DataRay Inc.

Interfacing to C++ in Visual Studio 2010

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

Start in the standard software:

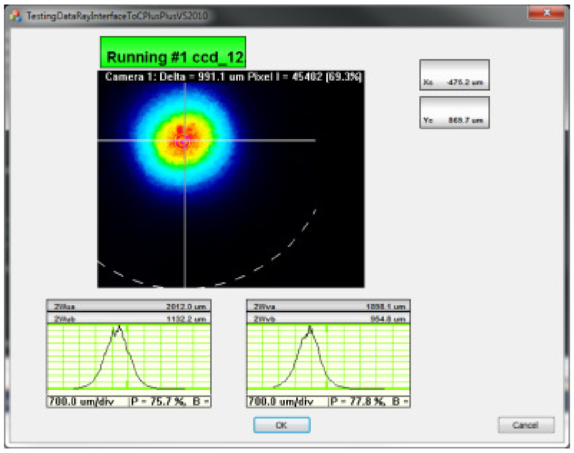
- *As Administrator*, install the DataRay software which came with your product.
- Attach the profiler product. Allow the drivers to install.
- Open the DataRay software and select your profiler in the **Device** pull-down menu.
- *Learn to use your product in the DataRay software*. Then close the software.

Add Visual Studio 2010: We do not claim to be Visual Studio 'experts', however we are able to create new 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/TestingDataRayInterfaceToCPlusPlusVS2010.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.



Camera 1: Delta = 991.1 um Pixel I = 45402 (69.3%)

Xc -475.2 um

Yc 869.7 um

2Wua 2012.0 um
2Wub 1132.2 um
700.0 um/div P = 75.7 %, B =

2Wva 1898.1 um
2Wvb 954.8 um
700.0 um/div P = 77.8 %, B =

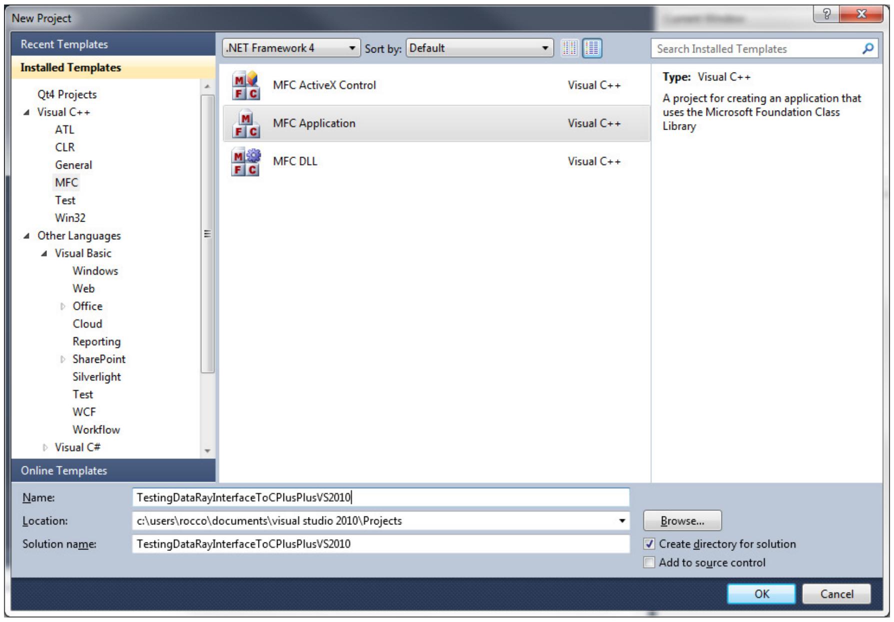
Overview of OCX: Your interfacing code communicates with DataRay products through the DataRay OCX. The OCX is an ActiveX component that can be accessed from a variety of Windows based environments. The OCX is automatically generated and registered with the Windows operating system upon installing the 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 C++, Windows MFC programming, and Visual Studio is required.
- The OCX is only functional as part of a GUI-based program.
- In the **Resource 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 MFC Application in VS2010:



Type: Visual C++

A project for creating an application that uses the Microsoft Foundation Class Library

Name: TestingDataRayInterfaceToCPlusPlusVS2010

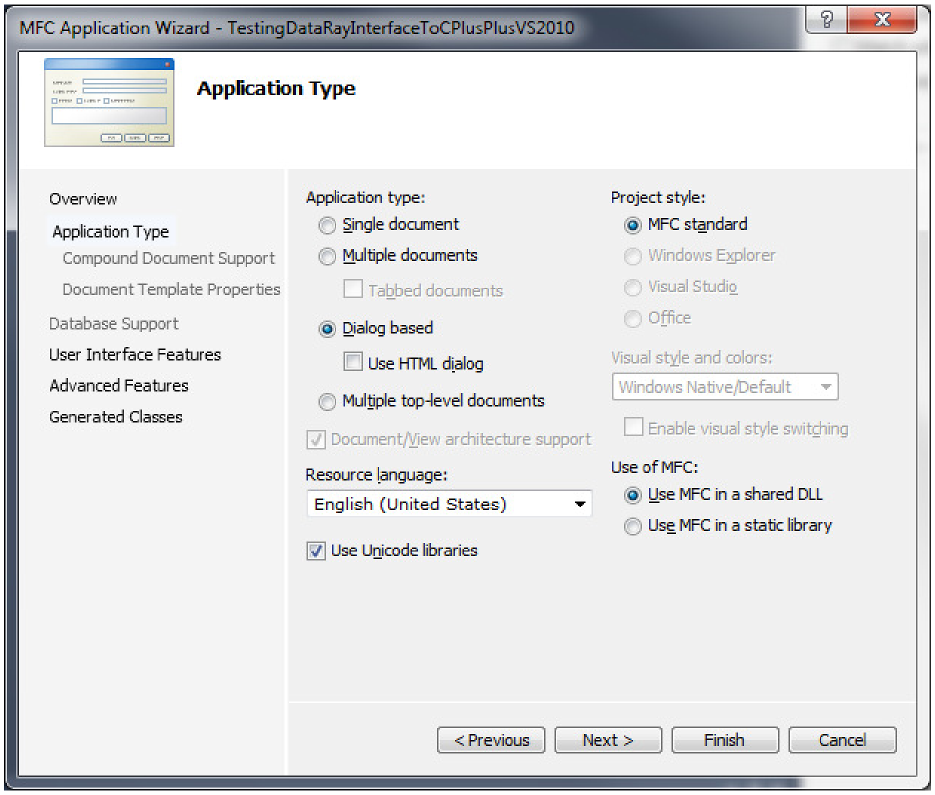
Location: c:\users\rocco\documents\visual studio 2010\Projects

Solution name: TestingDataRayInterfaceToCPlusPlusVS2010

☑ Create directory for solution

☐ Add to source control

Select **Dialog based** to simplify things (This isn't actually a requirement, but will allow the example to be simple):



Application type:

○ Single document

○ Multiple documents

☐ Tabbed documents

● Dialog based

☐ Use HTML dialog

○ Multiple top-level documents

☑ Document/View architecture support

Resource language: English (United States)

☑ Use Unicode libraries

Project style:

● MFC standard

○ Windows Explorer

○ Visual Studio

○ Office

Visual style and colors: Windows Native/Default

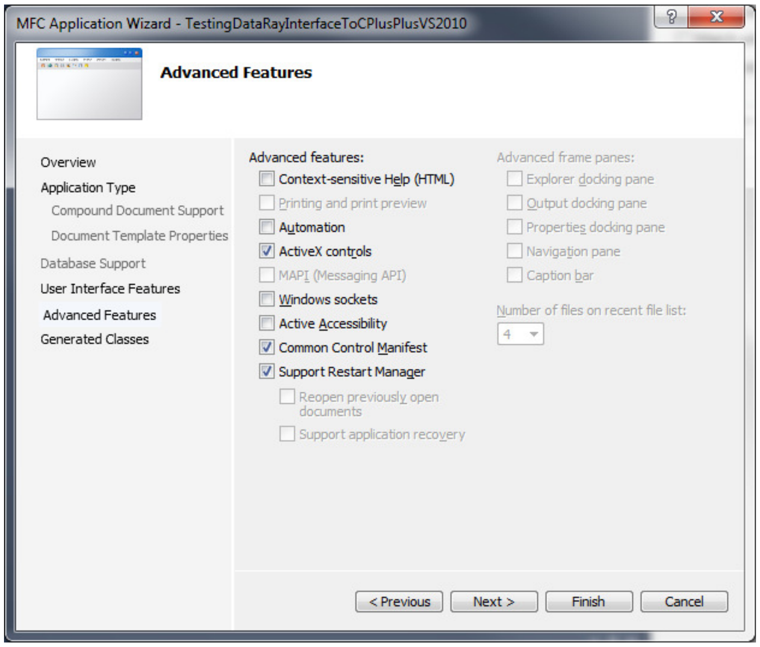
☐ Enable visual style switching

Use of MFC:

● Use MFC in a shared DLL

○ Use MFC in a static library

The default values in VS2010 are sufficient for this project. Under the **Advanced Features** tab, verify that **ActiveX controls** are enabled.


MFC Application Wizard - TestingDataRayInterfaceToCPlusPlusVS2010

Advanced Features

Advanced features:

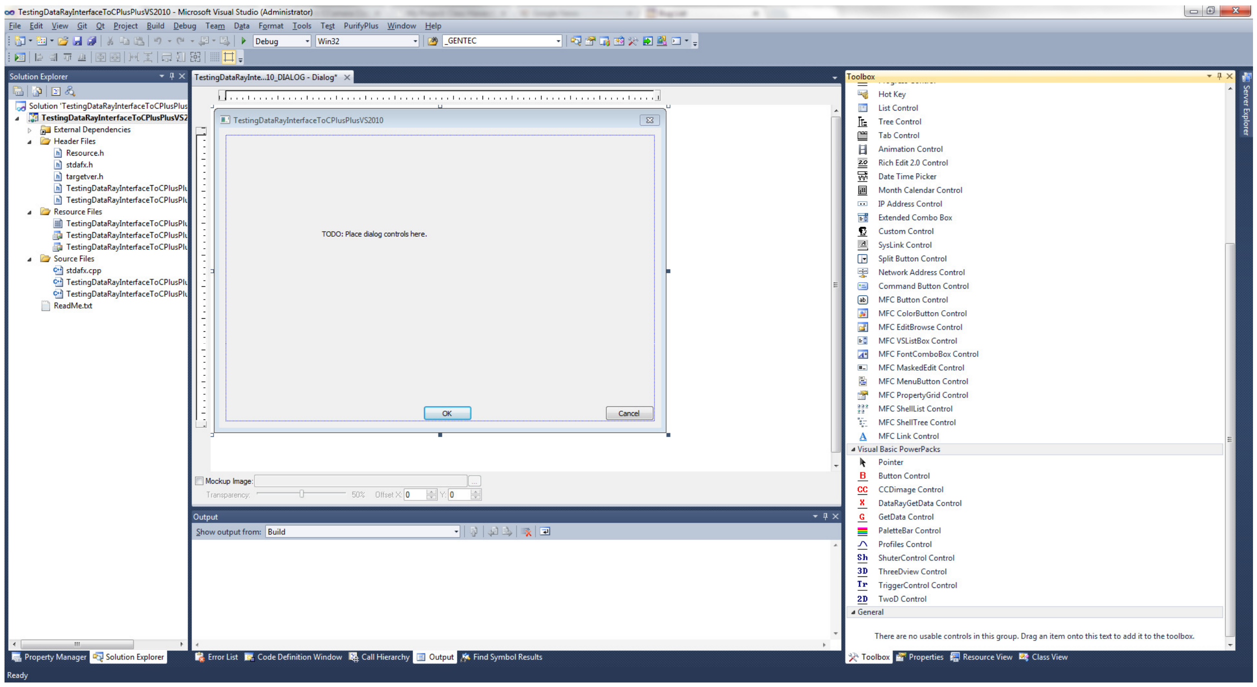
- [ ] Context-sensitive Help (HTML)
- [ ] Printing and print preview
- [ ] Automation
- [✓] ActiveX controls
- [ ] MAPI (Messaging API)
- [ ] Windows sockets
- [ ] Active Accessibility
- [✓] Common Control Manifest
- [✓] Support Restart Manager
  - [ ] Reopen previously open documents
  - [ ] Support application recovery

Advanced frame panes:

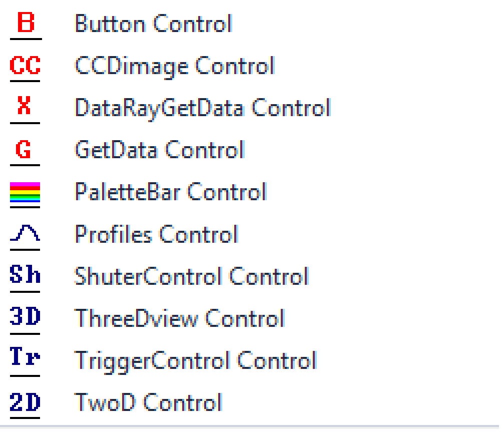
- [ ] Explorer docking pane
- [ ] Output docking pane
- [ ] Properties docking pane
- [ ] Navigation pane
- [ ] Caption bar

Number of files on recent file list: 4

The empty project should look like this:



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

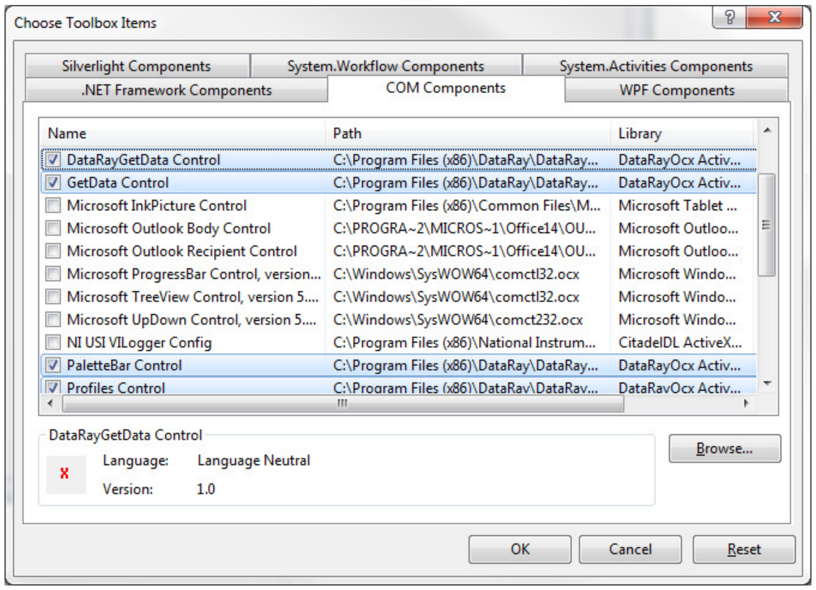


Button Control
CCDimage Control
DataRayGetData Control
GetData Control
PaletteBar Control
Profiles Control
ShuterControl Control
ThreeDview Control
TriggerControl Control
TwoD Control

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:



Choose Toolbox Items

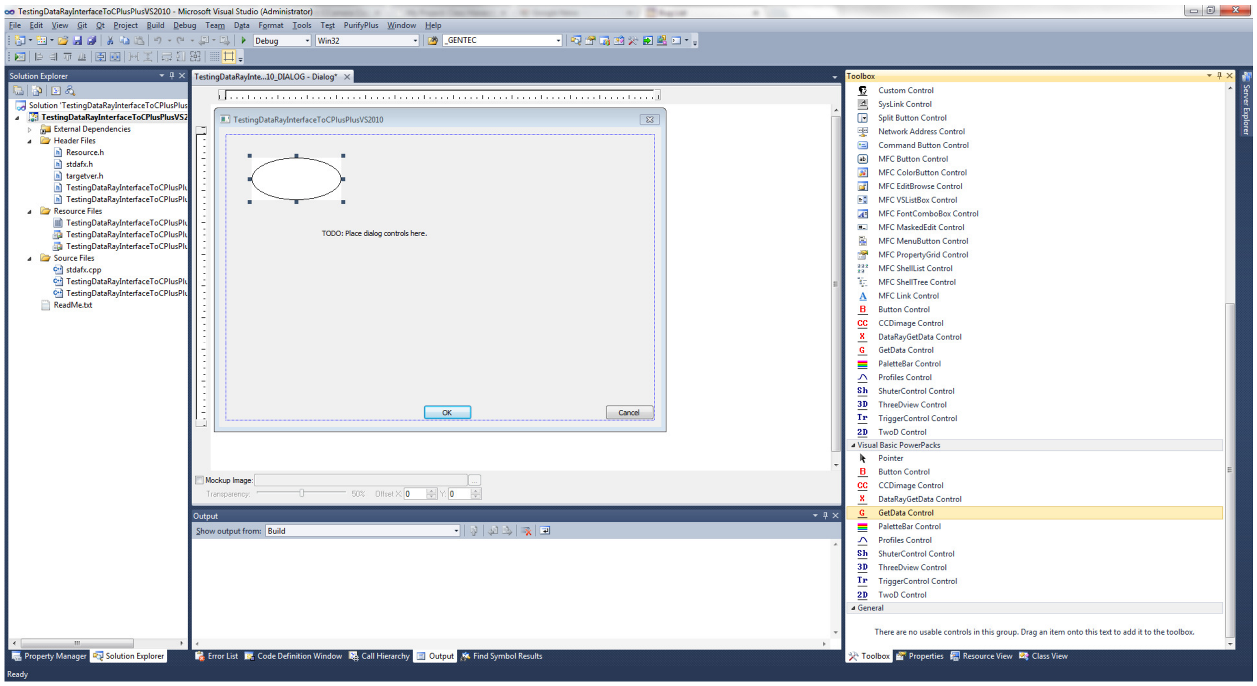
Silverlight Components
System.Workflow Components
System.Activities Components
.NET Framework Components
COM Components
WPF Components

Name	Path	Library
☑ DataRayGetData Control	C:\Program Files (x86)\DataRay\DataRay...	DataRayOcx Activ...
☑ GetData Control	C:\Program Files (x86)\DataRay\DataRay...	DataRayOcx Activ...
☐ Microsoft InkPicture Control	C:\Program Files (x86)\Common Files\M...	Microsoft Tablet ...
☐ Microsoft Outlook Body Control	C:\PROGRA~2\MICROS~1\Office14\OU...	Microsoft Outloo...
☐ Microsoft Outlook Recipient Control	C:\PROGRA~2\MICROS~1\Office14\OU...	Microsoft Outloo...
☐ Microsoft ProgressBar Control, version...	C:\Windows\SysWOW64\comctl32.ocx	Microsoft Windo...
☐ Microsoft TreeView Control, version 5....	C:\Windows\SysWOW64\comctl32.ocx	Microsoft Windo...
☐ Microsoft UpDown Control, version 5....	C:\Windows\SysWOW64\comct232.ocx	Microsoft Windo...
☐ NI USI VILogger Config	C:\Program Files (x86)\National Instrum...	CitadelDL ActiveX...
☑ PaletteBar Control	C:\Program Files (x86)\DataRay\DataRay...	DataRayOcx Activ...
☑ Profiles Control	C:\Program Files (x86)\DataRay\DataRay...	DataRayOcx Activ...

DataRayGetData Control
Language: Language Neutral
Version: 1.0

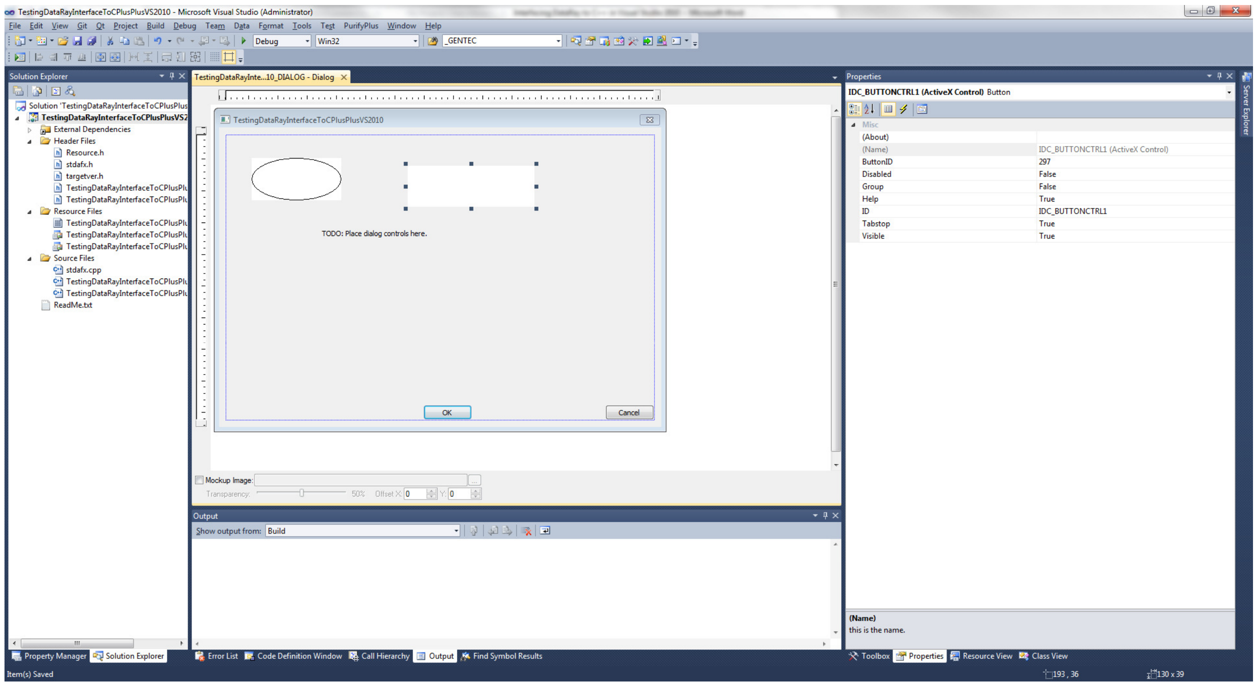
Your toolbox should now be populated with the DataRay Controls.

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. Right click on the button. Select **Edit Control**. Change the **ButtonID** to **297**.

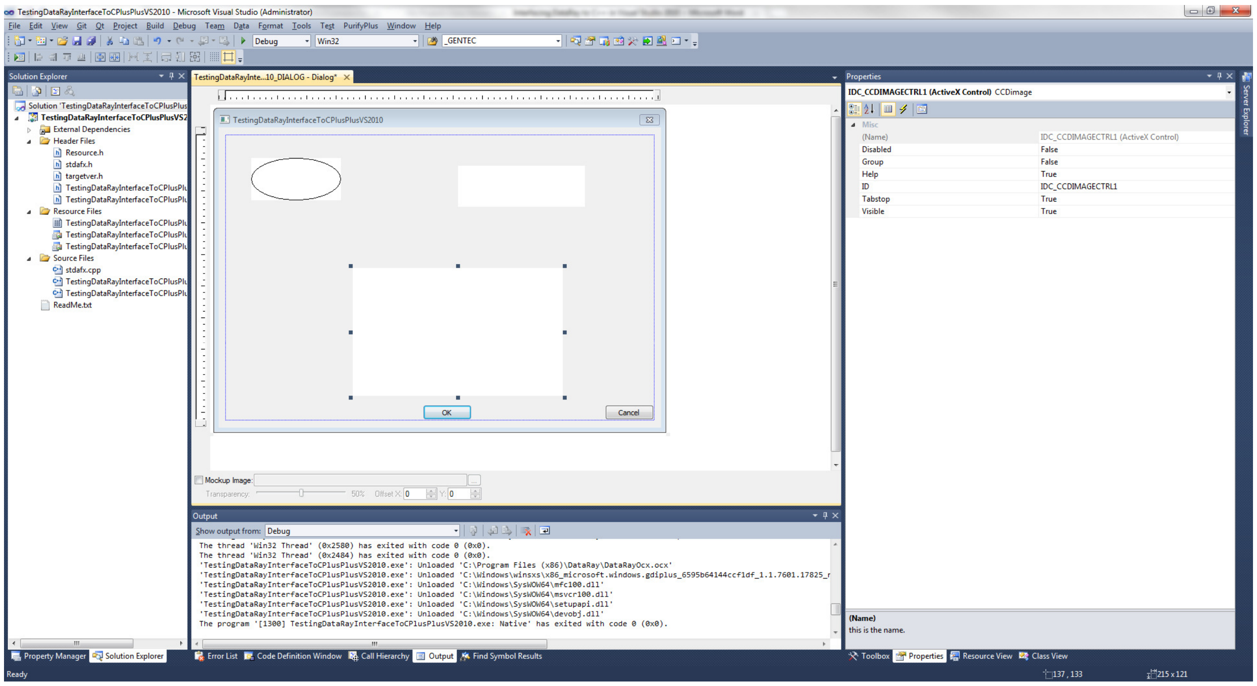

TestingDataRayInterfaceToCPlusPlusVS2010 - Microsoft Visual Studio (Administrator)
Solution Explorer
TestingDataRayInte...10_DIALOG - Dialog
TODO: Place dialog controls here.
OK
Cancel
Properties
Output
Item(s) Saved

IDC_BUTTONCTRL1 (ActiveX Control) Button

(About)	
(Name)	IDC_BUTTONCTRL1 (ActiveX Control)
ButtonID	297
Disabled	False
Group	False
Help	True
ID	IDC_BUTTONCTRL1
Tabstop	True
Visible	True

(Name)
this is the name.

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

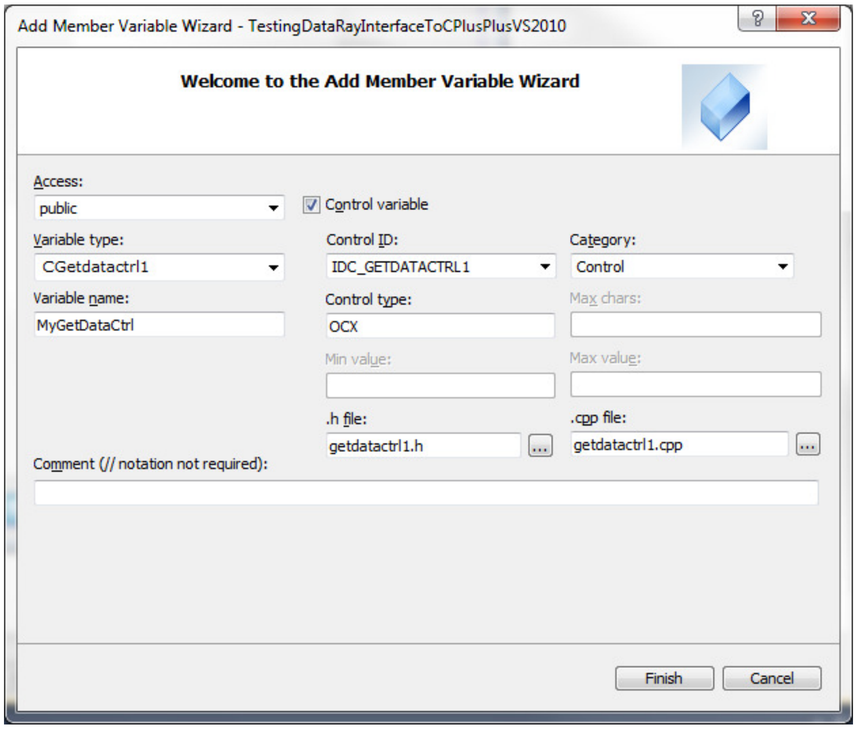

TestingDataRayInterfaceToCPlusPlusVS2010 - Microsoft Visual Studio (Administrator)
Solution Explorer
TestingDataRayInte...10_DIALOG - Dialog*
Properties
Output

IDC_CCDIMAGECTRL1 (ActiveX Control)	CCDimage
(Name)	IDC_CCDIMAGECTRL1 (ActiveX Control)
Disabled	False
Group	False
Help	True
ID	IDC_CCDIMAGECTRL1
Tabstop	True
Visible	True

Show output from: Debug

```
The thread 'Win32 Thread' (0x2580) has exited with code 0 (0x0).
The thread 'Win32 Thread' (0x2484) has exited with code 0 (0x0).
'TestingDataRayInterfaceToCPlusPlusVS2010.exe': Unloaded 'C:\Program Files (x86)\DataRay\DataRayOcx.ocx'
'TestingDataRayInterfaceToCPlusPlusVS2010.exe': Unloaded 'C:\Windows\winsxs\x86_microsoft.windows.gdiplus_6595b64144ccf1df_1.1.7601.17825_r
'TestingDataRayInterfaceToCPlusPlusVS2010.exe': Unloaded 'C:\Windows\SysWOW64\mfc100.dll'
'TestingDataRayInterfaceToCPlusPlusVS2010.exe': Unloaded 'C:\Windows\SysWOW64\msvcr100.dll'
'TestingDataRayInterfaceToCPlusPlusVS2010.exe': Unloaded 'C:\Windows\SysWOW64\setupapi.dll'
'TestingDataRayInterfaceToCPlusPlusVS2010.exe': Unloaded 'C:\Windows\SysWOW64\devobj.dll'
The program '[1300] TestingDataRayInterfaceToCPlusPlusVS2010.exe: Native' has exited with code 0 (0x0).
```

Now we need to add some code to the template. Right-click on the **GetData Control** and select **Add Variable**. Name the variable and make sure **Control variable** is checked.


Add Member Variable Wizard - TestingDataRayInterfaceToCPlusPlusVS2010
Welcome to the Add Member Variable Wizard
Finish
Cancel

Access: public
[✓] Control variable
Variable type: CGetdatactrl1
Control ID: IDC_GETDATACTRL1
Category: Control
Variable name: MyGetDataCtrl
Control type: OCX
Max chars:
Min value:
Max value:
.h file: getdatactrl1.h
.cpp file: getdatactrl1.cpp
Comment (// notation not required):

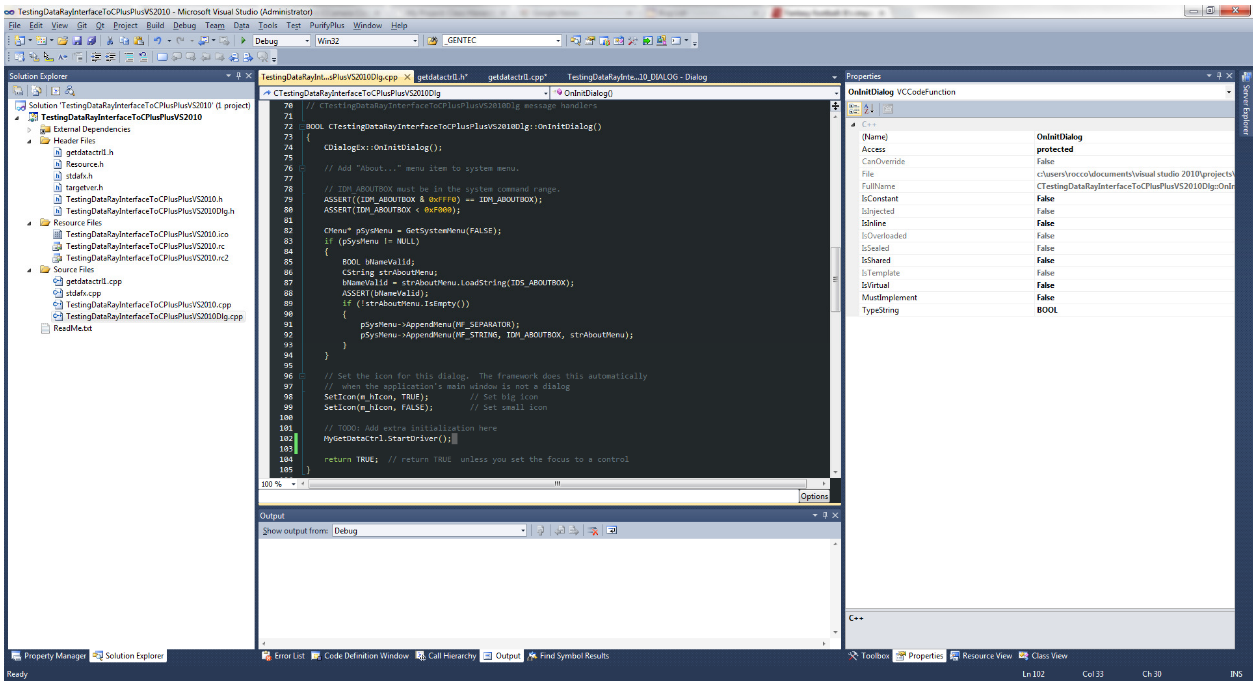
This creates a member control object in your dialog class named **MyGetDataCtrl**. The files **getdatactrl1.h** and **getdatactrl1.cpp** are automatically generated and do not need to be modified. This object contains the method **StartDriver()** that needs to be called to initialize the camera.

In your initialize dialog function

```
BOOL CTestingDataRayInterfaceToCPlusPlusVS2010Dlg::OnInitDialog()
```

Add the following line:

```
MyGetDataCtrl.StartDriver();
```

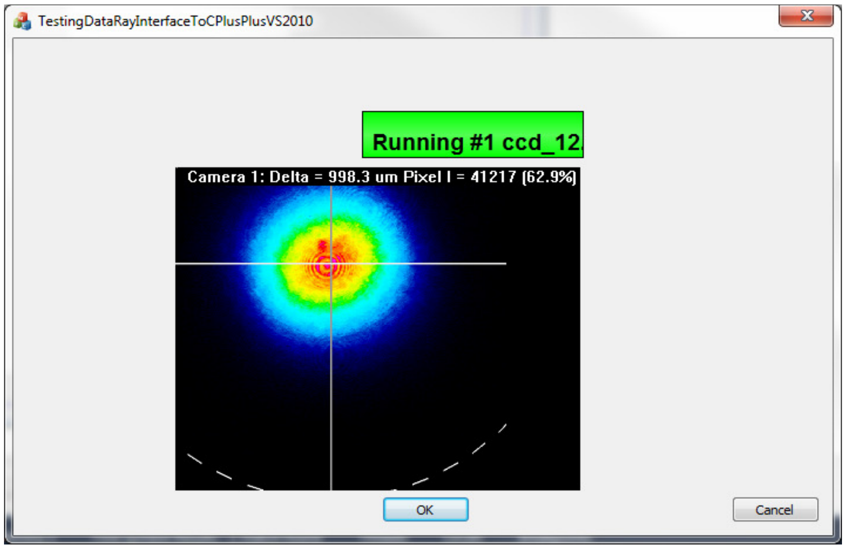

TestingDataRayInterfaceToCPlusPlusVS2010 - Microsoft Visual Studio (Administrator)
Solution Explorer
Properties
Output
Show output from: Debug
Ready
Ln 102
Col 33
Ch 30
INS

```
// CTestingDataRayInterfaceToCPlusPlusVS2010Dlg message handlers

BOOL CTestingDataRayInterfaceToCPlusPlusVS2010Dlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    // Add "About..." menu item to system menu.

    // IDM_ABOUTBOX must be in the system command range.
    ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
    ASSERT(IDM_ABOUTBOX < 0xF000);

    CMenu* pSysMenu = GetSystemMenu(FALSE);
    if (pSysMenu != NULL)
    {
        BOOL bNameValid;
        CString strAboutMenu;
        bNameValid = strAboutMenu.LoadString(IDS_ABOUTBOX);
        ASSERT(bNameValid);
        if (!strAboutMenu.IsEmpty())
        {
            pSysMenu->AppendMenu(MF_SEPARATOR);
            pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
        }
    }

    // Set the icon for this dialog.  The framework does this automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);         // Set big icon
    SetIcon(m_hIcon, FALSE);        // Set small icon

    // TODO: Add extra initialization here
    MyGetDataCtrl.StartDriver();

    return TRUE;  // return TRUE  unless you set the focus to a control
}
```

OnInitDialog VCCodeFunction

(Name)	OnInitDialog
Access	protected
CanOverride	False
File	c:\users\rocco\documents\visual studio 2010\projects\
FullName	CTestingDataRayInterfaceToCPlusPlusVS2010Dlg::OnIn
IsConstant	False
IsInjected	False
IsInline	False
IsOverloaded	False
IsSealed	False
IsShared	False
IsTemplate	False
IsVirtual	False
MustImplement	False
TypeString	BOOL

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):


TestingDataRayInterfaceToCPlusPlusVS2010
Running #1 ccd_12
Camera 1: Delta = 998.3 um Pixel I = 41217 (62.9%)
OK
Cancel

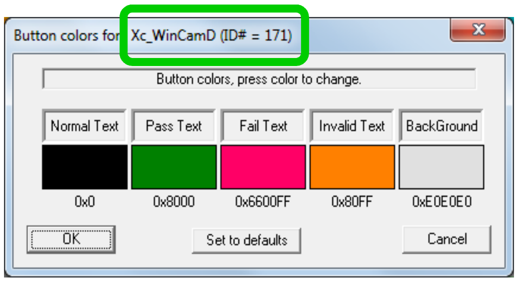
Congratulations, you are now interfacing with your DataRay device.

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

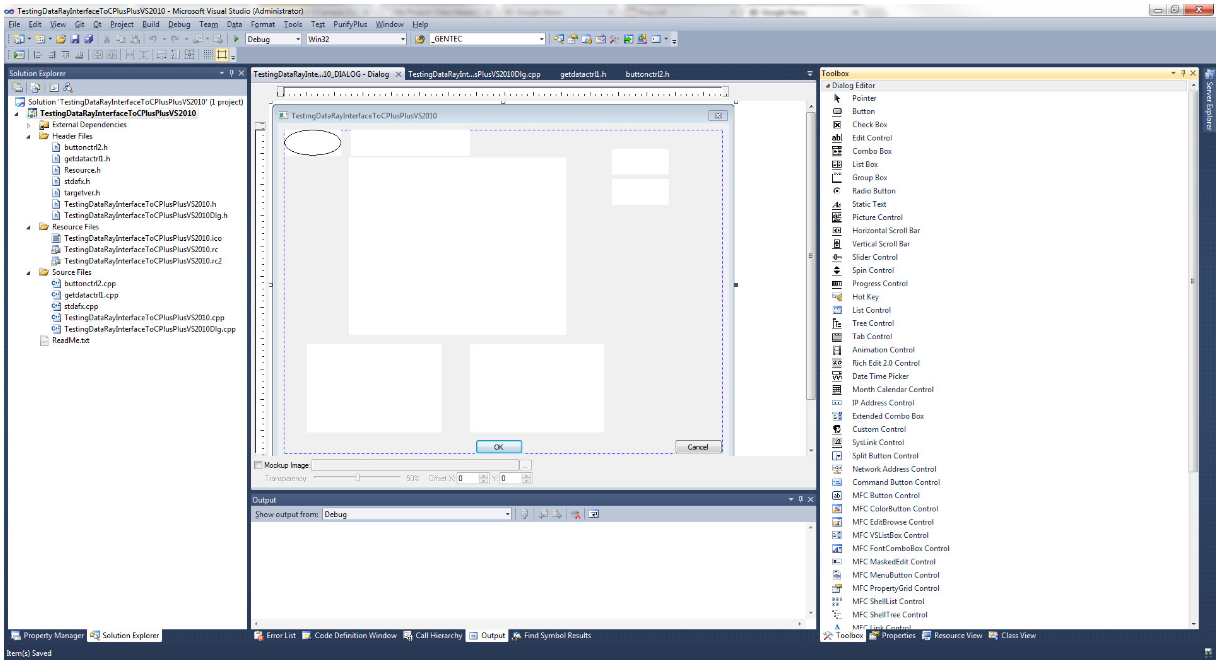
Add two **Profile Controls** and two **Button Controls**.

In order to find the correct **ButtonID** 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


Button colors for Xc_WinCamD (ID# = 171)
Button colors, press color to change.
OK
Set to defaults
Cancel

Normal Text	Pass Text	Fail Text	Invalid Text	BackGround
⬛	🟩	🟥	🟧	⬜
0x0	0x8000	0x6600FF	0x80FF	0xE0E0E0

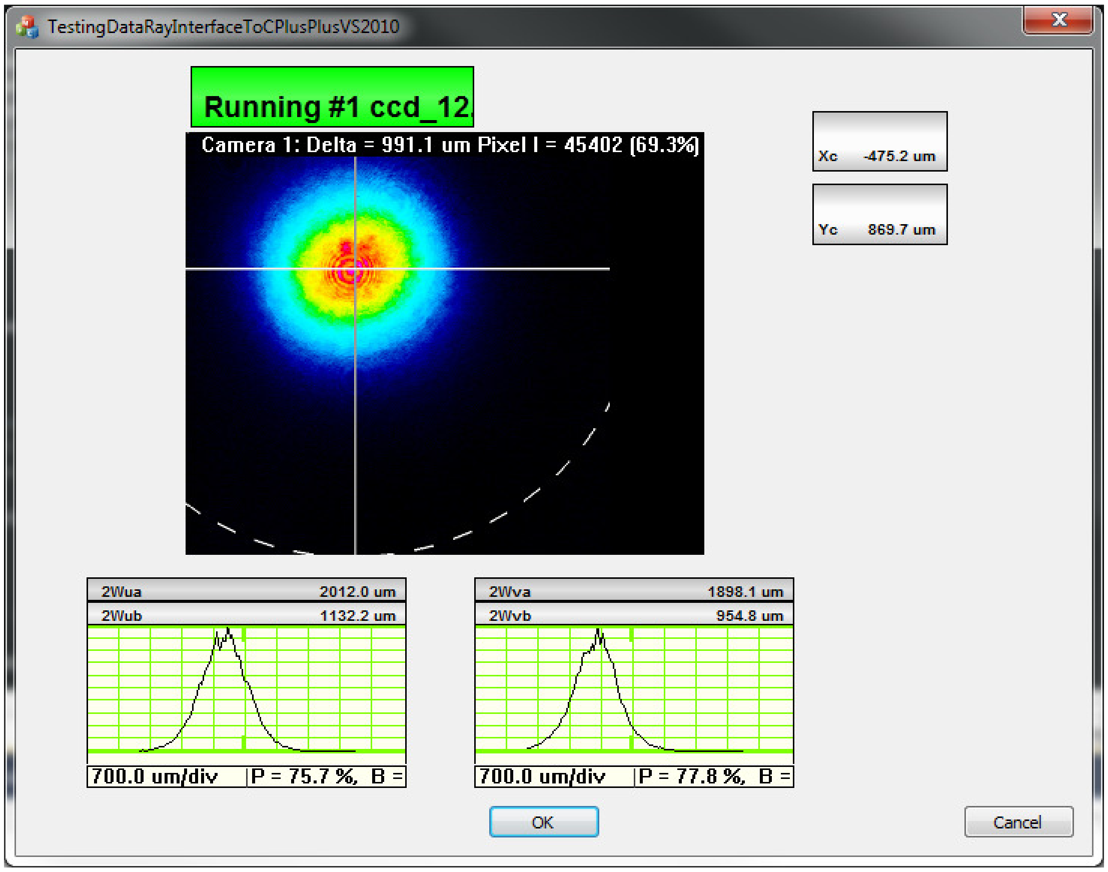

TestingDataRayInterfaceToCPlusPlusVS2010 - Microsoft Visual Studio (Administrator)
Solution Explorer
Solution 'TestingDataRayInterfaceToCPlusPlusVS2010' (1 project)
TestingDataRayInterfaceToCPlusPlusVS2010
External Dependencies
Header Files
buttonctrl2.h
getdatactrl1.h
Resource.h
stdafx.h
targetver.h
TestingDataRayInterfaceToCPlusPlusVS2010.h
TestingDataRayInterfaceToCPlusPlusVS2010Dlg.h
Resource Files
TestingDataRayInterfaceToCPlusPlusVS2010.ico
TestingDataRayInterfaceToCPlusPlusVS2010.rc
TestingDataRayInterfaceToCPlusPlusVS2010.rc2
Source Files
buttonctrl2.cpp
getdatactrl1.cpp
stdafx.cpp
TestingDataRayInterfaceToCPlusPlusVS2010.cpp
TestingDataRayInterfaceToCPlusPlusVS2010Dlg.cpp
ReadMe.txt
TestingDataRayInte...10_DIALOG - Dialog
TestingDataRayInterfaceToCPlusPlusVS2010
OK
Cancel
Output
Show output from: Debug
Toolbox
Dialog Editor
Item(s) Saved

Following these instructions, you'll be able to tell that to see **Xc** and **Yc**, we should change the **ButtonID**s to **171** and **172**. For the profiles, change the **ProfileID**s to **22** and **23**.

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

Build and run your program and you should see the following:



```
TestingDataRayInterfaceToCPlusPlusVS2010

Running #1 ccd_12
Camera 1: Delta = 991.1 um Pixel I = 45402 [69.3%]

Xc    -475.2 um
Yc     869.7 um

2Wua          2012.0 um      2Wva          1898.1 um
2Wub          1132.2 um      2Wvb           954.8 um
700.0 um/div |P = 75.7 %,  B =   700.0 um/div |P = 77.8 %,  B =
```

Finally, we will programmatically extract data from the OCX.

There are two main methods. One way is to create an instance of the control class (same steps as earlier for the **GetData Control**). For example, you could create a variable called **MyXcButton** for the Button with ID **171**. Then, the following line of code will give you the value from the button:

```
double Xc=MyXcButton.GetParameter();
```

You can also query the **GetData Control** directly for parameters:

```
double Xc_FromOCXResult=MyGetDataCtrl.GetOcxResult(171);
```

where the argument for the **GetOcxResult** method is the same number used to ID the button.

The OCX also supports sending arrays of data via variants:

```
VARIANT MyXVar=MyXProfile.GetProfileDataAsVariant();
```

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.