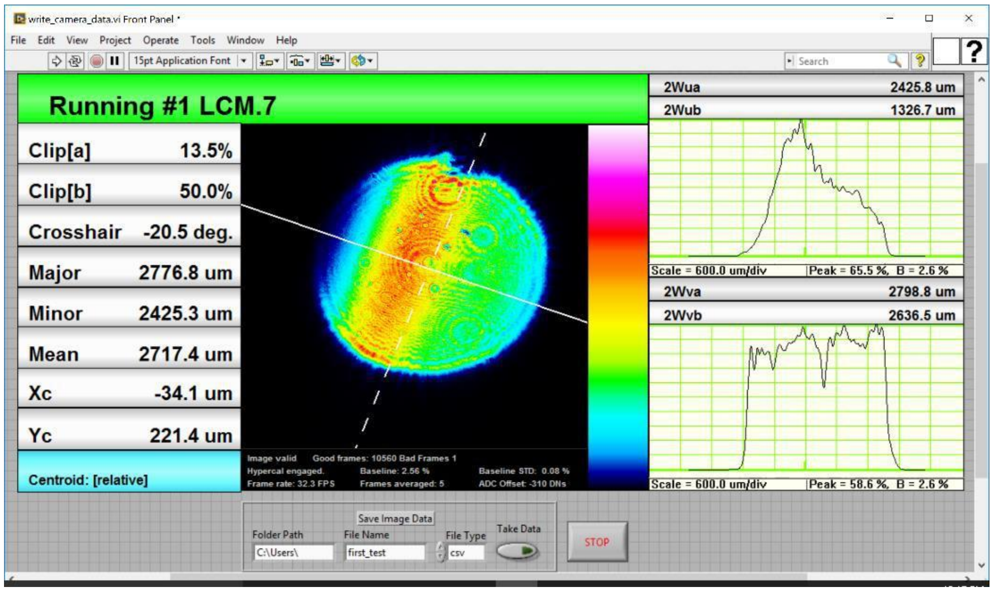
support@dataray.com
1675 Market Street
Redding, CA 96001
+1 530 395 2500

Interfacing to LabVIEW

OVERVIEW:


write_camera_data.vi Front Panel *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Save Image Data
Take Data
STOP

Running #1 LCM.7

Clip[a]	13.5%
Clip[b]	50.0%
Crosshair	-20.5 deg.
Major	2776.8 um
Minor	2425.3 um
Mean	2717.4 um
Xc	-34.1 um
Yc	221.4 um

Centroid: [relative]

Image valid Good frames: 10560 Bad Frames 1
Hypercal engaged. Baseline: 2.56 % Baseline STD: 0.08 %
Frame rate: 32.3 FPS Frames averaged: 5 ADC Offset: -310 DNs

2Wua	2425.8 um
2Wub	1326.7 um

Scale = 600.0 um/div Peak = 65.5 %, B = 2.6 %

2Wva	2798.8 um
2Wvb	2636.5 um

Scale = 600.0 um/div Peak = 58.6 %, B = 2.6 %

Folder Path: C:\Users\
File Name: first_test
File Type: csv

GETTING STARTED:

INTERFACING WITH OCX

Your interfacing LabVIEW VI communicates with DataRay products through the DataRay OCX. The OCX has ActiveX controls 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. ActiveX controls (figures, buttons, graphics etc.) from the DataRay standalone application can be added to the LabVIEW front panel. These ActiveX controls can be controlled on the LabVIEW front panel as in the DataRay standalone application (clicking on the panels during run mode). Additionally, the objects can be controlled using invoke and property nodes found in the LabVIEW software.

This tutorial will consist of a basic section on adding ActiveX controls to the front panel and an advanced section on acquiring data from the OCX. Once initialized in a VI, the ActiveX controls are always running, even while editing the VI. Therefore, the OCX's GUI elements (including figures) will still be active while the VI is in editing mode since the camera is still on.

SOME IMPORTANT NOTES:

- Since the OCX is 32-bit, you will need 32-bit LabVIEW and libraries
- The OCX and DataRay program cannot be used at the same time

INSTALLATION:

First we need to install the DataRay 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.

Second we need to install the associated LabVIEW software:

- Install a 32-bit version of LabVIEW.

Current support runs from LabVIEW 8.5 to LabVIEW 2015. You can download the interface developed in this tutorial. It exists as a collection of 5 LabVIEW files:

- **WinCamD Camera Example**: Download & unzip.
- **WinCamD Button Example:** Download & unzip**.**
- **WinCamD Event Example:** Download & unzip.
- **BeamMap Example:** Download & unzip.
- **WinCamD Multicamera Example:** Download & unzip.

Note: All the settings from the last DataRay.exe software session are recalled and used unless expressly changed in LabVIEW.

This example should build and run with no errors. Not working? Email support@dataray.com or call (530) 395-2500 with:

- Device name and serial number
- DataRay and Windows versions which you are using.

support@dataray.com
1675 Market Street
Redding, CA 96001
+1 530 395 2500

BASIC TUTORIAL:

We will show you step-by-step how the example VIs were created in LabVIEW for the examples below.

ADDING DATARAY OCX ELEMENTS

To add a DataRay OCX GUI element first create an **ActiveX Container** on the front panel (found in the **.NET & ActiveX** menu, see Figure 1). Once the container has been created right click the container and select **Insert ActiveX Object** from the menu (see Figure 2). A dialog box will appear with a drop down menu and a list of all ActiveX objects in Windows (see Figure 3). Select **Create Control** from the drop down menu and then the desired DataRay ActiveX Object from the list provided. The available DataRay OCX classes and their associated functions (which can be accessed using invoke or property nodes) can be found in the documentation. For buttons and profiles, an ID property of the object must be set to determine what values they will represent. To change a property of an object, right click the object and select object name>properties (e.g. **Button->Properties** or **Profiles->Properties**, see Figure 4). A dialog box will appear with a list of properties pertaining to that class (see Figure 5). Select a number from the list to assign it to the object. Some objects, such as **CCDimage Control** and **PaletteBar Control,** have only one option and thus no list will appear.

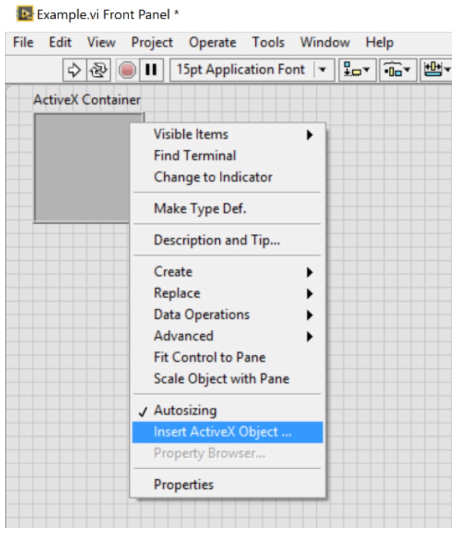


Figure 1

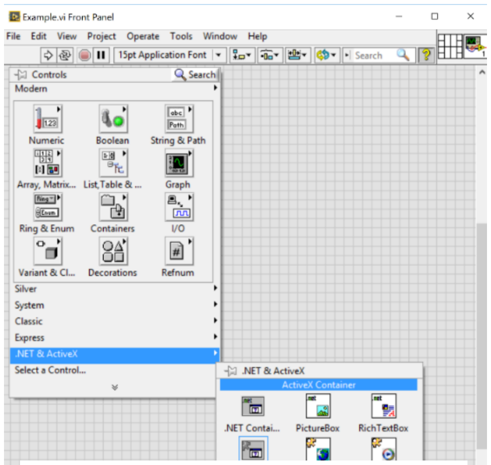
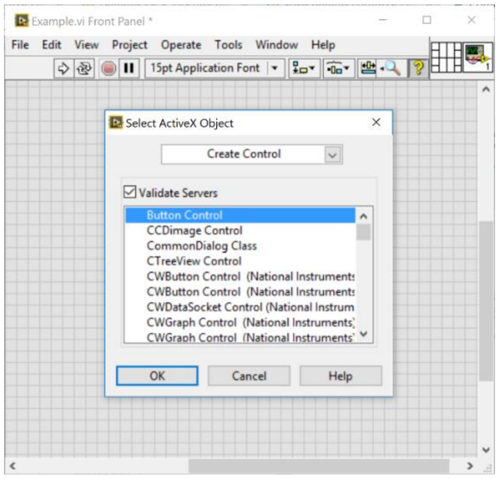


Figure 2


Example.vi Front Panel *
File Edit View Project Operate Tools Window Help
15pt Application Font
Select ActiveX Object
Create Control
Validate Servers
Button Control
CCDimage Control
CommonDialog Class
CTreeView Control
CWButton Control (National Instruments
CWButton Control (National Instruments
CWDataSocket Control (National Instrum
CWGraph Control (National Instruments)
CWGraph Control (National Instruments'
OK
Cancel
Help
Figure 3

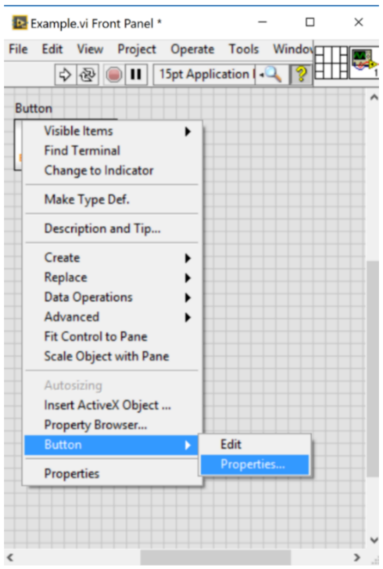
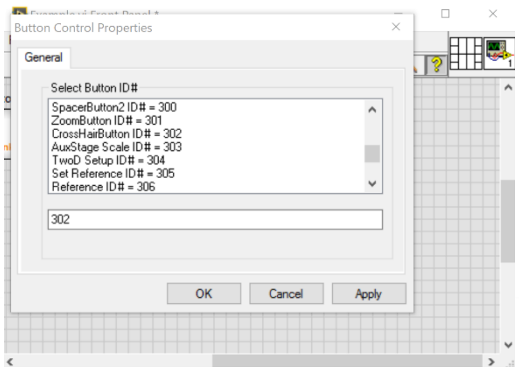


Figure 4



Select Button ID#
SpacerButton2 ID# = 300
ZoomButton ID# = 301
CrossHairButton ID# = 302
AuxStage Scale ID# = 303
TwoD Setup ID# = 304
Set Reference ID# = 305
Reference ID# = 306

302

Figure 5

CREATING A VI

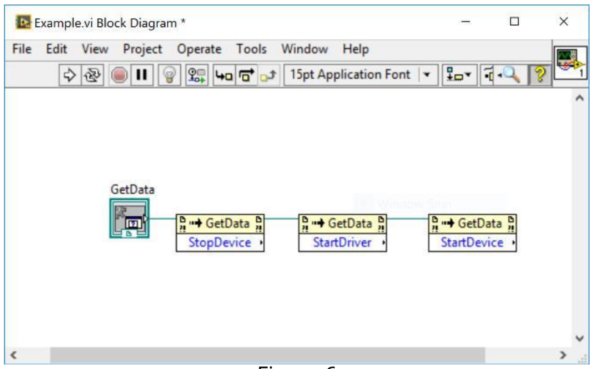


Figure 6

The **GetData Control** class and a few of its methods are absolutely necessary for an interface to the DataRayOCX. This class controls the DataRay device and is used to acquire data. To begin a VI, insert the **GetData Control** class into an **ActiveX Container**. Next, connect a succession of three invoke nodes to the **GetData Control** element (see Figure 6). In the first invoke node select **StopDevice** from the methods listed. This will stop any processes previously running on the device. Select **StartDriver** on the second invoke node from the methods listed. This will initialize the driver if not already running or restart the device driver. From the third invoke node select **StartDevice**. This will start the device's functionality. These three nodes in conjunction with the **GetData Control** object are essential to any VI running DataRay software. To add further control or acquire data, a combination of invoke

Note: If you have downloaded the example VIs, they may fail to compile due to a different **GetData Control** object in your DataRay software. Delete the **GetData Control** object, manually recreate it and then rewire it to the invoke nodes. Manually reselect the appropriate methods from the invoke nodes.

ADDING BEAM IMAGE

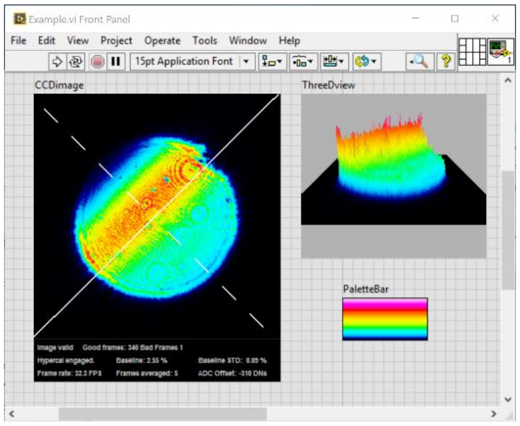


Figure 7: Images available for a camera profiler.

A variety of beam images can be displayed on the LabVIEW VI front panel. However, the type of 2-D image that can be displayed is dependent on the type of beam profiler used, either camera profiler or scanning slit profiler. Due to the communication between OCX elements and LabVIEW, the VI must be saved and then restarted before an image can be displayed on the image objects.

2D IMAGE

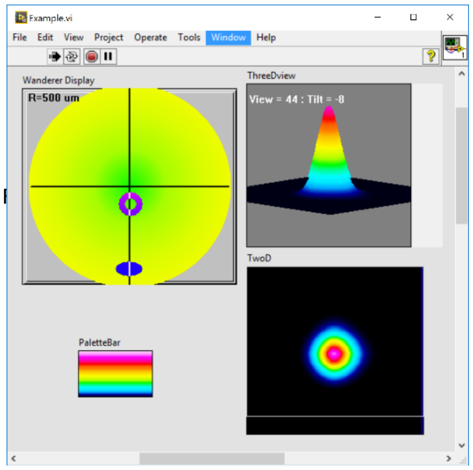


Figure 8: Images available for a scanning slit profiler.

- A direct image of the beam can be displayed when using a DataRay Camera Profiler. To display the image of the beam, both the **CCDimage Control** and **PaletteBar Control** ActiveX objects must be placed on the front panel. Initially, the **CCDimage Control** object on the front panel will be small, but you can select the object and drag to increase its size. If the **PaletteBar Control** object is not placed on the front panel, the **CCDimage Control** element will remain dark and without color. The **CCDimage Control object** is only compatible with DataRay cameras and not scanning slit profilers.

- When using a scanning slit profiler, the wander display (yellow-green, see Figure 8) can be displayed. The wander display shows the position of the beam relative to the calibrated (0,0) position. To display this image, insert a **Button Control** object onto the front panel. Right click the button and select **Button->Properties**. From the list of available button controls select 197. Select the display and enlarge. The wander display is only compatible with DataRay scanning slit profilers, not cameras.

- The 2-D intensity image can be displayed for scanning slit profilers as well. To insert this image, add the **TwoD Control** object onto the front panel. Select the display and enlarge. The **PaletteBar Control** object must be included in the VI for the image to be seen. The 2-D intensity image does not contain actual 2-D data. The 2D image is an artificial reconstruction which assumes the same X profile for all values of Y and the same Y profile for all values of X. The 2-D intensity display is only compatible with DataRay scanning slit profilers, not cameras.

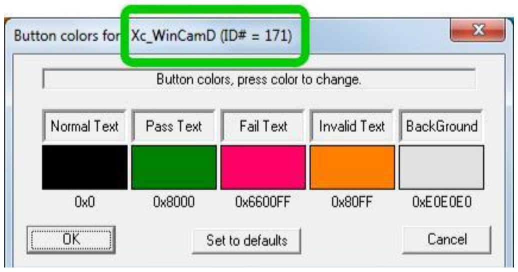
3D IMAGE

- To display a 3D image of the beam. Insert the **ThreeDview Control** object onto the front panel. Select the box and enlarge. The **PaletteBar Control** object must be included in the VI for the image to be seen. This object can be used both for cameras, and slit profilers. However, when used with the scanning slit profiler, the 3-D is an artificial reconstruction which assumes the same X profile for all values of Y and the same Y profile for all values of X.

EXTENDING THE PROGRAM

Besides the names of the ActiveX controls, you will need to know the ID's for specific button and profiles. In order to find the correct **Button ID#** to use for the buttons, you need to:

1) Close your GUI 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.

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

Figure 9

There are complete lists of ID's for profiles and buttons available in interface section of the DataRay website:

Profiles

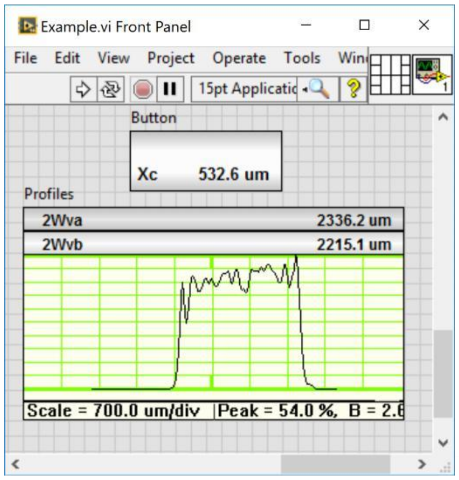
Buttons

Finally, there is documentation describing the methods and properties of all the ActiveX controls:

Documentation

ADDING A BEAM PROFILE

To display the profile of the beam, the **Profiles Control** ActiveX object must be placed on the front panel. After the object has been placed, right click the object and select **Profiles->Properties** from the menu. A dialog box will appear with a list of the profiles available. The profile number can be found via the instructions in the Extending the GUI section. The profile object will initially be small, but by selecting the object and increasing the size, the image and information can be seen.



Xc 532.6 um

2Wva 2336.2 um

2Wvb 2215.1 um

Scale = 700.0 um/div |Peak = 54.0 %, B = 2.6

Figure 10: Button and Profile

ADDING BUTTONS

The remaining elements in the LabVIEW examples provided are all of the **Button Control** class. To create a button panel, insert a **Button Control** ActiveX object on the front panel. After the object has been placed, right click the object and select **Profiles->Properties** from the menu. A dialog box will appear with a list of the profiles available. The button number can be found via the instructions in the Extending the GUI section.

support@dataray.com
1675 Market Street
Redding, CA 96001
+1 530 395 2500

ADVANCED TUTORIAL: ACQUIRING DATA

To acquire data from the device LabVIEW elements will be used in conjunction with ActiveX controls. The DataRay device can be controlled using invoke and property nodes via the **GetData Control** object. Additionally, data can be acquired with the **GetData Control** class. Two camera examples are included, one which records the camera pixel data and the other which records button data. Additionally, an example recording profile data and opening both M^2 and divergence dialog boxes with the BeamMap is included. The instructions and screenshots included provide instructions for creating a simple VI with the same principles as the VIs that can be downloaded. With previous LabVIEW experience, the more complex, downloadable examples should be easily understood.

EXAMPLE VI: WINCAMD CAMERA

To return the pixel data from the camera, attach an invoke node to the **GetData Control** object and select **GetWinCamDataAsVariant** from the methods listed. This will provide the camera pixel data in a variant data format. To convert the data, use the **Variant to Data Function** from the **Variant Functions** palette. Use a constant array of U16 as the type input on the **Variant to Data Function**. Using double or other signed variables for the input type will induce errors in the sign values produced because the data is unsigned. The pixel data is returned from the **Variant to Data Function** as a 1D U16 array. The 1D array consists of horizontal pixel rows concatenated together, so it will need to be converted to a 2D array matching the pixel layout of the camera to be useful. To begin, the horizontal and vertical pixel counts must be found. Attach three invoke nodes to the **GetWinCamDataAsVariant** successively. For the first invoke node, select **GetHorizontalPixels** from the methods available. For the second invoke node select **GetVerticalPixels** and for the third invoke node select **CaptureIsFullResolution. GetHorizontalPixels** and **GetVerticalPixels** return the horizontal and vertical pixel counts of the capture block respectively; they do not directly correspond to the number of data points collected. To know how many data points are collected, you need to know if the camera is operating in **Fast** or **Full** mode. **CaptureIsFullResolution** returns a -1 if the device is in **Full** mode, and a 0 if the device is in **Fast** mode. In **Fast** mode, every other pixel along each axis is recorded and so both pixel counts must be divided by two to match the recorded data points. Finally, a for loop takes the data array and parses it using the corrected pixel counts and **Array Subset Function**. Since the 1D array consists of horizontal pixel rows concatenated together, the length of the array subset is equal to the corrected horizontal pixel count, while the number of iterations of the for loop is equal to the corrected vertical pixel count.

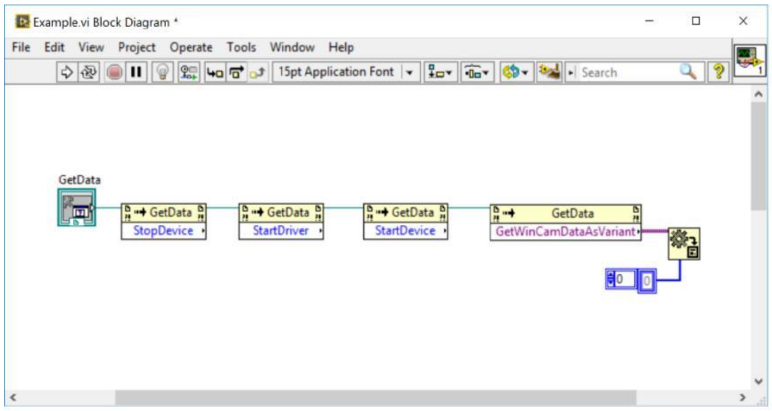

Example.vi Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
GetData
GetData StopDevice
GetData StartDriver
GetData StartDevice
GetData GetWinCamDataAsVariant
0
0

Figure 11

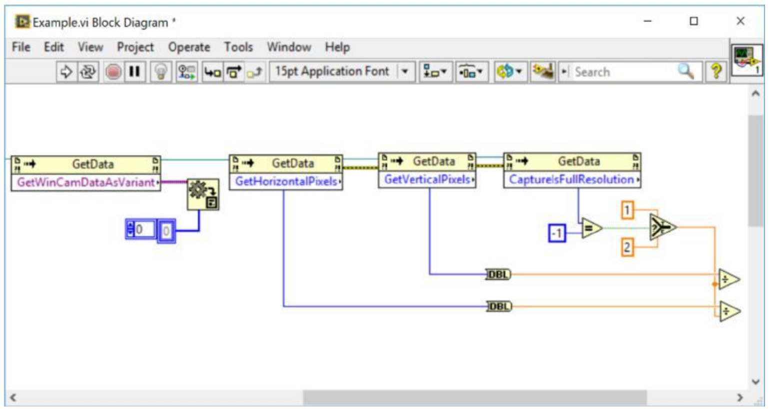

Example.vi Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
GetData GetWinCamDataAsVariant
GetData GetHorizontalPixels
GetData GetVerticalPixels
GetData CaptureIsFullResolution
0
0
-1
1
2
DBL
DBL

Figure 12

support@dataray.com
1675 Market Street
Redding, CA 96001
+1 530 395 2500

EXAMPLE VI: WINCAMD BUTTON

To acquire data from a button, the **GetData Control** object must be wired to an invoke node. Select **GetOcxResult** from the list of available methods on the invoke node (see Figure 13). The ID of the button whose data is requested must be wired to the **IndexToValue** input on the invoke node. The button ID can be found via the instructions in the Extending the GUI section. The invoke node will now return the button value. The majority of the time, the **GetData Control** object can return a button's data without the button ActiveX object being placed on the front panel.

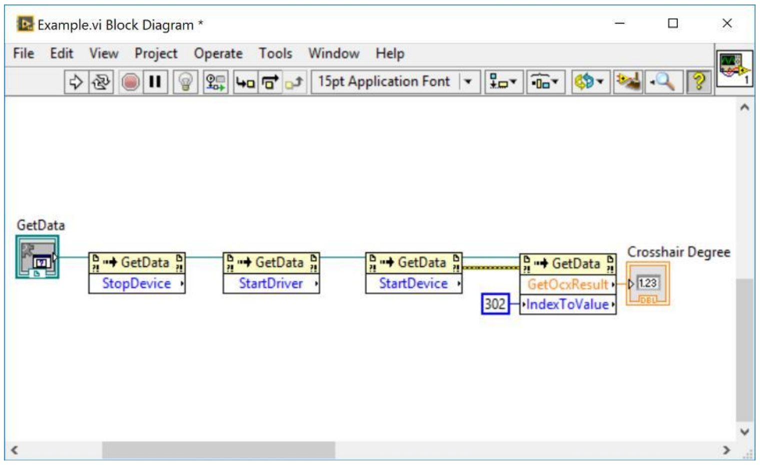

Example.vi Block Diagram *
File Edit View Project Operate Tools Window Help
15pt Application Font
GetData
GetData StopDevice
GetData StartDriver
GetData StartDevice
GetData GetOcxResult IndexToValue
302
Crosshair Degree
1.23

Figure 13

EXAMPLE VI: WINCAMD EVENT

Often times, the user wishes to record data based on when a new frame has entered the camera. In this case, we can use the **Register Event Callback** function for ActiveX (see Figure 14). Add the **Register Event Callback** function to the block diagram and wire the **GetData Control** reference to the **Event** input. Then select the **DataReady** event from the list of events provided. The **DataReady** event executes the event callback function whenever a new frame of data is available. Next, wire the **GetData Control** to the **To Variant** function and then to the **User Parameter** input (which requires a variant data type). The **Register Event Callback** calls a reference VI when the event is triggered, but the **User Parameter** input must be wired before creating the reference VI. Once all the previous steps have been completed, right click the **VI Ref** input and select Create Callback VI from the list provided. A reference VI will be created with the Event Common Data, Control Ref, and Variant variables inside (see Figure 15). This VI will be called each time a new event occurs, that is when the next frame is available. Next, in the reference VI convert the variant data back into the ActiveX reference data type. Now invoke and property nodes can be used with the **GetData Control** reference. However, the invoke nodes must be programmed with the correct method on the main VI and then copied over to the reference VI since the invoke and property nodes won't automatically populate a list of available methods. A while loop is included in the main VI to keep the program running and look for events and after the loop has been stopped, the **Register for Events** function closes the event reference.

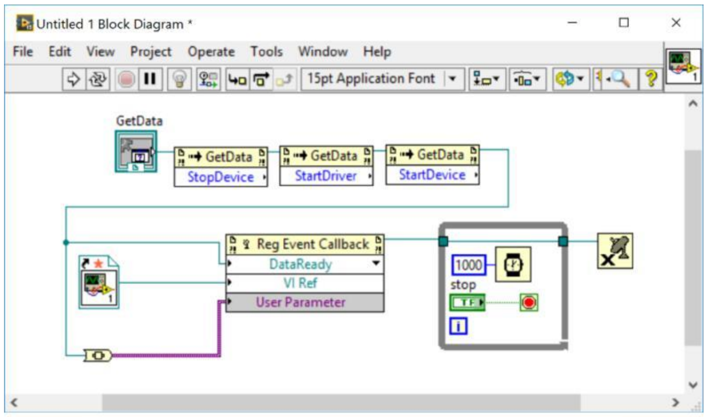


Figure 14: Main VI

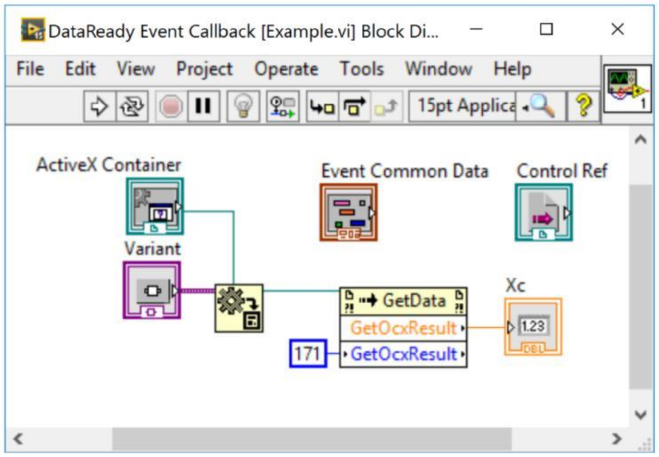

DataReady Event Callback [Example.vi] Block Di...
File Edit View Project Operate Tools Window Help
15pt Applica
ActiveX Container
Event Common Data
Control Ref
Variant
GetData GetOcxResult GetOcxResult
171
Xc
1.23

Figure 15: Reference VI

EXAMPLE VI: BEAMMAP

To acquire data from the BeamMap slit profiler, a similar process is used. The BeamMap does not have a camera from which data can be taken and instead only the integrated intensity along the x and y axes is measured. Information from these axes can be recorded by connecting an invoke node to the respective profile object (see Figure 16). After connecting the profile object to the invoke node, select **GetProfileDataAsVariant** from the list of methods provided. To convert the data, use the **Variant to Data Function** from the **Variant Functions** palette. Use a constant array of U16 as the type input on the **Variant to Data Function**. Using double or other signed variables for Figure 16 the input type will induce errors in the sign values produced because it is unsigned. The pixel data is returned from the **Variant to Data Function** as a 1D U16 array. The 1D array contains all the intensity values from the profile. The step size can also be returned by connecting a second invoke node to the profile object and selecting **GetStepSize** from the list of methods. With both the intensity data and the step size, the data may be recorded or graphed. In the example, the option to add the step size to the beginning of the 1D array is included. Additionally, ability to take button data is included in the VI. Lastly, to open up the M^2 and Divergence dialog boxes, wire two invoke nodes to the **GetData Control** object (see Figure 17). From the methods available select **IsMSquaredOpen** from the first invoke node, and **IsDivergenceOpen** from the second. To open a dialog box, wire a 1 to the invoke node input. Likewise, to close the dialog box wirea 0 to the invoke node input. If the dialog boxes fail to appear, or Figure 17 appears with strange dimensions, try lowering the display resolution.

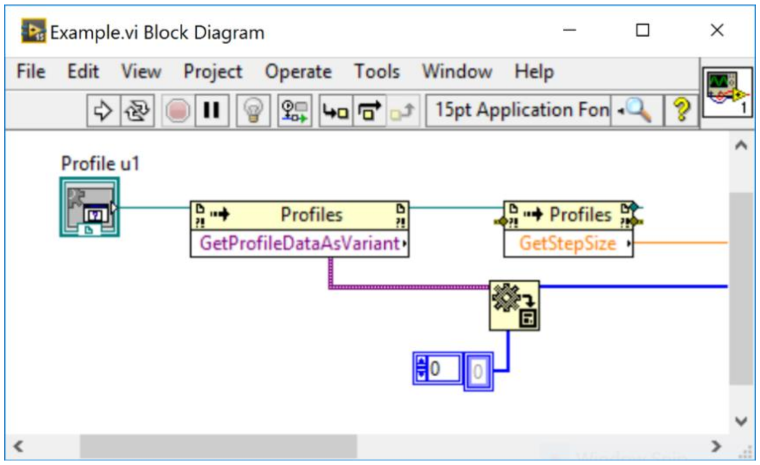


Figure 16

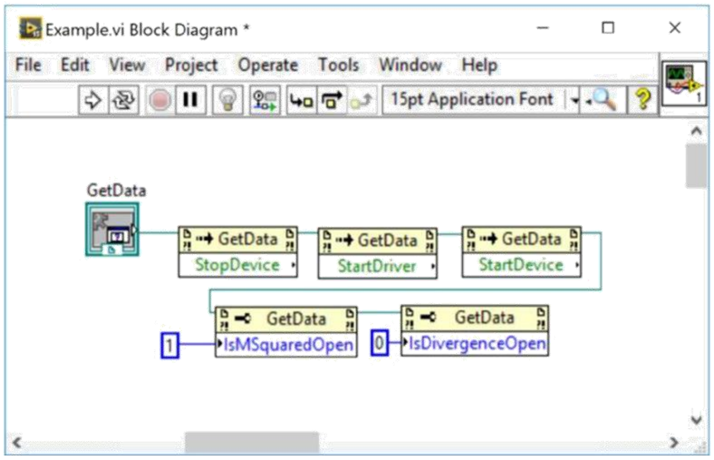
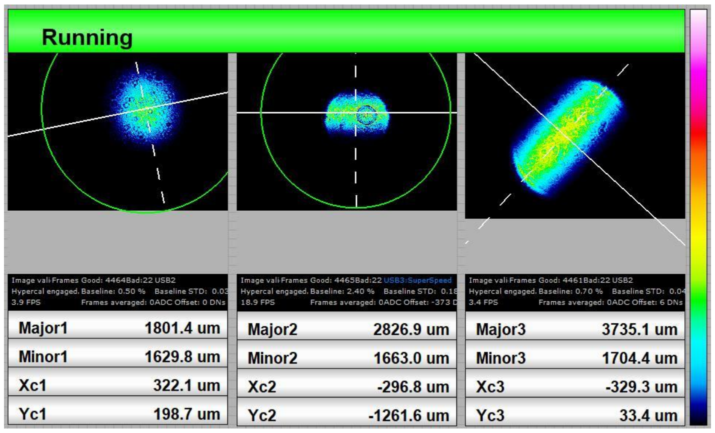


Figure 17

EXAMPLE VI: WINCAMD MULTICAMERA



```
Image vali Frames Good: 4464Bad:22 USB2
Hypercal engaged. Baseline: 0.50 %    Baseline STD:  0.03
3.9 FPS          Frames averaged: 0ADC Offset: 0 DNs
```

```
Image vali Frames Good: 4465Bad:22 USB3:SuperSpeed
Hypercal engaged. Baseline: 2.40 %    Baseline STD:  0.18
18.9 FPS          Frames averaged: 0ADC Offset: -373 D
```

```
Image vali Frames Good: 4461Bad:22 USB2
Hypercal engaged. Baseline: 0.70 %    Baseline STD:  0.04
3.4 FPS          Frames averaged: 0ADC Offset: 6 DNs
```

Major1	1801.4 um	Major2	2826.9 um	Major3	3735.1 um
Minor1	1629.8 um	Minor2	1663.0 um	Minor3	1704.4 um
Xc1	322.1 um	Xc2	-296.8 um	Xc3	-329.3 um
Yc1	198.7 um	Yc2	-1261.6 um	Yc3	33.4 um

Figure 18

The OCX can run up to four concurrent camera-based DataRay devices. This example VI allows the user to run and record measurements from three devices in parallel (see Figure 18). First, add a **GetData Control** and a **PaletteBar Control** by using the methods described in the basic tutorial. Next, create a **CCDimage Control**, but repeat the process two additional times to create a total of three **CCDimage Controls**. Attach three invoke nodes to the **GetData Control**. From the first invoke node, select **StopDevice**. From the second invoke node, select **StartDriver**. From the third invoke, select **SetCurrentDevice** method and set the DeviceType variable to **10** (see Figure 19). This value indicates that we will be using three DataRay devices concurrently. When using two concurrent devices, this value must be set to 9. When using four concurrent devices, this value must be set to 11. For more information on the **SetCurrentDevice** function, please see the OCX documentation. Instead using an additional invoke node to programmatically start the device, we will include a Status Button that allows us to start and stop all the devices from the GUI. To create a Status Button, create a **Button Control** and change the ButtonID to **297**. Finally, we will include additional **Button Controls** that will display the major beam diameter, minor beam diameter, and the X and Y coordinates of the beam centroid for each device. When using **Button Controls** to display beam metrics in multicamera mode, it is necessary to choose button indexes that apply to the camera of interest. When using multicamera mode, we are typically interested in button indexes **337-424**. For example, to display the major beam diameter recorded by the **first camera**, use button index **345**. To display the major beam diameter recorded by the **second camera**, use button index **346**. All relevant button indexes are available here.

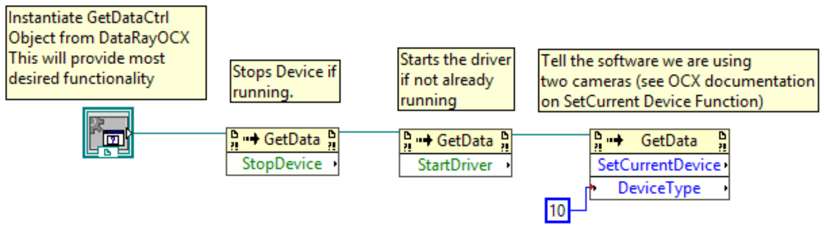

Instantiate GetDataCtrl Object from DataRayOCX This will provide most desired functionality
Stops Device if running.
Starts the driver if not already running
Tell the software we are using two cameras (see OCX documentation on SetCurrent Device Function)
GetData
StopDevice
GetData
StartDriver
GetData
SetCurrentDevice
DeviceType
10

Figure 19