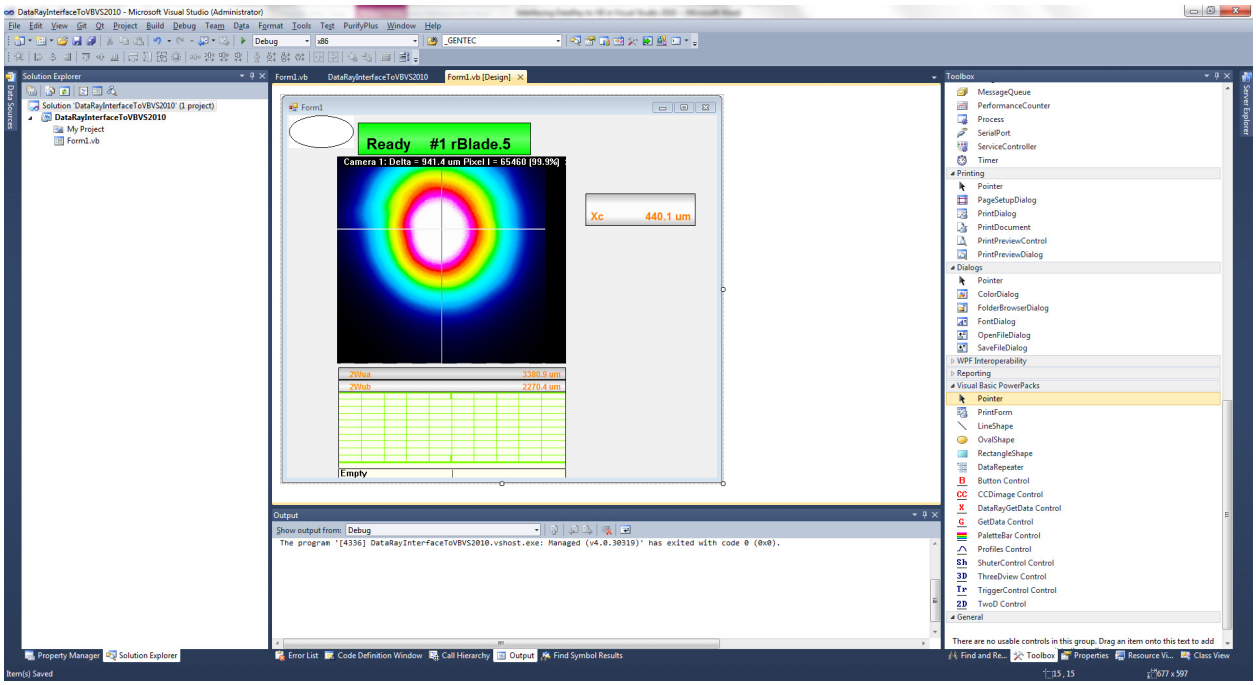
Now, we will add a few more objects. For this example, we want to display the X-axis profile, and the calculated X-axis centroid position (**Xc**).



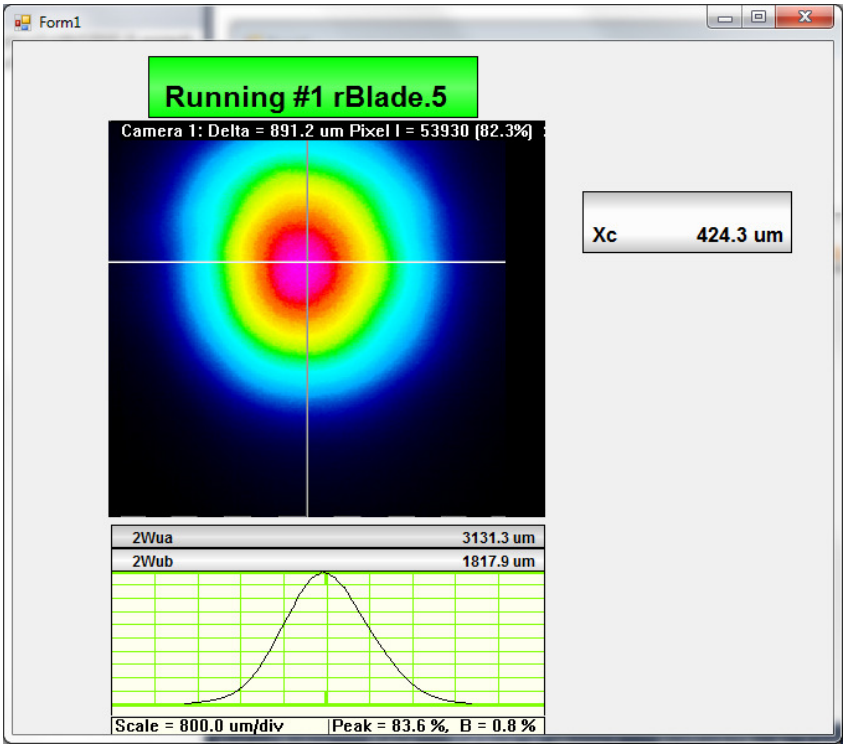
Add a **Profile Controls** and a **Button Controls**.

In order to find the correct **Button ID#** to use for the buttons, you need to:

1) Close VS2010 and open the DataRay software
2) Right click on any button, to see the dialog on the right
3) Note the current **Name** and **ID#** for this result at the top of the dialog
4) Repeat for all the results of interest. Close the DataRay Software



Following these instructions, you'll be able to tell that to see **Xc** you need to change the **Button ID#** to **171**. For the profile, change the **Profile ID#** to **22**. Build and run your program and you should see the following:



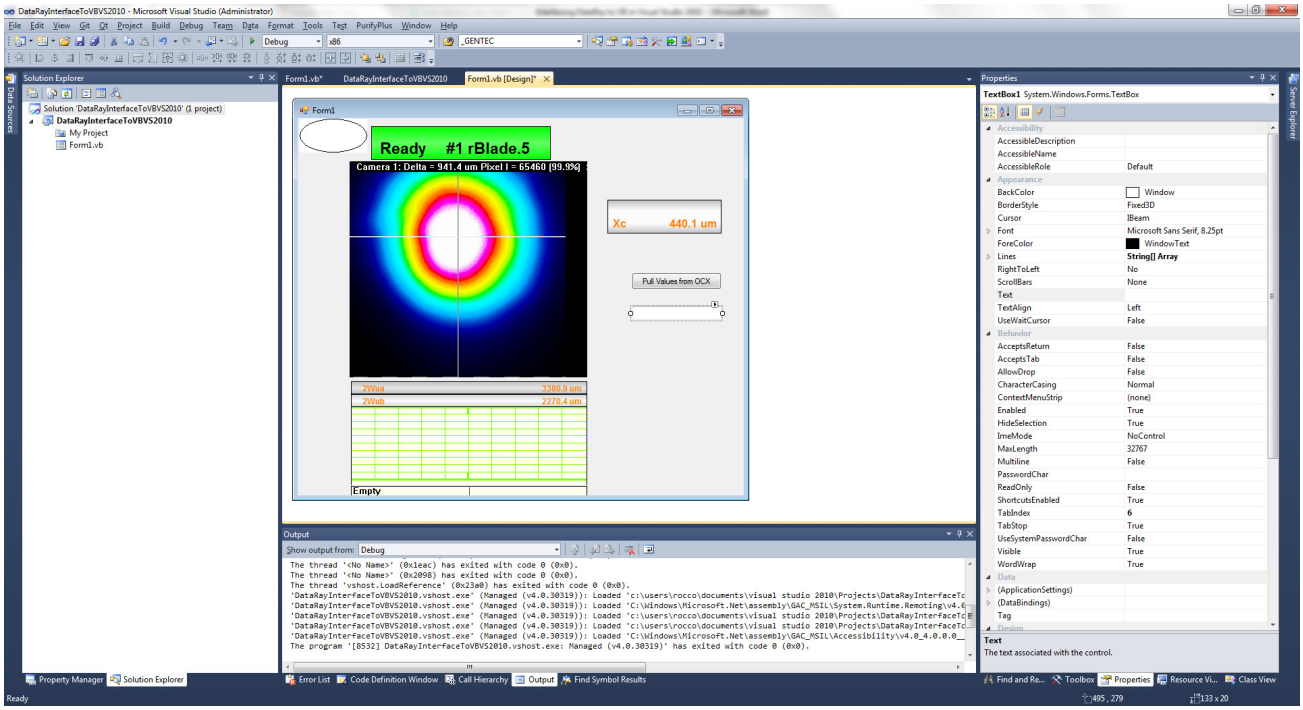
A complete list of Profile IDs can be found here: http://www.dataray.com/UserFiles/file/ProfilesEnum.pdf

A complete list of Button IDs can be found here: http://www.dataray.com/UserFiles/file/IndexToTestParametersEnum.pdf

Finally, we will programmatically extract data from the OCX. To extract single values, you can use the method **GetOcxResult** from the **GetData Control** class where the argument for the method is the same number used to ID the button:

```
Xc = AxGetData1.GetOcxResult(171)
```

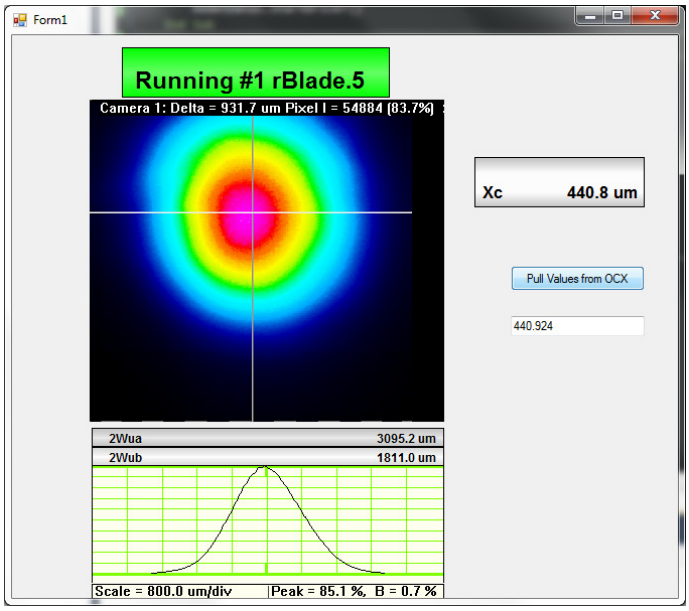
To illustrate this point, we added a Visual Basic standard **Button** and **TextBox** to the form:



In the function for handling clicking the button, we've added the following code:

```
'Get single value from OCX example
Dim Xc As Single
Xc = AxGetData1.GetOcxResult(171)
'Display Result in TextBox
TextBox1.Text = Xc
```

Now, clicking the box in runtime will pull the **Xc** value from the OCX.



The OCX also supports sending arrays of data via variants:

```
'Get variant Example
Dim Var As Object
Var = AxProfiles1.GetProfileDataAsVariant()
```

This code will read the data from the Profile Control as a variant. This is the preferred method for reading large amounts of data from the OCX.

This completes the tutorial! **Problems/Questions?** As above, contact us with the information listed above.