

Interfacing iDataRay.exe to LabVIEW™

LabVIEW™ is a registered trademark of National Instruments.

Applies to:

- The BeamMap series: BeamMap, BeamMap-C, BeamMap Collimate, Beam'R
- BeamScope-P7
- The WinCamD series: WincamD and TaperCamD
- Versions 5.00L and higher of **iDataRay.exe**. Does not apply to lower software revisions.

You need to know that:

- a) **iDataRay.exe** must be installed before you attempt to link to LabVIEW.
- b) ACTIVE X architecture allows the software to simultaneously run on-screen & interface to other software.
- c) These instructions were developed on LabVIEW 6.0. Higher or lower numbered LabVIEW Versions may show, &/or require, slightly different steps &/or screens.
- d) We are experts in our software. We are not LabVIEW experts. To use iDataRay with LabVIEW, you must at least be LabVIEW-competent *before* you attempt to use this Application Note.
- e) An example VI (Virtual Instrument) for Beam'R is available at the website and shown at the end of this note.
- f) Please send suggestions on additions/deletions/changes to: **support@dataray.com**

Create an Ocx Access library

Here are the steps to create the 'basic' LabVIEW elements: In this example we will build a Beam'R parts. First we need a library to access the DataRay Ocx, and to create a DataRay control.

Within LabVIEW, click the **New VI** button.

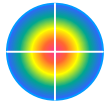


Step 1: Add the DataRayOcx to LabView VI

Select: Controls (right click) → ActiveX → Automation Refnum.

Right click the Automation Refnum icon and select 'Select ActiveX Class' then select DATARAYOCXlib.GetData if not listed, select browse and go to Program Files\DataRay\ and select DataRayOcx.Active X.

At this point DATARAYOCXlib.GetData icon should be in your VI.



Step 2: Add a DataRayOcx Method to LabView VI

Right click the Automation Refnum icon and select 'Find Terminal'.

You now should be in the diagram window. Right click the DATARAYOCXlib.GetData icon and select 'Create Method' (Note: the .Getdata may not be completely shown until you increase the text region size), select 'StartDriver' from the list.

To summarize: Right click icon → Create → Method → StartDriver.

To wire this method, simply attach the DATARAYOCXlib.GetData icon to the upper left terminal of StartDriver.

Now let's add another method 'SetCurrentDevice'.

Right click GetData icon → Create → Method → SetCurrentDevice.

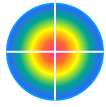
To wire this method, attach output *reference* terminal of StartDriver to the input *reference* terminal of SetCurrentDevice. Then add a *constant* to the *DeviceType* input terminal of SetCurrentDevice. **It is assumed that you know and understand LabView lingo, in this case 'add a constant'.**

In the DataRay directory there should be a file named 'datastructs.h' which contains all the structures and enumerations used by the DataRay OCX.

Here is the 'Selected_Device' enumeration found in 'datastructs.h'.

```
enum _Selected_Device_ {  
    IS_BEAMSCOPE=1,  
    IS_BEAMR=2,  
    IS_BEAMMAP=3,  
    IS_BEAMC=4,  
    IS_WINCAM=5,  
    IS_TWOD_SCAN=6,  
    IS_WINCAM_LOG=7,  
    IS_SPARE3=8,  
    IS_SPARE4=9,  
    LAST_DEVICE=10,  
};
```

Now change the *DeviceType constant* to the desired DataRay device. This note will assume the device is a BeamR or IS_BEAMR, thus change the constant to 2. This function is now complete.



Step 3: Add a DataRayOcx Button to LabView VI

Go back to the control window.

Select: Controls (right click) → ActiveX → Container.

Right click the new Container and select 'Insert ActiveX Object' → 'Button Control'

Add another DataRay button now in the same manner. Note: **You CANNOT copy and paste controls!**

Right click the first button and select 'Button' → Properties.

A dialog should appear, that allows us to define which button we want.

Here is an excerpt from the *IndexToDataTestParameters* enumeration:

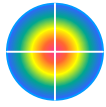
```
Last_PARAMETER_ID = 229,  
  
SetClipLevel1 = 230,  
SetClipLevel2 = 231,  
PeakButton = 232,  
StatusButton = 233,  
ZeroButton = 234,  
SpacerButton1 = 235,  
SpacerButton2 = 236,  
ZoomButton = 237,  
CrossHairButton = 238,  
AuxStageScale_2D = 239,  
TwoDsetup_2D = 240,  
SetReference_2D = 241,  
Reference_2D = 242,  
HomeStage_2D = 243,  
Scan_2D = 244,  
DoSearch = 245,  
FindCenter_2D = 246,  
ReDo2D = 247,  
Last_ID = 248,  
};
```

We want this button to be the 'Status Button' so we set it to 233 and press APPLY, then press OK.

You can now resize the button to what ever you wish.

Change the second button to 245 'DoSearch'.

Note that there are over 200 possible values; every button is used in the main DataRay.exe program thus the program should be used as a guideline on how they are used.



Step 4: Add a DataRayOcx Profile to LabView VI

Go back to the control window.

Select: Controls (right click) → ActiveX → Container.

Right click the new container and select 'Insert ActiveX Object' → 'Profiles Control'

Repeat to create another profile.

The 'Profiles Control' is managed in a very similar way to the 'Button Control'.

Right click the first profile and select 'Profiles' → Properties.

A dialog should appear, that allows us to define which profile we want.

There is a list box, select the profile desired; In this case we will create two profiles, BeamR_X and BeamR_Y.

You can now resize the button to what ever you wish.

Step 5: Accessing data from the DataRayOcx via the LabView VI

Here is where people get lost, but don't worry it is very simple!

Go back to the Diagram window.

Right click the DATARAYOCXlib.GetData icon
→ Create → Method → GetOcxResult.

Note: There can be as many instances of this Method as desired and you can use 'Paste & Copy' to duplicate them.

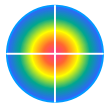
To wire this Method, attach the Output *reference* terminal of StartDriver to the Input *reference* terminal of GetOcxResult. Then add a *Constant* to the *IndexToValue* input terminal of GetOcxResult. **Again it is assumed that you know and understand LabView lingo, in this case 'add a Constant'.**

Add a Indicator to the GetOcxResult terminal.

Now reference the enumeration **IndexToDataTestParameters** in datastructs.h. Let's select Xc (the beam position in X), that would be Xc_BeamR (86), so change the constant to 86 and change the Method label to Xc or whatever.

That's it!

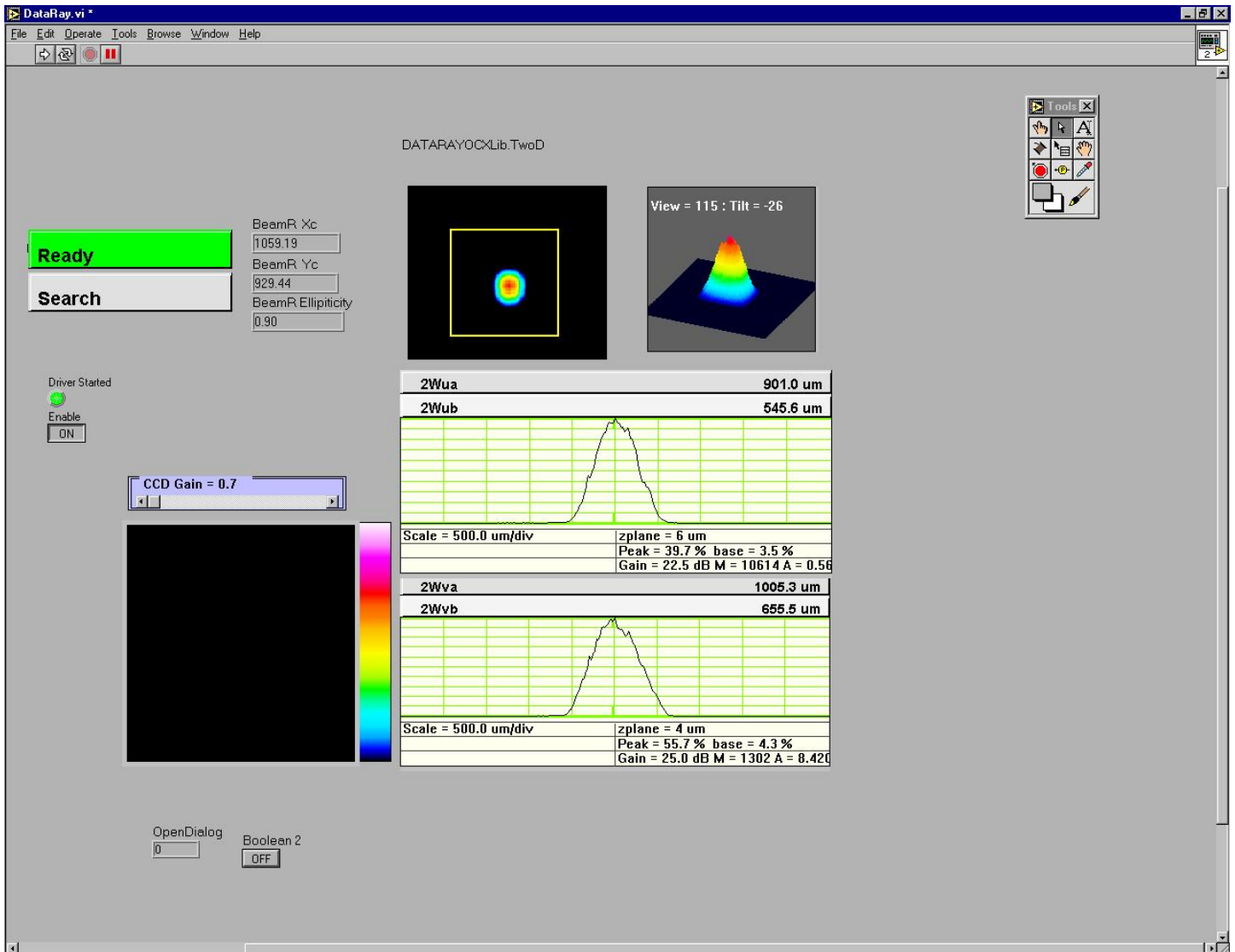
Note that there is not a direct way to transfer arrays of data like raw profiles or images. However, we could easily add this given an example how LabView does it. Send specific software requests to **support@dataray.com**. Please be VERY specific and give examples.

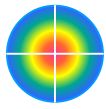


Example VI for Beam'R

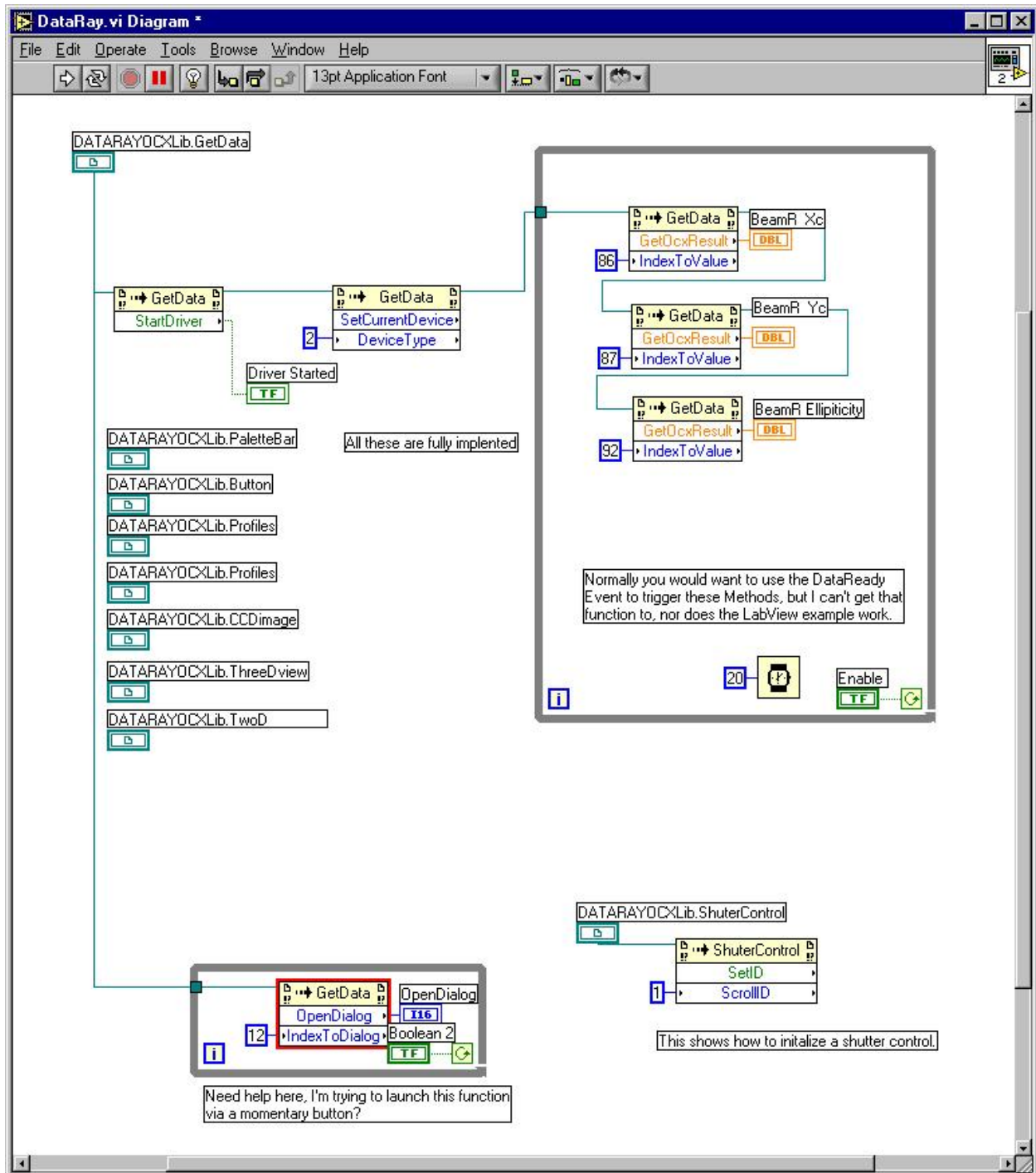
This example VI (Virtual Instrument) for Beam'R is available at the website. The same principles apply to VI's for all other products supported by the DataRay software.

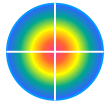
In LabVIEW it looks like this:





The VI diagram is shown below.





Other Controls:

We covered Button and Profiles Controls, but there are others:

TwoD Control Installs just like the Button or Profiles except there no need to set Properties.

ThreeD Control Installs just like the Button or Profiles except there no need to set Properties.

PaletteBar Control Installs just like the Button or Profiles except there no need to set Properties.

CCDimage Control WinCamD only

Installs just like the Button or Profiles except there no need to set Properties.

Shutter Control WinCamD only

Installs just like the Button or Profiles except instead of setting a Property.

You must create and run the SetID Method. (Example shown in DataRay.vi)

The possible IDs are defined:

```
#define _TRIGGER_    0  
#define _GAIN_      1  
#define _SHUTTER_   2
```

Known Problems

1) I can't get the ActiveX callback mechanism (An OCX Event) to work, even though LabView can and does display the parameters correctly. Since the DataRay OCX Event **DataReady** is fired whenever new data is available, this would be the preferred method to use.

LabView supplies an example of this, but alas, it doesn't work either. We were forced to use the timer to update the data.

P.S. Events do work in the EXE program and in Visual Basic.

2) It's a bit confusing that OCX objects are active even while in edit mode. This is because once the OCX is launched by LabView (or any other application) the OCX is active and remains active for the life of the calling application and there is no way to close it other than closing the calling application ... except of course in C++ that can do anything.

Still need more help?

DataRay OCX stuff: **support@dataray.com**

LabView stuff: **PLEASE contact National Instruments ...**