DataRay Inc.
Advancing the Technology of Laser Beam Analysis

Application Note

Interfacing to C++ in Visual Studios 2013

Covers: Interfacing DataRay Camera and Slit Scan Profilers to C++ in Visual Studios 2013 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 13:

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 2013 on your computer (earlier versions should work, but exact details will change). Download the example from the DataRay website:

- Cameras: Download and unzip: TestingDataRayInterfaceToCPlusPlusVS2013.zip
- BeamMap2, BeamR2, ColliMate: TBA

Build and run example:

The example should build and run with no errors (see Fig. 1). Not working? Email support@dataray.com or call **866-946-2263 x2002** with:

- Device name and serial number
- DataRay, Windows and Visual Studio versions which you are using. Only Visual Studio 2013 and later are fully supported. The DataRay OCX still works in VS2006, VS2008, and VS2010, 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 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 VS2013, elements may appear as white boxes or as the actual GUI element they represent.

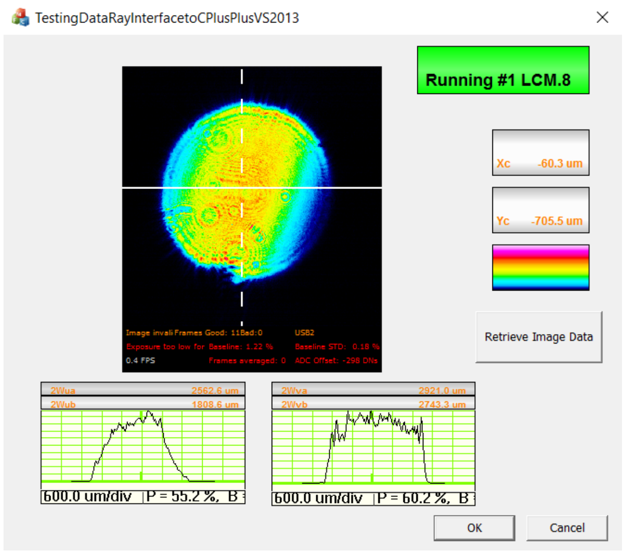
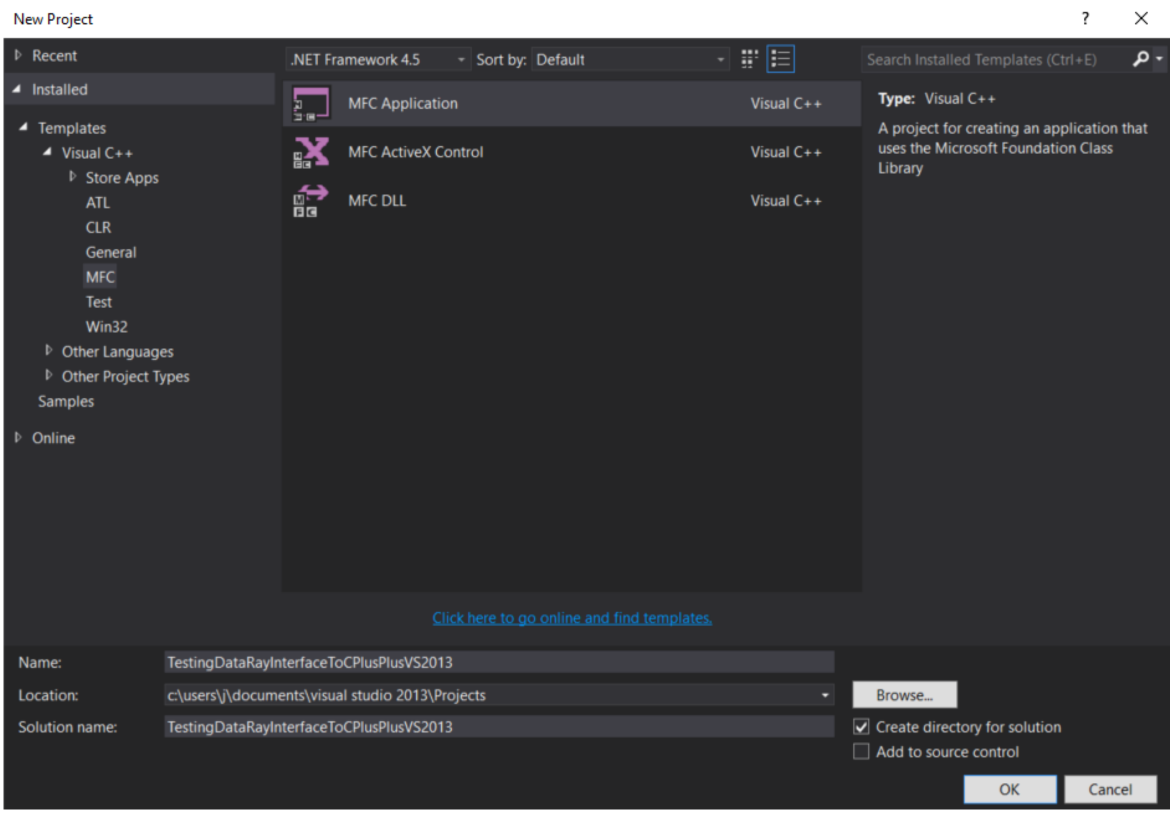


Figure 1: Example application built and running.

Tutorial:

We will show you step-by-step how the example program was created. Follow the instructions included in the series of figures listed below.



New Project

- Recent
- Installed
  - Templates
    - Visual C++
      - Store Apps
      - ATL
      - CLR
      - General
      - MFC
      - Test
      - Win32
    - Other Languages
    - Other Project Types
    - Samples
- Online

.NET Framework 4.5 Sort by: Default

Search Installed Templates (Ctrl+E)

MFC Application	Visual C++
MFC ActiveX Control	Visual C++
MFC DLL	Visual C++

Type: Visual C++

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

Click here to go online and find templates.

Name:	TestingDataRayInterfaceToCPlusPlusVS2013
Location:	c:\users\j\documents\visual studio 2013\Projects
Solution name:	TestingDataRayInterfaceToCPlusPlusVS2013

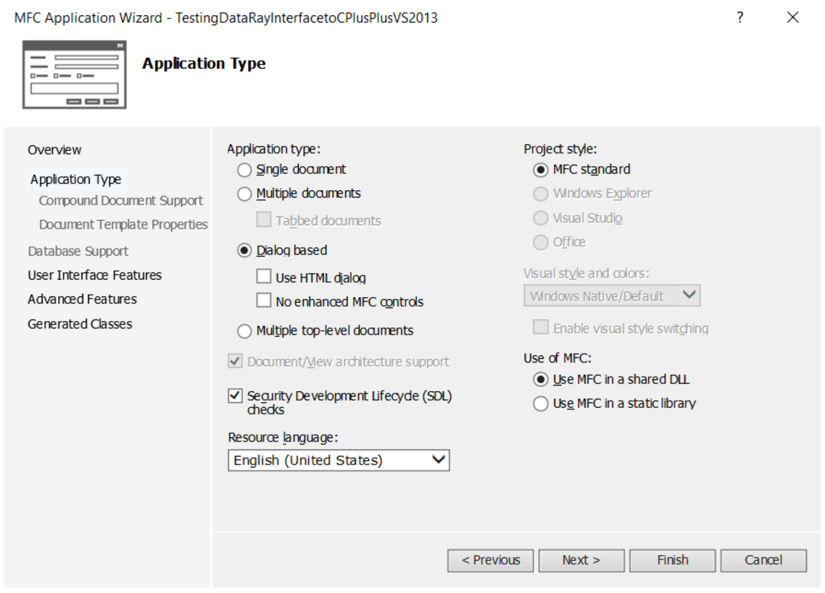
Browse...

[✓] Create directory for solution

[ ] Add to source control

OK Cancel

Figure 2: First, create a new MFC Application in VS2013.


MFC Application Wizard - TestingDataRayInterfacetoCPlusPlusVS2013

Application Type

Overview
Application Type
Compound Document Support
Document Template Properties
Database Support
User Interface Features
Advanced Features
Generated Classes

Application type:
- ○ Single document
- ○ Multiple documents
  - ☐ Tabbed documents
- [✓ ● Dialog based]
  - ☐ Use HTML dialog
  - ☐ No enhanced MFC controls
- ○ Multiple top-level documents
- ☑ Document/View architecture support
- ☑ Security Development Lifecycle (SDL) checks

Resource language: English (United States)

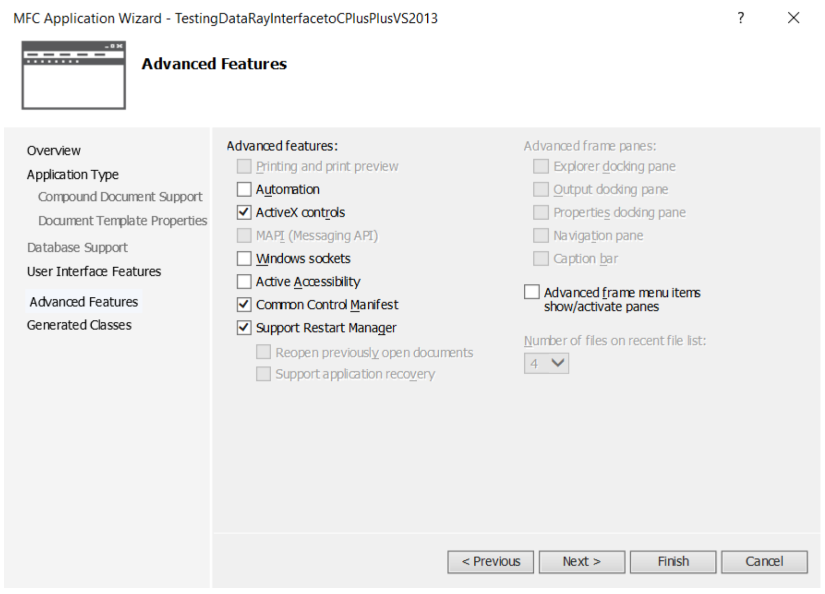
Project style:
- [✓ ● MFC standard]
- ○ Windows Explorer
- ○ Visual Studio
- ○ Office

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

< Previous | Next > | Finish | Cancel

Figure 3: Select **Dialog based** to simplify things. This is not actually a requirement, but will allow the example to be simple.


MFC Application Wizard - TestingDataRayInterfacetoCPlusPlusVS2013

Advanced Features

Overview
Application Type
Compound Document Support
Document Template Properties
Database Support
User Interface Features
Advanced Features
Generated Classes

Advanced features:
- ☐ 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
- ☐ Advanced frame menu items show/activate panes

Number of files on recent file list: 4

< Previous | Next > | Finish | Cancel

Figure 4: The default values in VS2013 are sufficient for this project. Under the **Advanced Features** tab, verify that **ActiveX controls** are enabled.

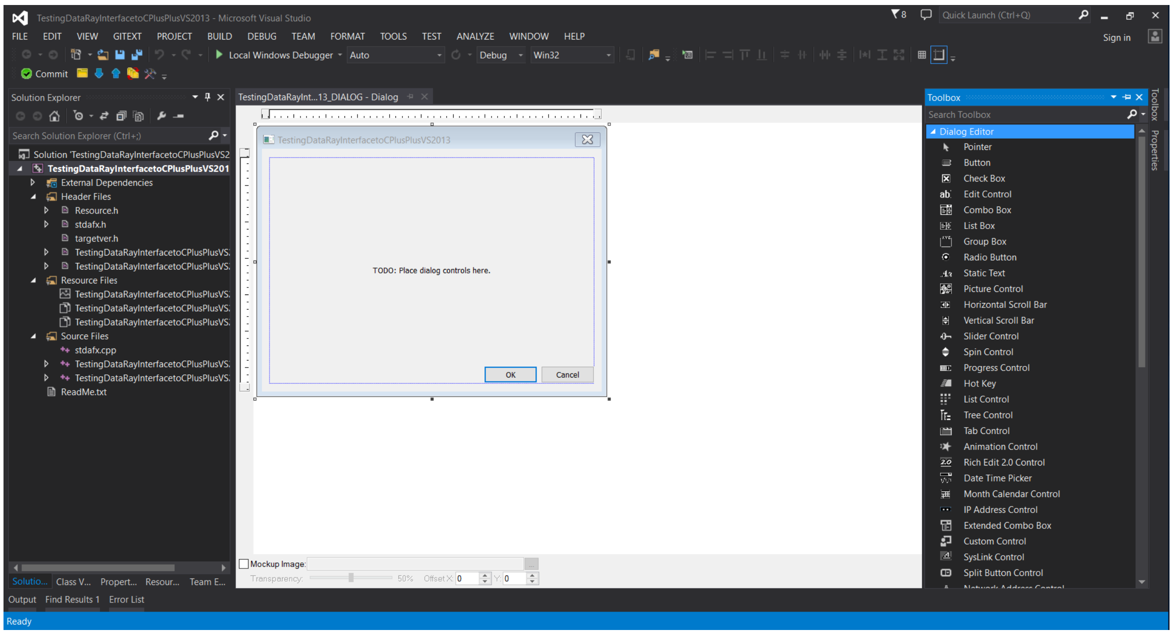
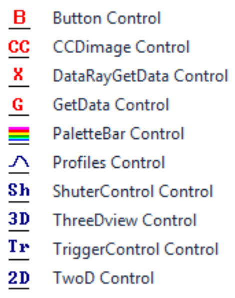


Figure 5: The empty project should look like this.



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

Figure 6: Open the **Toolbox.** You should see the DataRay components that are displayed above. 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
6. Your toolbox should now be populated with DataRay Controls.

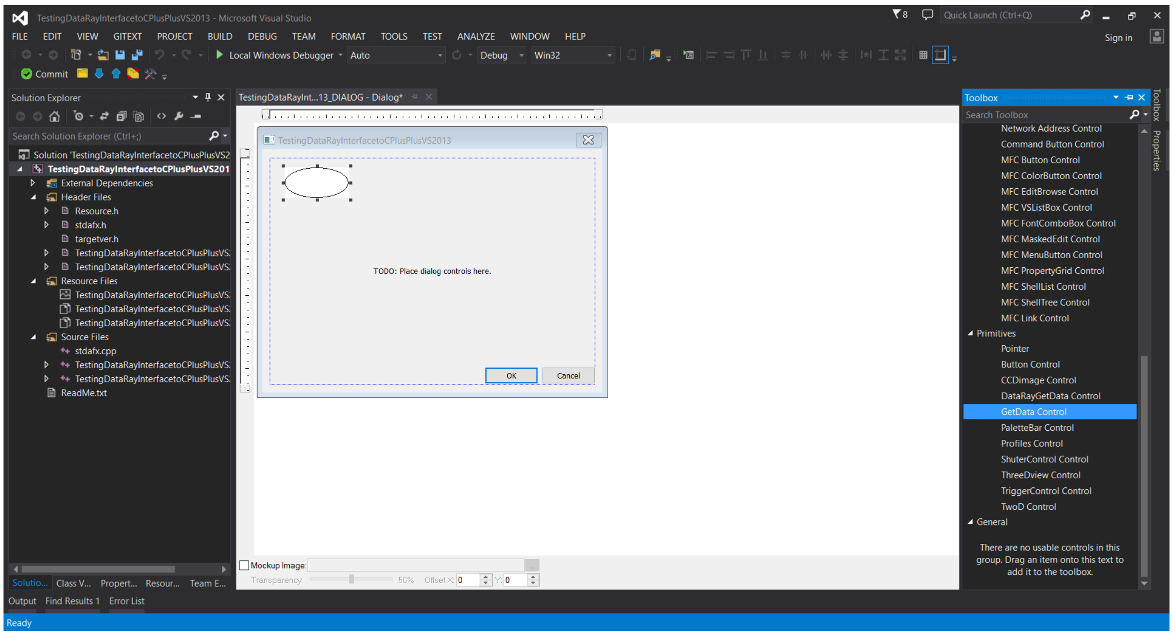
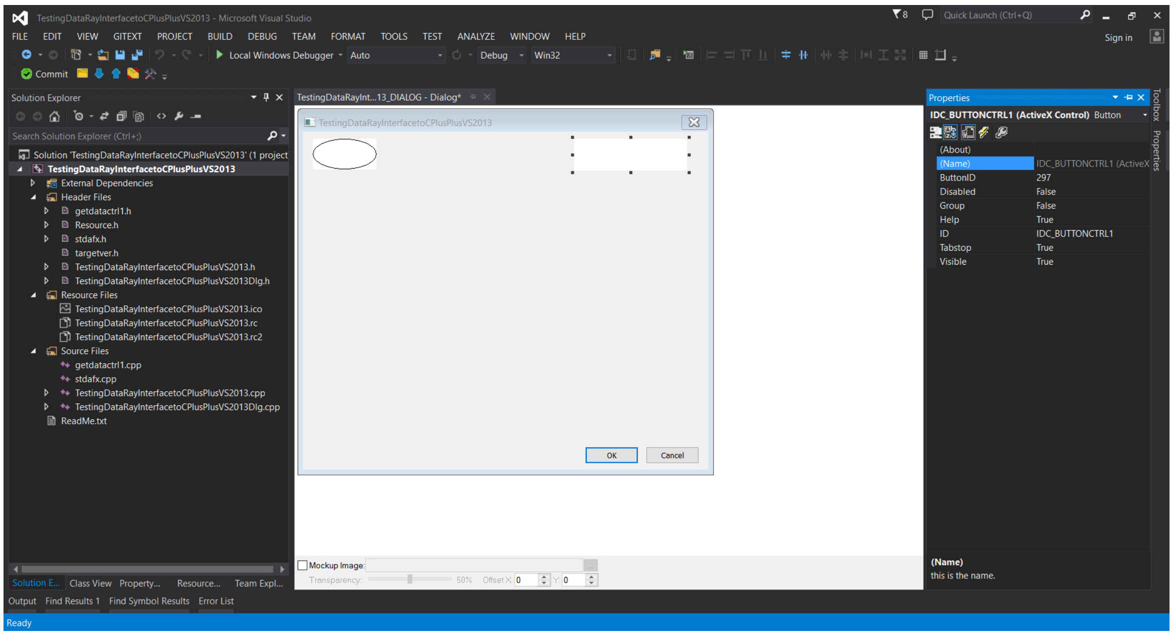


Figure 7: Now we can begin building the actual program. First drag a **GetData Control** (**not DataRayGetData Control**) onto the dialog box. This is the only OCX control class required for interfacing to DataRay cameras.


TestingDataRayInterfacetoCPlusPlusVS2013 - Microsoft Visual Studio
Solution Explorer
TestingDataRayInt...13_DIALOG - Dialog*
OK
Cancel
Properties
IDC_BUTTONCTRL1 (ActiveX Control) Button
Ready

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

(Name)
this is the name.

Figure 8: We will 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** and then exit the **Edit Control** window. Right click on the button again. This time, select **Properties**. Change the **ButtonID** to **297**.

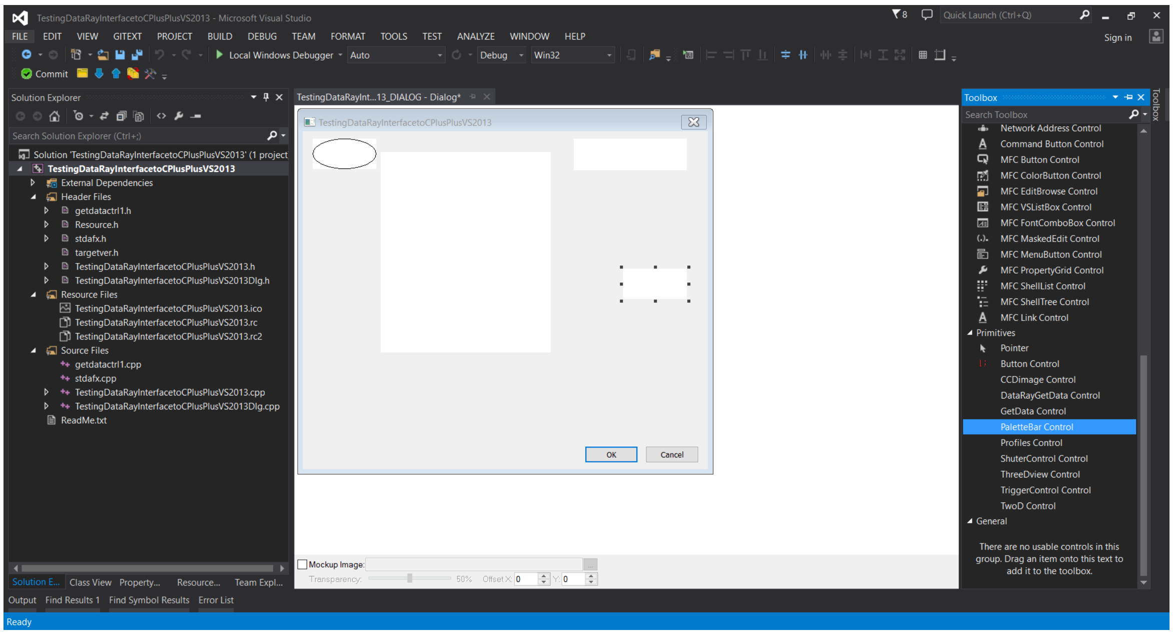
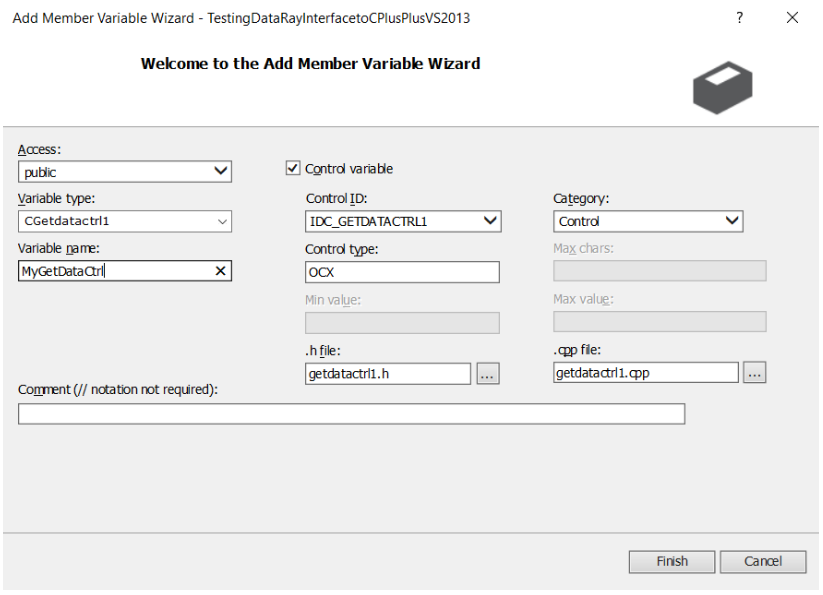

TestingDataRayInterfacetoCPlusPlusVS2013 - Microsoft Visual Studio
FILE EDIT VIEW GITEXT PROJECT BUILD DEBUG TEAM FORMAT TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Debug
Win32
Commit
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'TestingDataRayInterfacetoCPlusPlusVS2013' (1 project
TestingDataRayInterfacetoCPlusPlusVS2013
External Dependencies
Header Files
getdatactrl1.h
Resource.h
stdafx.h
targetver.h
TestingDataRayInterfacetoCPlusPlusVS2013.h
TestingDataRayInterfacetoCPlusPlusVS2013Dlg.h
Resource Files
TestingDataRayInterfacetoCPlusPlusVS2013.ico
TestingDataRayInterfacetoCPlusPlusVS2013.rc
TestingDataRayInterfacetoCPlusPlusVS2013.rc2
Source Files
getdatactrl1.cpp
stdafx.cpp
TestingDataRayInterfacetoCPlusPlusVS2013.cpp
TestingDataRayInterfacetoCPlusPlusVS2013Dlg.cpp
ReadMe.txt
TestingDataRayInt...13_DIALOG - Dialog*
TestingDataRayInterfacetoCPlusPlusVS2013
OK
Cancel
Mockup Image:
Transparency:
Toolbox
Search Toolbox
Network Address Control
Command Button Control
MFC Button Control
MFC ColorButton Control
MFC EditBrowse Control
MFC VSListBox Control
MFC FontComboBox Control
MFC MaskedEdit Control
MFC MenuButton Control
MFC PropertyGrid Control
MFC ShellList Control
MFC ShellTree Control
MFC Link Control
Primitives
Pointer
Button Control
CCDimage Control
DataRayGetData Control
GetData Control
PaletteBar Control
Profiles Control
ShuterControl Control
ThreeDview Control
TriggerControl Control
TwoD Control
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Solution E... Class View Property... Resource... Team Expl...
Output Find Results 1 Find Symbol Results Error List
Ready

Figure 9: Now drag one **CCDimage Control** from the toolbox to the dialog box. This object will be used to view the image produced by the hardware, and its height must be 50 pixels greater than its width. Finally, drag one **PaletteBar Control** from the toolbox to the dialog box. This completes the basic layout of the dialog box.


Add Member Variable Wizard - TestingDataRayInterfacetoCPlusPlusVS2013

Welcome to the Add Member Variable Wizard

Access: public

[✓] Control variable

Variable type: CGetdatactrl1

Control ID: IDC_GETDATACTRL1

Category: Control

Variable name: MyGetDataCtrl

Control type: OCX

Max chars:

Min value:

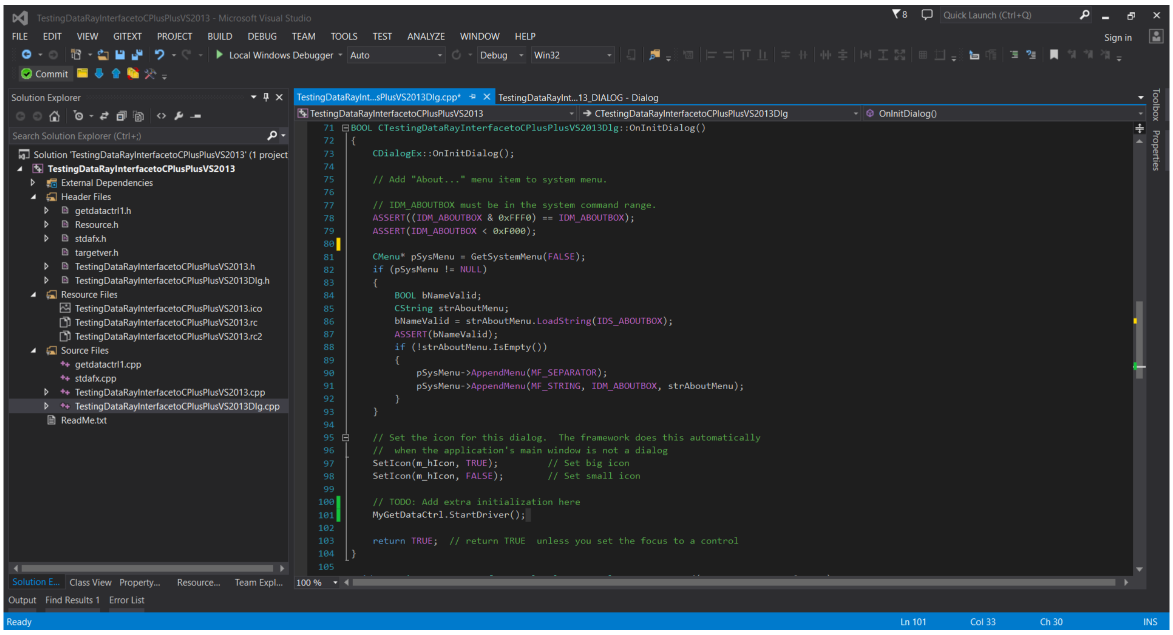
Max value:

.h file: getdatactrl1.h

.cpp file: getdatactrl1.cpp

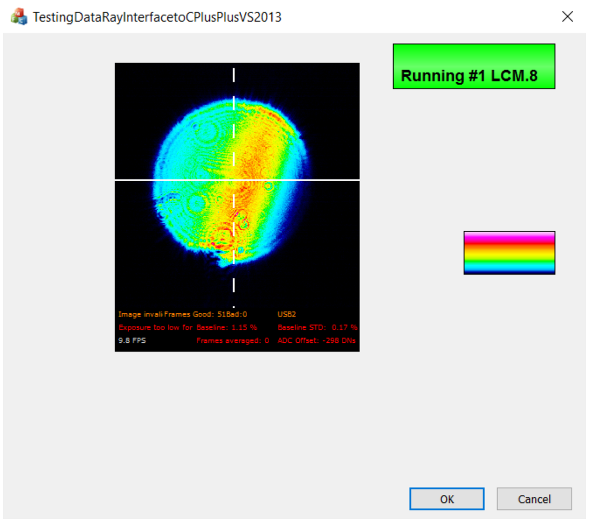
Comment (// notation not required):

Figure 10: Now we need to add some code to the template. Right-click on the **GetData Control** and select **Add Variable**. Name the variable **MyGetDataCtrl** and make sure **Control variable** is checked. 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.


TestingDataRayInterfacetoCPlusPlusVS2013 - Microsoft Visual Studio
FILE EDIT VIEW GITEXT PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Debug
Win32
Commit
Solution Explorer
Solution 'TestingDataRayInterfacetoCPlusPlusVS2013' (1 project
TestingDataRayInterfacetoCPlusPlusVS2013
External Dependencies
Header Files
getdatactrl1.h
Resource.h
stdafx.h
targetver.h
TestingDataRayInterfacetoCPlusPlusVS2013.h
TestingDataRayInterfacetoCPlusPlusVS2013Dlg.h
Resource Files
TestingDataRayInterfacetoCPlusPlusVS2013.ico
TestingDataRayInterfacetoCPlusPlusVS2013.rc
TestingDataRayInterfacetoCPlusPlusVS2013.rc2
Source Files
getdatactrl1.cpp
stdafx.cpp
TestingDataRayInterfacetoCPlusPlusVS2013.cpp
TestingDataRayInterfacetoCPlusPlusVS2013Dlg.cpp
ReadMe.txt
TestingDataRayInt...sPlusVS2013Dlg.cpp*
TestingDataRayInt...13_DIALOG - Dialog
CTestingDataRayInterfacetoCPlusPlusVS2013Dlg
OnInitDialog()
Ready
Ln 101
Col 33
Ch 30
INS

```
BOOL CTestingDataRayInterfacetoCPlusPlusVS2013Dlg::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
}
```

Figure 11: Add the following line: MyGetDataCtrl.StartDriver(); to your initialize dialog function BOOL CTestingDataRayInterfaceToCPlusPlusVS2013Dlg::OnInitDialog().



```
Image invali Frames Good: 51Bad:0          USB2
Exposure too low for  Baseline: 1.15 %     Baseline STD:  0.17 %
9.8 FPS           Frames averaged: 0      ADC Offset: -298 DNs
```

Figure 12: 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!

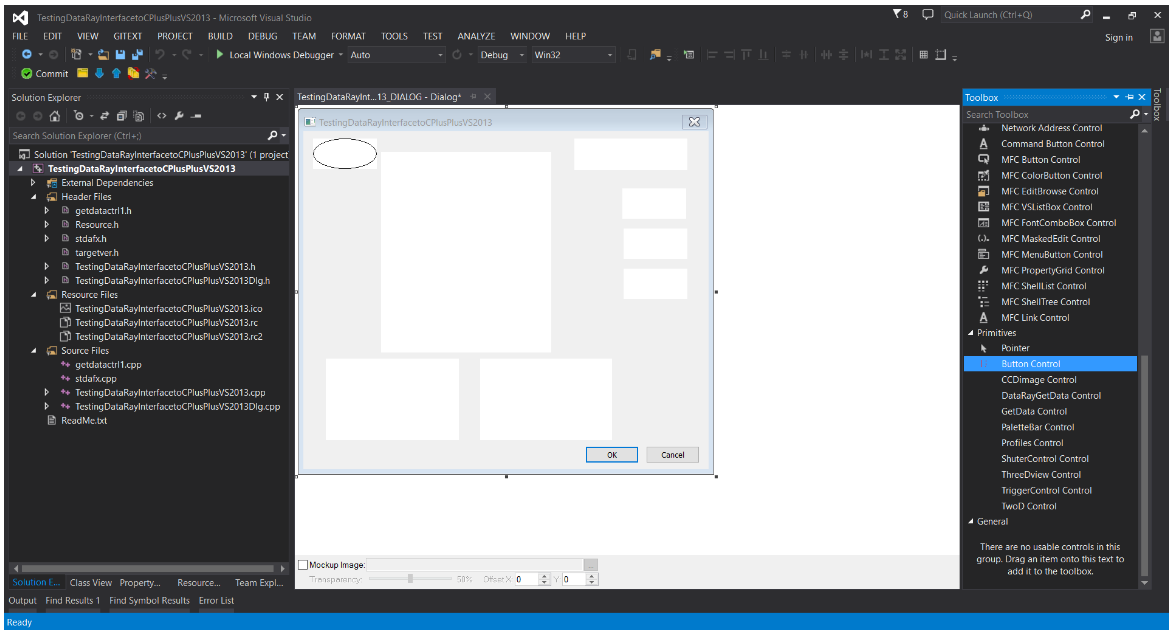

TestingDataRayInterfacetoCPlusPlusVS2013 - Microsoft Visual Studio
FILE EDIT VIEW GITEXT PROJECT BUILD DEBUG TEAM FORMAT TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Debug
Win32
Commit
Solution Explorer
Solution 'TestingDataRayInterfacetoCPlusPlusVS2013' (1 project
TestingDataRayInterfacetoCPlusPlusVS2013
External Dependencies
Header Files
getdatactrl1.h
Resource.h
stdafx.h
targetver.h
TestingDataRayInterfacetoCPlusPlusVS2013.h
TestingDataRayInterfacetoCPlusPlusVS2013Dlg.h
Resource Files
TestingDataRayInterfacetoCPlusPlusVS2013.ico
TestingDataRayInterfacetoCPlusPlusVS2013.rc
TestingDataRayInterfacetoCPlusPlusVS2013.rc2
Source Files
getdatactrl1.cpp
stdafx.cpp
TestingDataRayInterfacetoCPlusPlusVS2013.cpp
TestingDataRayInterfacetoCPlusPlusVS2013Dlg.cpp
ReadMe.txt
TestingDataRayInt...13_DIALOG - Dialog*
OK
Cancel
Mockup Image
Toolbox
Network Address Control
Command Button Control
MFC Button Control
MFC ColorButton Control
MFC EditBrowse Control
MFC VSListBox Control
MFC FontComboBox Control
MFC MaskedEdit Control
MFC MenuButton Control
MFC PropertyGrid Control
MFC ShellList Control
MFC ShellTree Control
MFC Link Control
Primitives
Pointer
Button Control
CCDimage Control
DataRayGetData Control
GetData Control
PaletteBar Control
Profiles Control
ShuterControl Control
ThreeDview Control
TriggerControl Control
TwoD Control
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Output Find Results 1 Find Symbol Results Error List
Ready

Figure 13: 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**.

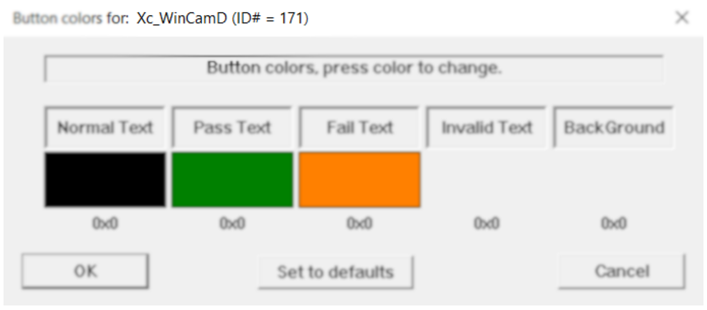


Figure 14: In order to find the correct **ButtonID** to use for each object in your custom interface, you need to:

1. Close VS2013 and open the DataRay software
2. Right click on any button, to see the dialog
3. Note the current Name and ID# for this result at the top of the dialog
4. Repeat for all the results of interest
5. Close the DataRay Software

Following these instructions, you will be able to tell that to see **Xc** and **Yc**, we should change the **ButtonIDs** to **171** and **172**. For the profiles, change the **ProfileIDs** to **22** and **23**.
A complete list of Button IDs:
http://www.dataray.com/UserFiles/file/IndexToTestParametersEnum.pdf
A complete list of Profile IDs:
http://www.dataray.com/UserFiles/file/ProfilesEnum.pdf

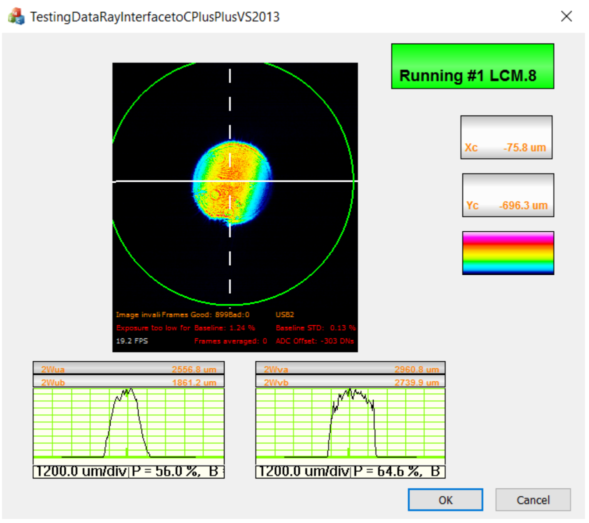
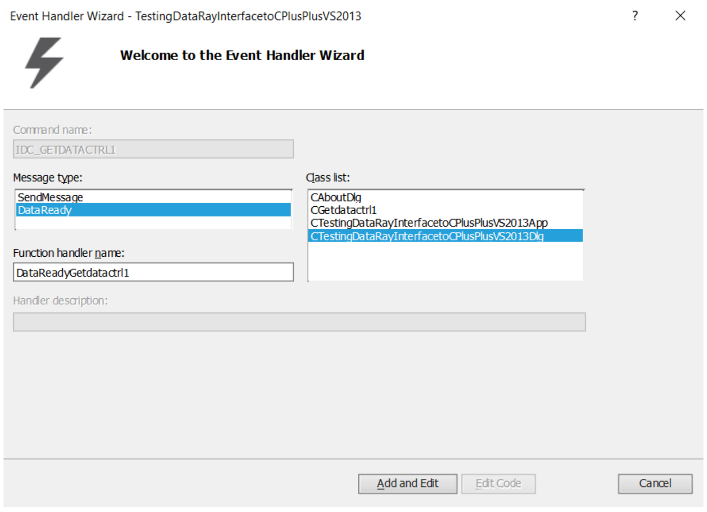


Image invali Frames Good: 899Bad:0 USB2
Exposure too low for Baseline: 1.24 % Baseline STD: 0.13 %
19.2 FPS Frames averaged: 0 ADC Offset: -303 DNs

Figure 15: Build and run your program and you should see your custom interface featuring the **Xc button**, **Yc button**, **Xc profile** and **Yc profile**. This completes the basic tutorial! **Problems/Questions?** Contact us with the information listed on the first page of this document. Continue reading for additional tutorials regarding event handling and programmatically extracting data from the OCX.

Event Handling:

Consider utilizing event handling if you would like to include code in your project that will be executed only when a specified event occurs. In this tutorial, we will be handling GetData Control DataReady events. This is an effective way to log data on a per frame basis, because the DataReady event is called every time a new frame is available for processing. Our example code contains a function that gets called every time a DataReady event is received. You can insert whatever code you want into this function and it will be executed once per new image.



Event Handler Wizard - TestingDataRayInterfacetoCPlusPlusVS2013

Welcome to the Event Handler Wizard

Command name: IDC_GETDATACTRL1

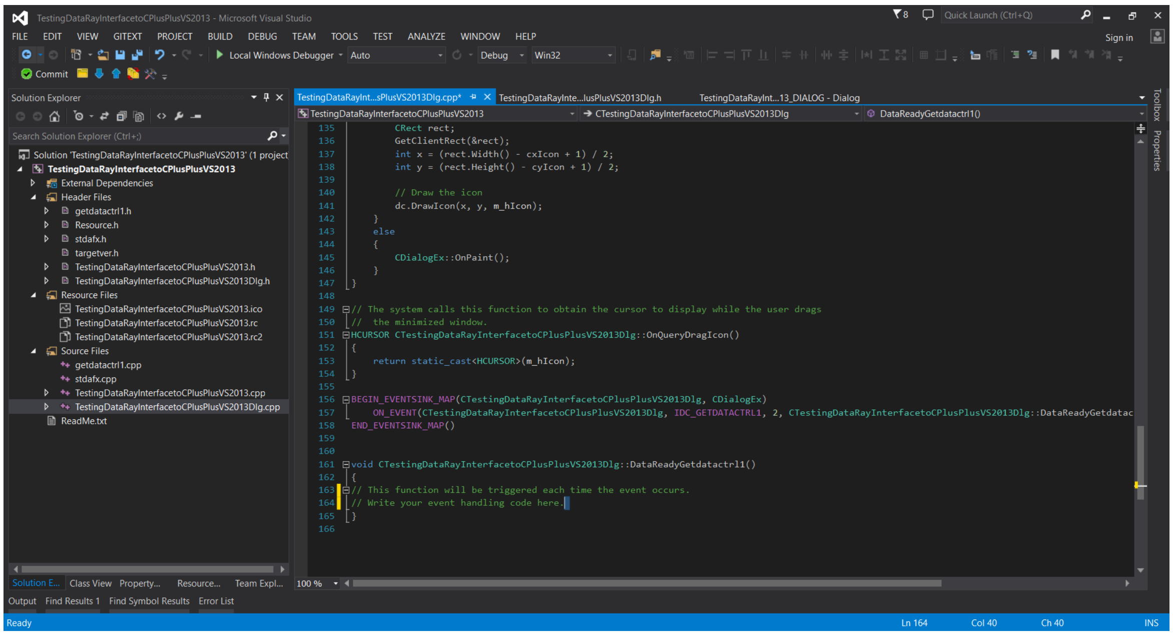
Message type: SendMessage, DataReady (selected)

Class list: CAboutDlg, CGetdatactrl1, CTestingDataRayInterfacetoCPlusPlusVS2013App, CTestingDataRayInterfacetoCPlusPlusVS2013Dlg (selected)

Function handler name: DataReadyGetdatactrl1

Handler description:

Figure 16: To begin event handling for GetData Control events, right click the GetData Control object in the dialog box and select **Add Event Handler**. Choose **DataReady** from the "Message type:" menu. The function handler name will be set to DataReadyGetdatactrl1.



TestingDataRayInterfacetoCPlusPlusVS2013 - Microsoft Visual Studio

```
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) / 2;
        int y = (rect.Height() - cyIcon + 1) / 2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialogEx::OnPaint();
    }
}

// The system calls this function to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CTestingDataRayInterfacetoCPlusPlusVS2013Dlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}

BEGIN_EVENTSINK_MAP(CTestingDataRayInterfacetoCPlusPlusVS2013Dlg, CDialogEx)
    ON_EVENT(CTestingDataRayInterfacetoCPlusPlusVS2013Dlg, IDC_GETDATACTRL1, 2, CTestingDataRayInterfacetoCPlusPlusVS2013Dlg::DataReadyGetdatac
END_EVENTSINK_MAP()


void CTestingDataRayInterfacetoCPlusPlusVS2013Dlg::DataReadyGetdatactrl1()
{
// This function will be triggered each time the event occurs.
// Write your event handling code here.
}

```

FILE EDIT VIEW GITEXT PROJECT BUILD DEBUG TEAM TOOLS TEST ANALYZE WINDOW HELP
Local Windows Debugger
Auto
Debug
Win32
Commit
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'TestingDataRayInterfacetoCPlusPlusVS2013' (1 project
TestingDataRayInterfacetoCPlusPlusVS2013
External Dependencies
Header Files
getdatactrl1.h
Resource.h
stdafx.h
targetver.h
TestingDataRayInterfacetoCPlusPlusVS2013.h
TestingDataRayInterfacetoCPlusPlusVS2013Dlg.h
Resource Files
TestingDataRayInterfacetoCPlusPlusVS2013.ico
TestingDataRayInterfacetoCPlusPlusVS2013.rc
TestingDataRayInterfacetoCPlusPlusVS2013.rc2
Source Files
getdatactrl1.cpp
stdafx.cpp
TestingDataRayInterfacetoCPlusPlusVS2013.cpp
TestingDataRayInterfacetoCPlusPlusVS2013Dlg.cpp
ReadMe.txt
TestingDataRayInt...sPlusVS2013Dlg.cpp*
TestingDataRayInte...lusPlusVS2013Dlg.h
TestingDataRayInt...13_DIALOG - Dialog
CTestingDataRayInterfacetoCPlusPlusVS2013Dlg
DataReadyGetdatactrl1()
Ready
Ln 164
Col 40
Ch 40
INS

Figure 17: Locate the function called DataReadyGetdatactrl1() in TestingDataRayInterfacetoCPlusPlusVS2013Dlg.cpp. This function will be triggered each time a GetData Control event occurs. Write your event handling code here, within the function's definition. This completes the event handling tutorial.

Programmatically Extracting Data from the OCX:

There are two main methods for extracting data from the OCX in your program. One method is to create an instance of the control class (same steps as earlier for the **GetData Control** (see Fig. 10)). 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 large amounts of data via variants:

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

This is the preferred method for reading large amounts of data from the OCX. In this section of tutorial, we will create a button that will, when clicked, read and retrieve the 2D image data via a variant and store the information as an array of pixel data in a .csv file. The image data is obtained from the OCX through invocation of the GetWinCamDataAsVariant() function of the GetData ActiveX control (see Fig. 19 for sample code).

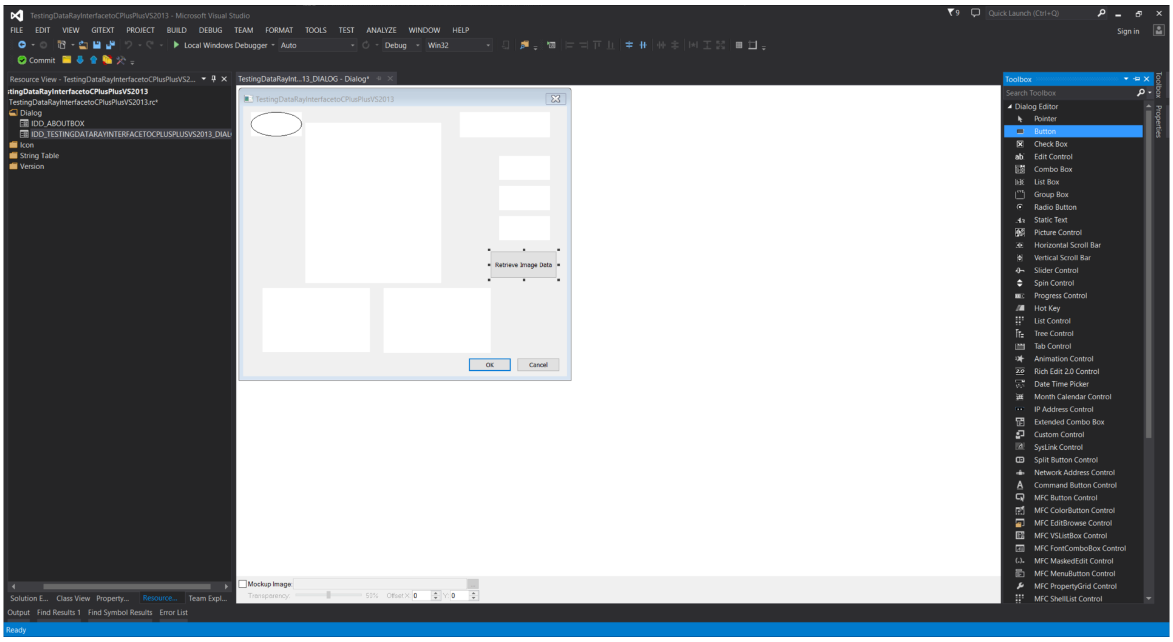
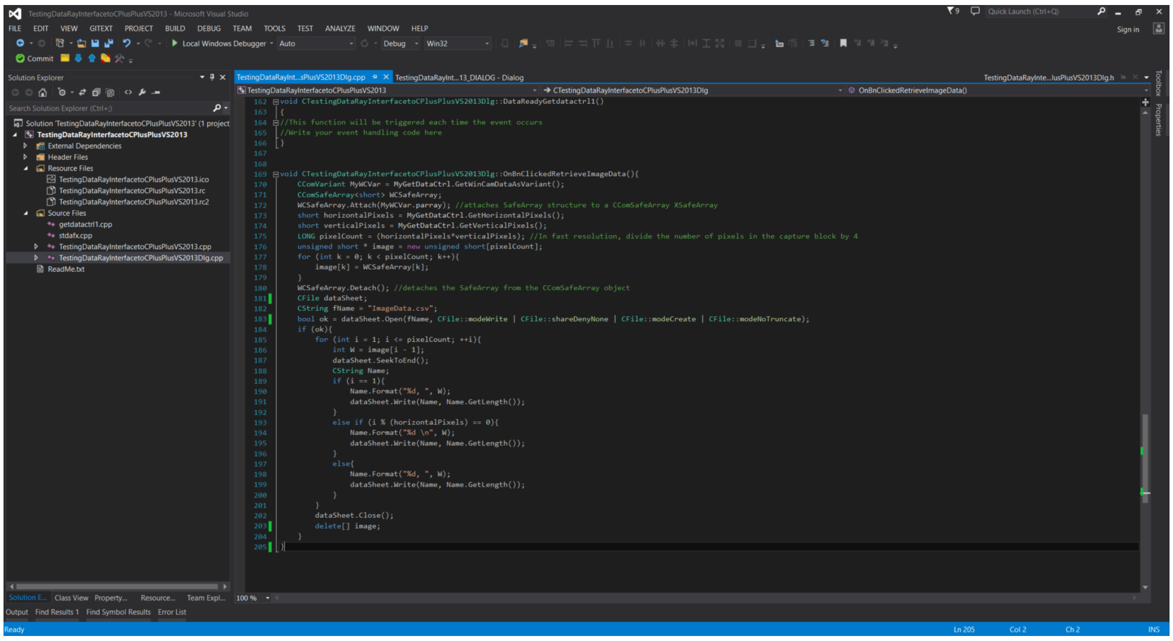


Figure 18: Using the toolbox, add an object that users can click by dragging a **Button** from the Dialog Editor section (not a Button Control from the Primitives section). Right click on the button and choose **Properties**. Change the caption to Retrieve Image Data. Next, right click on the button and choose **Add Event Handler**. Make sure the "Message type" is set to **BN_CLICKED** and set the Function handler name to **OnBnClickedRetrieveImageData**.



```
void CTestingDataRayInterfacetoCPlusPlusVS2013Dlg::DataReadyGetdatactrl1()
{
//This function will be triggered each time the event occurs
//Write your event handling code here
}


void CTestingDataRayInterfacetoCPlusPlusVS2013Dlg::OnBnClickedRetrieveImageData(){
    CComVariant MyWCVar = MyGetDataCtrl.GetWinCamDataAsVariant();
    CComSafeArray<short> WCSafeArray;
    WCSafeArray.Attach(MyWCVar.parray); //attaches SafeArray structure to a CComSafeArray XSafeArray
    short horizontalPixels = MyGetDataCtrl.GetHorizontalPixels();
    short verticalPixels = MyGetDataCtrl.GetVerticalPixels();
    LONG pixelCount = (horizontalPixels*verticalPixels); //In fast resolution, divide the number of pixels in the capture block by 4
    unsigned short * image = new unsigned short[pixelCount];
    for (int k = 0; k < pixelCount; k++){
        image[k] = WCSafeArray[k];
    }
    WCSafeArray.Detach(); //detaches the SafeArray from the CComSafeArray object
    CFile dataSheet;
    CString fName = "ImageData.csv";
    bool ok = dataSheet.Open(fName, CFile::modeWrite | CFile::shareDenyNone | CFile::modeCreate | CFile::modeNoTruncate);
    if (ok){
        for (int i = 1; i <= pixelCount; ++i){
            int W = image[i - 1];
            dataSheet.SeekToEnd();
            CString Name;
            if (i == 1){
                Name.Format("%d, ", W);
                dataSheet.Write(Name, Name.GetLength());
            }
            else if (i % (horizontalPixels) == 0){
                Name.Format("%d \n", W);
                dataSheet.Write(Name, Name.GetLength());
            }
            else{
                Name.Format("%d, ", W);
                dataSheet.Write(Name, Name.GetLength());
            }
        }
        dataSheet.Close();
        delete[] image;
    }
}
```

Figure 19: Adding the event handler to the Retrieve Image Data button will automatically create a function called OnBnClickedRetrieveImageData() in the TestingDataRayInterfacetoCPlusPlusVS2013.cpp source file. Within this function's definition, you will write your code that retrieves the 2D image data from the DataRay OCX and save the data as a .csv file. This sample code saves data retrieved from GetWinCamDataAsVariant() in a filed called **ImageData.csv**, which is created in the TestingDataRayInterfacetoCPlusPlusVS2013 folder when the button is clicked. If you plan on using the CComSafeArray to handle the variant data, be sure to include the atlsafe.h library in the header file. This completes the tutorial. **Problems/Questions?** Contact us with the information listed on the first page of this document.