

## Interfacing to DataRay OCX Software - Rev 3, Feb '06

**IN THIS REVISION:** Page 32 lists new Methods added in this Revision, followed by a complete listing of the DataRayOcx.odl : type library source for ActiveX Control project.

### Applies to:

- The BeamMap series: BeamMap, BeamMap-C, BeamMap Collimate, Beam'R
- BeamScope-P7
- The WinCamD series: WinCamD 'Classic', -Uxx and TaperCamD versions
- Versions 6.00N11 and higher of **iDataRay.exe**. May not apply to lower software revisions
- Interfacing to Microsoft C++ and Microsoft Visual Studio, plus, obliquely, to LabView®.

### You need to know that:

- If you print this document, use a **color** printer or information will be lost.
- It is *necessary* that you first become familiar with the normal operation of the DataRay software for the product that you are using. This will dramatically simplify your interpretation of the command list. Items which may initially seem obscure will become more self-explanatory.
- To determine the name of any result that you see on the screen, left-click on it to see the Pass-Fail dialog box and the name used by the software.
- We are experts in our software and C++. We may not be experts in the software to which you are interfacing. To use DataRayOcx with other software, you must at least be competent in the other software *before* you attempt to use this Application Note. We have neither the resources nor the skills to teach software interfacing.
- Source Code is supplied in the full version of the software as **DataRaySource.zip**.
- **iDataRayOcx.exe** must be installed before you attempt to link to other software.
- Only Visual C++ and Visual Basic are described, but the mechanism is similar for LabVIEW™, VBA and other platforms. [Note that Microsoft Visual Studio has now replaced Microsoft Visual Basic. We recommend that unless you have really good reasons for starting something new in Visual Basic, you invest your time in Visual Studio.]
- In Microsoft C++ the OCX *MUST* be re-installed each and *EVERY* time a Method or Property is changed. If the OCX is not re-installed, you *CAN* connect to some of the OCX objects but others will throw out errors such as **Method not found**, even though they do exist, but since the OCX changed, they are now scrambled.  
  
This *DOES NOT* make sense, (and is *NOT* true with standard DLLs). Take the issue up with Microsoft.  
  
To Re-install the OCX, simply follow the same procedure that you originally followed, except that the software will ask if you can overwrite the link files? Answer YES.
- This is an evolving document. Please send suggestions for additions/deletions/changes to: **support@dataray.com**

### Microsoft Visual Studio

If not already installed, install and *understand* Microsoft Visual Studio *first*.

With C++, see the **DataRaySource.zip** file in the DataRay directory. When expanded into a working directory, this expands into a full Visual Studio project.

- Double click **DataRay.dsw** file to open the project.
- File **MainFrm.cpp** contains the code that installs and links to the DataRay OCX.

- See **InstallOcx()** for the simple code to create an instance of the OCX.

**Note:** Only one instance of the DataRay OCX should be open at one time, this includes the **DataRay.exe**. Also note that this project always contains ALL of the code that interfaces to the DataRay OCX in the DataRay.exe release.

## Microsoft Visual Basic

If not already installed, install and *understand* Microsoft Visual Basic *first*.

Open Visual Basic, and open a new **Standard Exe** project, then save the project to your desired name.

## DataRay OCX Objects

Open menu item **Project→Components** then select **Insertable Objects** then select **DataRayOcx ActiveX Control Module**, press **OK**. This inserts ten **Objects** into the tool bar.

The DataRay OCX **Objects** are:

- **GetData**      The only interface for code.

**Visual Objects**      Drag onto Screen

- **Buttons**
- **Profiles**
- **Palette**
- **2D Image**      The only Essential Object for use of a Camera
- **3D Image**
- **CCD View**
- **Shutter Control**
- **Trigger Control**
- Blank (Spare)

These **Objects** are described below. Here's what the abbreviations below mean:

BS = BeamScope  
 BM = BeamMap  
 BR = Beam'R  
 BC = BeamMap Collimate  
 WC = WinCam  
 DIV = Divergence  
 X or U = the horizontal axis  
 Y or V = the vertical axis


(BeamMap)

ZM2      = Zero Minus 2  
 ZM1      = Zero Minus 1  
 Z        = Zero  
 ZP1      = Zero Plus 1  
 ZP2      = Zero Plus 2  
 P45      = Slit at Plus 45 degrees  
 M45      = Slit at Minus 45 degrees

(BeamMap Collimate)

U1,V1    = The First Z plane  
 U2,V2    = The Second Z plane  
 U3,V3    = The Third Z plane  
 U4,V4    = The Fourth Z plane



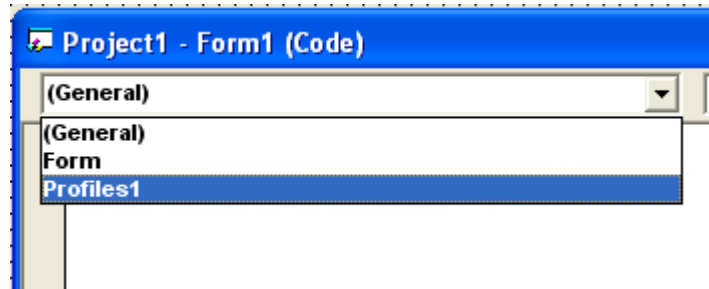
**Visual Basic** Select the gadget  on the tool bar, then place and size the Profile Object in the form. The first Profile Object becomes Profiles1 in the project.

Then define ID:

```
Profiles1.SetMyID(BS_PROFILE_S)
```

Where are the IDs defined?

Add Class Module:



**Project->Add Module -> DataRayOcx (DataRayOcx.sls)**

Open the DatarayOcx.cls mode and find **DataRay\_Profiles:**

Enum DataRay\_Profiles

```
DEFAULT_PROFILE = 0
BS_PROFILE_X = 1
BS_PROFILE_Y = 2
BS_PROFILE_S = 3
BR_PROFILE_X = 4
BR_PROFILE_Y = 5
BM_PROFILE_ZM2 = 6
BM_PROFILE_ZM1 = 7
BM_PROFILE_Z = 8
BM_PROFILE_ZP1 = 9
BM_PROFILE_ZP2 = 10
BM_PROFILE_M45 = 11
BM_PROFILE_P45 = 12
BM_PROFILE_Z2 = 13

BC_PROFILE_U1 = 14
BC_PROFILE_V1 = 15
BC_PROFILE_U2 = 16
BC_PROFILE_V2 = 17
BC_PROFILE_U3 = 18
BC_PROFILE_V3 = 19
BC_PROFILE_U4 = 20
BC_PROFILE_V4 = 21

WC_PROFILE_X = 22
WC_PROFILE_Y = 23
WC_DIV_X = 24
WC_DIV_Y = 25
```

End Enum

**2) Buttons:** This object is used to insert the DataRay Button objects as shown below:



**Visual C++.** A profile is first automatically inserted into the project as **ocxbutton.cpp**, as shown here.

Menu item **Project->Add to Project->Components and Controls**, then **Registered ActiveX Controls** then **Button Control**.

```
COcxButton    *aButton = 0 ;
aButton = new COcxButton() ;
aButton ->SetButtonID(SetClipLevel1) ;
```

Open file **datastructs.h**, search for **IndexToDataTestParameters** or **StatusButton**:

**Note:** As indicated by **IndexToDataTestParameters**, in most cases, each button displays a particular parameter and is also used to get that result. But which one to use? In the main **DataRay.exe** program, **LEFT CLICK** the button you want, the name of that button is displayed!

```
enum IndexToDataTestParameters {

    BlankParameters    = 0,
    DefaultParameters = 1,

// Beam'R

    u_BeamR_Width_at_Clip_1    = 2,
    u_BeamR_Width_at_Clip_2    = 3,
    u_BeamR_GFit               = 4,
    u_BeamR_TopHat             = 5,
    v_BeamR_Width_at_Clip_1    = 6,
    v_BeamR_Width_at_Clip_2    = 7,
    v_BeamR_GFit               = 8,
    v_BeamR_TopHat             = 9,
    v1_BeamC_Width_at_Clip_1   = 10,
    v1_BeamC_Width_at_Clip_2   = 11,
    v1_BeamC_GFit              = 12,
    v1_BeamC_TopHat            = 13,
    u1_BeamC_Width_at_Clip_1   = 14,
    u1_BeamC_Width_at_Clip_2   = 15,
    u1_BeamC_GFit              = 16,
    u1_BeamC_TopHat            = 17,
    v2_BeamC_Width_at_Clip_1   = 18,
    v2_BeamC_Width_at_Clip_2   = 19,
    v2_BeamC_GFit              = 20,
    v2_BeamC_TopHat            = 21,
    u2_BeamC_Width_at_Clip_1   = 22,
    u2_BeamC_Width_at_Clip_2   = 23,
    u2_BeamC_GFit              = 24,
    u2_BeamC_TopHat            = 25,
    v3_BeamC_Width_at_Clip_1   = 26,
    v3_BeamC_Width_at_Clip_2   = 27,
    v3_BeamC_GFit              = 28,
    v3_BeamC_TopHat            = 29,
    u3_BeamC_Width_at_Clip_1   = 30,
    u3_BeamC_Width_at_Clip_2   = 31,
    u3_BeamC_GFit              = 32,
    u3_BeamC_TopHat            = 33,
```

v4_BeamC_Width_at_Clip_1	= 34,
v4_BeamC_Width_at_Clip_2	= 35,
v4_BeamC_GFit	= 36,
v4_BeamC_TopHat	= 37,
u4_BeamC_Width_at_Clip_1	= 38,
u4_BeamC_Width_at_Clip_2	= 39,
u4_BeamC_GFit	= 40,
u4_BeamC_TopHat	= 41,
minus2_BeamMap_Width_at_Clip_1	= 42,
minus2_BeamMap_Width_at_Clip_2	= 43,
minus2_BeamMap_GFit	= 44,
minus2_BeamMap_TopHat	= 45,
minus1_BeamMap_Width_at_Clip_1	= 46,
minus1_BeamMap_Width_at_Clip_2	= 47,
minus1_BeamMap_GFit	= 48,
minus1_BeamMap_TopHat	= 49,
zero_BeamMap_Width_at_Clip_1	= 50,
zero_BeamMap_Width_at_Clip_2	= 51,
zero_BeamMap_GFit	= 52,
zero_BeamMap_TopHat	= 53,
plus1_BeamMap_Width_at_Clip_1	= 54,
plus1_BeamMap_Width_at_Clip_2	= 55,
plus1_BeamMap_GFit	= 56,
plus1_BeamMap_TopHat	= 57,
plus2_BeamMap_Width_at_Clip_1	= 58,
plus2_BeamMap_Width_at_Clip_2	= 59,
plus2_BeamMap_GFit	= 60,
plus2_BeamMap_TopHat	= 61,
plus45_BeamMap_Width_at_Clip_1	= 62,
plus45_BeamMap_Width_at_Clip_2	= 63,
plus45_BeamMap_GFit	= 64,
plus45_BeamMap_TopHat	= 65,
minus45_BeamMap_Width_at_Clip_1	= 66,
minus45_BeamMap_Width_at_Clip_2	= 67,
minus45_BeamMap_GFit	= 68,
minus45_BeamMap_TopHat	= 69,
u_BeamScope_Width_at_Clip_1	= 70,
u_BeamScope_Width_at_Clip_2	= 71,
u_BeamScope_GFit	= 72,
u_BeamScope_TopHat	= 73,
v_BeamScope_Width_at_Clip_1	= 74,
v_BeamScope_Width_at_Clip_2	= 75,
v_BeamScope_GFit	= 76,
v_BeamScope_TopHat	= 77,
u_WinCamD_Width_at_Clip_1	= 78,
u_WinCamD_Width_at_Clip_2	= 79,
u_WinCamD_GFit	= 80,
u_WinCamD_TopHat	= 81,
v_WinCamD_Width_at_Clip_1	= 82,
v_WinCamD_Width_at_Clip_2	= 83,
v_WinCamD_GFit	= 84,
v_WinCamD_TopHat	= 85,
Xc_BeamR	= 86,
Yc_BeamR	= 87,
uM2_NA_BeamC_alt	= 88,
vM2_NA_BeamC_alt	= 89,
Uniformity_WinCamD	= 90,
Ewidth_WinCamD	= 91,
Ellipticity_BeamR	= 92,

Power_BeamR	= 93,	
Xc1_BeamC	= 94,	
Yc1_BeamC	= 95,	
Xg1_BeamC_not_used	= 96,	
Yg1_BeamC_not_used	= 97,	
Xp1_BeamC_not_used	= 98,	
Yp1_BeamC_not_used	= 99,	
Xc2_BeamC	= 100,	
Yc2_BeamC	= 101,	
uMsquared_BeamC_Alt	= 102,	
uM2_Zo_BeamC_Alt	= 103,	
Xp2_BeamC_not_used	= 104,	
Yp2_BeamC_not_used	= 105,	
Xc3_BeamC	= 106,	
Yc3_BeamC	= 107,	
vMsquared_BeamC_Alt	= 108,	
vM2_Zo_BeamC_Alt	= 109,	
Xp3_BeamC_not_used	= 110,	
Yp3_BeamC_not_used	= 111,	
Xc4_BeamC	= 112,	
Yc4_BeamC	= 113,	
Xg4_BeamC_not_used	= 114,	
Yg4_BeamC_not_used	= 115,	
Xp4_BeamC_not_used	= 116,	
Yp4_BeamC_not_used	= 117,	
Ellipticity_BeamC	= 118,	
Power_BeamC	= 119,	
Xc_BeamMap	= 120,	
Yc_BeamMap	= 121,	
Xg_BeamMap_not_used	= 122,	// Reserved
Yg_BeamMap_not_used	= 123,	// Reserved
Xp_BeamMap_not_used	= 124,	// Reserved
Yp_BeamMap_not_used	= 125,	// Reserved
Ellipticity_BeamMap	= 126,	
Power_BeamMap	= 127,	
DivergenceNA_BeamMap	= 128,	
DivergenceDegrees_BeamMap	= 129,	
DivergenceRadians_BeamMap	= 130,	
Msquared_BeamMap	= 131,	
M2_2Wo_BeamMap	= 132,	
M2_Zo_BeamMap	= 133,	
M2_Zr_BeamMap	= 134,	
M2_Theta_BeamMap	= 135,	
M2_NA_BeamMap	= 136,	
Xc_BeamScope	= 137,	
Yc_BeamScope	= 138,	
Xg_BeamScope	= 139,	
Yg_BeamScope	= 140,	
Xp_BeamScope	= 141,	
Yp_BeamScope	= 142,	
Ellipticity_BeamScope	= 143,	
Power_BeamScope	= 144,	
uDivergenceNA_BeamC	= 145,	
uDivergenceDegrees_BeamC	= 146,	
uDivergenceRadians_BeamC	= 147,	
uMsquared_BeamC	= 148,	
uM2_2Wo_BeamC	= 149,	
uM2_Zo_BeamC	= 150,	
uM2_Zr_BeamC	= 151,	

uM2_Theta_BeamC	= 152,
uM2_NA_BeamC	= 153,
vDivergenceNA_BeamC	= 154,
vDivergenceDegrees_BeamC	= 155,
vDivergenceRadians_BeamC	= 156,
vMsquared_BeamC	= 157,
vM2_2Wo_BeamC	= 158,
vM2_Zo_BeamC	= 159,
vM2_Zr_BeamC	= 160,
vM2_Theta_BeamC	= 161,
vM2_NA_BeamC	= 162,
Xc_WinCamD	= 163,
Yc_WinCamD	= 164,
Xg_WinCamD	= 165,
Yg_WinCamD	= 166,
Xp_WinCamD	= 167,
Yp_WinCamD	= 168,
Ellipticity_WinCamD	= 169,
Power_WinCamD	= 170,
Oreintation_WinCamD	= 171,
MajorWidth_WinCamD	= 172,
MinorWidth_WinCamD	= 173,
MeanWidth_WinCamD	= 174,
Peak_WinCamD	= 175,
AverageFluence_WinCamD	= 176,
uM2_M2_BeamScope	= 177,
uM2_2Wo_BeamScope	= 178,
uM2_Zo_BeamScope	= 179,
uM2_Zr_BeamScope	= 180,
uM2_Theta_BeamScope	= 181,
uM2_NA_BeamScope	= 182,
vM2_M2_BeamScope	= 183,
vM2_2Wo_BeamScope	= 184,
vM2_Zo_BeamScope	= 185,
vM2_Zr_BeamScope	= 186,
vM2_Theta_BeamScope	= 187,
vM2_NA_BeamScope	= 188,
ID_WANDER	= 189, // Special case
PointingX_BeamMapC	= 190,
PointingY_BeamMapC	= 191,
PointingX_BeamMap	= 192,
Msquared_BeamMap_Alt	= 193,
M2_Zo_BeamMap_Alt	= 194,
M2_Theta_BeamMap_Alt	= 195,
M2_Zo_BeamMap_Alt2	= 196,
u_BeamR_GFitWidth	= 197,
v_BeamR_GFitWidth	= 198,
v1_BeamC_GFitWidth	= 199,
u1_BeamC_GFitWidth	= 200,
v2_BeamC_GFitWidth	= 201,
u2_BeamC_GFitWidth	= 202,
v3_BeamC_GFitWidth	= 203,
u3_BeamC_GFitWidth	= 204,
v4_BeamC_GFitWidth	= 205,
u4_BeamC_GFitWidth	= 206,
minus2_BeamMap_GFitWidth	= 207,
minus1_BeamMap_GFitWidth	= 208,
zero_BeamMap_GFitWidth	= 209,
plus1_BeamMap_GFitWidth	= 210,

```

plus2_BeamMap_GFitWidth      = 211,
plus45_BeamMap_GFitWidth     = 212,
minus45_BeamMap_GFitWidth    = 213,
u_BeamScope_GFitWidth       = 214,
v_BeamScope_GFitWidth       = 215,
u_WinCamD_GFitWidth         = 216,
v_WinCamD_GFitWidth         = 217,
MajorWidth_ISO11146         = 218,
MinorWidth_ISO11146         = 219,
ELPDEG_ISO11146             = 220,
x_WinCam_Divergence_at_Clip_1 = 221,
x_WinCam_Divergence_at_Clip_2 = 222,
y_WinCam_Divergence_at_Clip_1 = 223,
y_WinCam_Divergence_at_Clip_2 = 224,
Xu_WinCamD                   = 225,
Yu_WinCamD                   = 226,
Spare49                       = 227,
Spare50                       = 228,

Last_PARAMETER_ID            = 229,

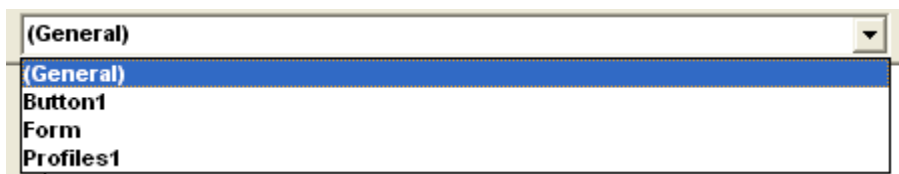
SetClipLevel1                = 230,
SetClipLevel2                = 231,
PeakButton                   = 232,
StatusButton                 = 233,
ZeroButton                   = 234,
SpacerButton1                = 235,
SpacerButton2                = 236,
ZoomButton                   = 237,
CrossHairButton              = 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,
MessageButton                 = 248,

Best_u_Divergence            = 249,
Best_v_Divergence            = 250,
Reset_Divergence             = 251,

Last_ID                       = 252,
};

```

**Visual Basic.** Select the gadget **B** on the tool bar, then place and size the Button Object in the form. The first Button Object becomes Button1 in the project.



**Button Definitions:**

Add Class Module:

**Project->Add Module -> DataRayOcx (DataRayOcx.sls)** If needed.

Open the DatarayOcx.cls mode and find **DataRay\_Profiles:**

*Note: As indicated by DataRay\_Parameters\_Buttons, in most cases, each button displays a particular parameter and is also used to get that result. But which one to use? In the main DataRay.exe program, LEFT CLICK the button you want, the name of that button is displayed!*

Enum DataRay\_Parameters\_Buttons

- BlankParameters = 0
- DefaultParameters = 1
- u\_BeamR\_Width\_at\_Clip\_1 = 2
- u\_BeamR\_Width\_at\_Clip\_2 = 3
- u\_BeamR\_GFit = 4
- u\_BeamR\_TopHat = 5
- v\_BeamR\_Width\_at\_Clip\_1 = 6
- v\_BeamR\_Width\_at\_Clip\_2 = 7
- v\_BeamR\_GFit = 8
- v\_BeamR\_TopHat = 9
- v1\_BeamC\_Width\_at\_Clip\_1 = 10
- v1\_BeamC\_Width\_at\_Clip\_2 = 11
- v1\_BeamC\_GFit = 12
- v1\_BeamC\_TopHat = 13
- u1\_BeamC\_Width\_at\_Clip\_1 = 14
- u1\_BeamC\_Width\_at\_Clip\_2 = 15
- u1\_BeamC\_GFit = 16
- u1\_BeamC\_TopHat = 17
- v2\_BeamC\_Width\_at\_Clip\_1 = 18
- v2\_BeamC\_Width\_at\_Clip\_2 = 19
- v2\_BeamC\_GFit = 20
- v2\_BeamC\_TopHat = 21
- u2\_BeamC\_Width\_at\_Clip\_1 = 22
- u2\_BeamC\_Width\_at\_Clip\_2 = 23
- u2\_BeamC\_GFit = 24
- u2\_BeamC\_TopHat = 25
- v3\_BeamC\_Width\_at\_Clip\_1 = 26
- v3\_BeamC\_Width\_at\_Clip\_2 = 27
- v3\_BeamC\_GFit = 28
- v3\_BeamC\_TopHat = 29
- u3\_BeamC\_Width\_at\_Clip\_1 = 30
- u3\_BeamC\_Width\_at\_Clip\_2 = 31
- u3\_BeamC\_GFit = 32
- u3\_BeamC\_TopHat = 33
- v4\_BeamC\_Width\_at\_Clip\_1 = 34
- v4\_BeamC\_Width\_at\_Clip\_2 = 35
- v4\_BeamC\_GFit = 36

v4_BeamC_TopHat	= 37
u4_BeamC_Width_at_Clip_1	= 38
u4_BeamC_Width_at_Clip_2	= 39
u4_BeamC_GFit	= 40
u4_BeamC_TopHat	= 41
minus2_BeamMap_Width_at_Clip_1	= 42
minus2_BeamMap_Width_at_Clip_2	= 43
minus2_BeamMap_GFit	= 44
minus2_BeamMap_TopHat	= 45
minus1_BeamMap_Width_at_Clip_1	= 46
minus1_BeamMap_Width_at_Clip_2	= 47
minus1_BeamMap_GFit	= 48
minus1_BeamMap_TopHat	= 49
zero_BeamMap_Width_at_Clip_1	= 50
zero_BeamMap_Width_at_Clip_2	= 51
zero_BeamMap_GFit	= 52
zero_BeamMap_TopHat	= 53
plus1_BeamMap_Width_at_Clip_1	= 54
plus1_BeamMap_Width_at_Clip_2	= 55
plus1_BeamMap_GFit	= 56
plus1_BeamMap_TopHat	= 57
plus2_BeamMap_Width_at_Clip_1	= 58
plus2_BeamMap_Width_at_Clip_2	= 59
plus2_BeamMap_GFit	= 60
plus2_BeamMap_TopHat	= 61
plus45_BeamMap_Width_at_Clip_1	= 62
plus45_BeamMap_Width_at_Clip_2	= 63
plus45_BeamMap_GFit	= 64
plus45_BeamMap_TopHat	= 65
minus45_BeamMap_Width_at_Clip_1	= 66
minus45_BeamMap_Width_at_Clip_2	= 67
minus45_BeamMap_GFit	= 68
minus45_BeamMap_TopHat	= 69
u_BeamScope_Width_at_Clip_1	= 70
u_BeamScope_Width_at_Clip_2	= 71
u_BeamScope_GFit	= 72
u_BeamScope_TopHat	= 73
v_BeamScope_Width_at_Clip_1	= 74
v_BeamScope_Width_at_Clip_2	= 75
v_BeamScope_GFit	= 76
v_BeamScope_TopHat	= 77
u_WinCamD_Width_at_Clip_1	= 78
u_WinCamD_Width_at_Clip_2	= 79
u_WinCamD_GFit	= 80
u_WinCamD_TopHat	= 81
v_WinCamD_Width_at_Clip_1	= 82
v_WinCamD_Width_at_Clip_2	= 83
v_WinCamD_GFit	= 84
v_WinCamD_TopHat	= 85
Xc_BeamR	= 86
Yc_BeamR	= 87
uM2_NA_BeamC_alt	= 88
vM2_NA_BeamC_alt	= 89
Uniformity_WinCamD	= 90
Ewidth_WinCamD	= 91
Ellipticity_BeamR	= 92
Power_BeamR	= 93
Xc1_BeamC	= 94
Yc1_BeamC	= 95

Xg1_BeamC_not_used	= 96
Yg1_BeamC_not_used	= 97
Xp1_BeamC_not_used	= 98
Yp1_BeamC_not_used	= 99
Xc2_BeamC	= 100
Yc2_BeamC	= 101
uMsquared_BeamC_Alt	= 102
uM2_Zo_BeamC_Alt	= 103
Xp2_BeamC_not_used	= 104
Yp2_BeamC_not_used	= 105
Xc3_BeamC	= 106
Yc3_BeamC	= 107
vMsquared_BeamC_Alt	= 108
vM2_Zo_BeamC_Alt	= 109
Xp3_BeamC_not_used	= 110
Yp3_BeamC_not_used	= 111
Xc4_BeamC	= 112
Yc4_BeamC	= 113
Xg4_BeamC_not_used	= 114
Yg4_BeamC_not_used	= 115
Xp4_BeamC_not_used	= 116
Yp4_BeamC_not_used	= 117
Ellipticity_BeamC	= 118
Power_BeamC	= 119
Xc_BeamMap	= 120
Yc_BeamMap	= 121
Xg_BeamMap_not_used	= 122
Yg_BeamMap_not_used	= 123
Xp_BeamMap_not_used	= 124
Yp_BeamMap_not_used	= 125
Ellipticity_BeamMap	= 126
Power_BeamMap	= 127
DivergenceNA_BeamMap	= 128
DivergenceDegrees_BeamMap	= 129
DivergenceRadians_BeamMap	= 130
Msquared_BeamMap	= 131
M2_2Wo_BeamMap	= 132
M2_Zo_BeamMap	= 133
M2_Zr_BeamMap	= 134
M2_Theta_BeamMap	= 135
M2_NA_BeamMap	= 136
Xc_BeamScope	= 137
Yc_BeamScope	= 138
Xg_BeamScope	= 139
Yg_BeamScope	= 140
Xp_BeamScope	= 141
Yp_BeamScope	= 142
Ellipticity_BeamScope	= 143
Power_BeamScope	= 144
uDivergenceNA_BeamC	= 145
uDivergenceDegrees_BeamC	= 146
uDivergenceRadians_BeamC	= 147
uMsquared_BeamC	= 148
uM2_2Wo_BeamC	= 149
uM2_Zo_BeamC	= 150
uM2_Zr_BeamC	= 151
uM2_Theta_BeamC	= 152
uM2_NA_BeamC	= 153
vDivergenceNA_BeamC	= 154


vDivergenceDegrees\_BeamC = 155  
vDivergenceRadians\_BeamC = 156  
vMsquared\_BeamC = 157  
vM2\_2Wo\_BeamC = 158  
vM2\_Zo\_BeamC = 159  
vM2\_Zr\_BeamC = 160  
vM2\_Theta\_BeamC = 161  
vM2\_NA\_BeamC = 162  
Xc\_WinCamD = 163  
Yc\_WinCamD = 164  
Xg\_WinCamD = 165  
Yg\_WinCamD = 166  
Xp\_WinCamD = 167  
Yp\_WinCamD = 168  
Ellipticity\_WinCamD = 169  
Power\_WinCamD = 170  
Oreintation\_WinCamD = 171  
MajorWidth\_WinCamD = 172  
MinorWidth\_WinCamD = 173  
MeanWidth\_WinCamD = 174  
Peak\_WinCamD = 175  
AverageFluence\_WinCamD = 176  
uM2\_M2\_BeamScope = 177  
uM2\_2Wo\_BeamScope = 178  
uM2\_Zo\_BeamScope = 179  
uM2\_Zr\_BeamScope = 180  
uM2\_Theta\_BeamScope = 181  
uM2\_NA\_BeamScope = 182  
vM2\_M2\_BeamScope = 183  
vM2\_2Wo\_BeamScope = 184  
vM2\_Zo\_BeamScope = 185  
vM2\_Zr\_BeamScope = 186  
vM2\_Theta\_BeamScope = 187  
vM2\_NA\_BeamScope = 188  
ID\_WANDER = 189  
PointingX\_BeamMapC = 190  
PointingY\_BeamMapC = 191  
PointingX\_BeamMap = 192  
Msquared\_BeamMap\_Alt = 193  
M2\_Zo\_BeamMap\_Alt = 194  
M2\_Theta\_BeamMap\_Alt = 195  
M2\_Zo\_BeamMap\_Alt2 = 196  
u\_BeamR\_GFitWidth = 197  
v\_BeamR\_GFitWidth = 198  
v1\_BeamC\_GFitWidth = 199  
u1\_BeamC\_GFitWidth = 200  
v2\_BeamC\_GFitWidth = 201  
u2\_BeamC\_GFitWidth = 202  
v3\_BeamC\_GFitWidth = 203  
u3\_BeamC\_GFitWidth = 204  
v4\_BeamC\_GFitWidth = 205  
u4\_BeamC\_GFitWidth = 206  
minus2\_BeamMap\_GFitWidth = 207  
minus1\_BeamMap\_GFitWidth = 208  
zero\_BeamMap\_GFitWidth = 209  
plus1\_BeamMap\_GFitWidth = 210  
plus2\_BeamMap\_GFitWidth = 211  
plus45\_BeamMap\_GFitWidth = 212  
minus45\_BeamMap\_GFitWidth = 213  
u\_BeamScope\_GFitWidth = 214

v_BeamScope_GFitWidth	= 215
u_WinCamD_GFitWidth	= 216
v_WinCamD_GFitWidth	= 217
MajorWidth_ISO11146	= 218
MinorWidth_ISO11146	= 219
ELPDEG_ISO11146	= 220
x_WinCam_Divergence_at_Clip_1	= 221
x_WinCam_Divergence_at_Clip_2	= 222
y_WinCam_Divergence_at_Clip_1	= 223
y_WinCam_Divergence_at_Clip_2	= 224
Xu_WinCamD	= 225
Yu_WinCamD	= 226
Spare49	= 227
Spare50	= 228
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
MessageButton	= 248
Best_u_Divergence	= 249
Best_v_Divergence	= 250
Reset_Divergence	= 251
Last_ID	= 252

End Enum

3) **GetData**: This object is only non visual object, it is the interface object, and as its name implies, its main function is to get the data parameters and control the device.

Visual C++:

With Visual Basic, select the  gadget on the tool bar, then size and place it into the form, it will be hidden in run mode, so remember where you put. The first (and only!) GetData Object becomes GetData1 in the project.

File **MainFrm.cpp** contains the code that installs and links to the DataRay OCX. See InstallOcx() for the simple code to create an instance of the OCX.

In the C++ example the GetData object is linked as "Ocx", and referenced as Ocx->function(parameter,...).

Example Initialization:

```
Ocx->StartDriver() ;  
Ocx->SetCurrentDevice(IS_WINCAM) ;  
Ocx->StartDevice() ;
```

## A to Z Descriptions of Functions (Methods) & Parameters (Properties)

**Color Code**  
**Brown:** Methods or Properties  
**Red:** How defined when installed in Visual C++  
**Green:** OCX example  
**Green:** Visual Basic example  
**Blue:** Comment(s)

Property: AtAim  
**BOOL CGetDataOcx::GetAtAim()**  
**void CGetDataOcx::SetAtAim(BOOL propVal)**  
ok = Ocx->GetAtAim()  
Ocx->SetAtAim(TRUE)  
ok = GetData1.AtAim  
Setdata1.AtAim = TRUE  
BeamScope only, if true, the finger is at the aim position.

Method: AutoCrosshairs  
**void CGetDataCtrl::AutoCrosshairs ()**  
Ocx->CGetDataCtrl::AutoCrosshairs () ;  
GetData1.CGetDataCtrl::AutoCrosshairs () ;  
WinCam only, turns on auto crosshairs (orients to the beam).  
See also, ForceCrosshairsToZero() and ForceCrosshairsTo45();

Property: AutoNaming  
**short CGetDataOcx::GetAutoNaming()**  
**void CGetDataOcx::SetAutoNaming(short propVal)**  
on = Ocx->GetAutoNaming()  
Ocx->SetAutoNaming(TRUE)  
on = GetData1.AutoNaming  
GetData1.AutoNaming = True  
If true, Auto naming is enabled.  
Example auto name = WC\_05\_4\_25\_10\_29\_22.wcf  
WinCam\_year\_month\_day\_hour\_minute\_second.extention

Property: AutoShutterOn  
**BOOL CGetDataOcx::GetAutoShutterOn()**  
**void CGetDataOcx::SetAutoShutterOn(BOOL propVal)**  
on = Ocx->GetAutoShutterOn ()  
Ocx->SetAutoShutterOn (TRUE) ;  
GetData1.AutoShutterOn = True  
WinCam only, controls auto exposure (auto shutter) .

Property: AutoSnap  
short CGetDataOcx::GetAutoSnap()  
void CGetDataOcx::SetAutoSnap(short propVal)  
SnapMode = Ocx->GetAutoSnap ()  
Ocx->Set AutoSnap(SNAP\_CENTROID)  
SnapMode = Getdata1.AutoShutterOn  
Getdata1.AutoShutterOn = SNAP\_CENTROID  
WinCam only, controls cursor method.  
#define SNAP\_CENTROID 0 // cursor follows centroid  
#define SNAP\_CENTER 1 // cursor follows geometric center  
#define SNAP\_PEAK 2 // cursor follows peak  
#define SNAP\_USER 3 // use placed

Property: BackGroundSubtraction  
Obsolete, do not use

Property: BaselineLocked  
short CGetDataCtrl::GetBaselineLocked()  
void CGetDataCtrl::SetBaselineLocked(short nNewValue)  
locked = Ocx->GetBaselineLocked () ;  
Ocx-> SetBaselineLocked(TRUE)  
locked = Getdata1.BaselineLocked  
Getdata1.BaselineLocked = True  
Locks the current baseline (zero level) turning off any futher adjustments.

Property: BeamMapCdefaultXc  
long CGetDataCtrl::GetBeamMapCdefaultXc()  
void CGetDataCtrl::SetBeamMapCdefaultXc(long nNewValue)  
plane = Ocx->GetBeamMapCdefaultXc() ;  
Ocx->SetBeamMapCdefaultXc(newPlane) ;  
plane = Getdata1.GetBeamMapCdefaultXc  
Getdata1.SetBeamMapCdefaultXc = newPlane  
BeamMap-Collimate only, sets the plane (one of four) that is used for Beam Wander and displayed Xc/Yc.

Property: BeamScopeScanControl  
Obsolete, do not use

Property: CameraSelect  
Obsolete, do not use

Property: CameraSequence  
long CGetDataCtrl::GetCameraSequence()  
void CGetDataCtrl::SetCameraSequence(long nNewValue)  
list = Ocx->CameraSequence () ;  
Ocx->SetCameraSequence (list)  
list = Getdata1.CameraSequence  
Getdata1.SetCameraSequence = list  
WinCam only, If multiple cameras are used this property is used to enable multiple cameras the will be used in sequence.

Example:

Cameras 1, 3 and 9 to be enabled:  
Int list = 1 + 8 + 256  
Where camera 9 = 256 or  $2^{(camera-1)}$

Method: CaptureIsFullResolution

```
BOOL CGetDataCtrl::CaptureIsFullResolution()
```

```
Full = Ocx->CaptureIsFullResolution() ;
```

```
Full = Getdata1.CaptureIsFullResolution()
```

WinCam only. The WinCam has two capture modes, FULL and FAST. Where FULL is every pixel and every line and FAST where every other pixel and every other line is transferred. But in FAST mode the adjacent pixels and lines are added together first so that no pixel is unused.

Property: CentroidClipLevel

```
double CGetDataCtrl::GetCentroidClipLevel()
```

```
void CGetDataCtrl::SetCentroidClipLevel(double newValue)
```

```
clip = Ocx->GetCentroidClipLevel() ;
```

```
Ocx->SetCentroidClipLevel( .135 ) ; // set to 13.5%
```

```
clip = Getdata1.GetCentroidClipLevel
```

```
Getdata1.SetCentroidClipLevel = .135 rem set to 13.5%
```

The centroid clip level is used in the centroid calculations to help suppress the influence of noise, it is not used directly.

Method: CloseDialog

```
short CGetDataCtrl::CloseDialog(short IndexToDialog)
```

```
success = Ocx-> CloseDialog(CENTROID_CLIP_DLG) ;
```

```
success = Getdata1.CloseDialog(CENTROID_CLIP_DLG)
```

Closes a previously opened dialog.

```
enum OpenCloseDialogs {  
    M2_DEBUG_DLG=1,  
    M2_BEAMSCOPE_DLG=2,  
    M2_BEAMMAP_DLG=3,  
    M2_BEAMC_DLG=4,  
    DIV_BEAMMAP_DLG=5,  
    DIV_BEAMC_DLG=6,  
    WC_FLUENCE_DLG=7,  
    NUMERIC_DISPLAY_MODES=8,  
    EPPROM_DATA=9,  
    LOGGING_DLG=10,  
    BS_PULSED_DLG=11,  
    WAVE_LENGTH_DLG=12,  
    CAPTURE_DLG=13,  
    PCI_EEPROM_DLG=14,  
    WANDER_DLG=15,  
    CENTROID_CLIP_DLG=16,  
    WC_IMAGE_LOG_SETUP_DLG=17,  
    WC_IMAGE_LOG_DLG=18,  
    WC_LOGGING_DLG=19,  
    BEAM_FIT_DLG=20,  
    TRIGGER_SETUP=21,  
    M_FACTOR_DLG=22,  
    GET_EWIDTH_CLIP=23,  
    UCM_TEST_DLG=24,  
};
```

Property: CurrentCamera

```
short CGetDataCtrl::GetCurrentCamera()
```

```
void CGetDataCtrl::SetCurrentCamera(short nNewValue) // not used
```

```
Ocx->SetCurrentCamera(7) ; // set camera 8 as current (if there)
```

```
camera = Ocx->GetCurrentCamera() ;
```

```
Getdata1.SetCurrentCamera= 7
```

```
camera = Getdata1.GetCurrentCamera
```

WinCam only, selects the camera to use.

WARNING: The number is zero based, so Camera1 = 0;

Property: CustomSetup  
Not used.

Method: DeviceRunning  
**BOOL CGetDataOcx::DeviceRunning()**  
Running = Ocx->DeviceRunning() ;  
Running = Getdata1.DeviceRunning()  
Returns true if device is active.

Property: DisableEffectiveCentroidsX  
Not used.

Property: DisableEffectiveCentroidsY  
Not used.

Method: DoSearch  
**BOOL CGetDataOcx::DoSearch()**  
Success = Ocx->DoSearch() ;  
Success = Getdata1.DoSearch()  
Initiates BeamScope search or re-finds the beam for the BeamMap series.

Property: EffectiveCentroidFilterInPixels  
Do not use.

Property: EnableEffectiveCentroidsX  
Do not use.

Property: EnableEffectiveCentroidsY  
Do not use.

Method: EnableUseEffectiveSlits  
**long CGetDataOcx::EnableUseEffectiveSlits(long Enable)**  
enabled = Ocx->EnableUseEffectiveSlits(TRUE) ;  
enabled = Getdata1.EnableUseEffectiveSlits(TRUE)  
WinCam only, selects the effective slits (the sum of the rows and columns) as the profile source.

Property: eTrapOn  
Do not use.

Method: ExportAsBitMap  
**BOOL CGetDataOcx::ExportAsBitMap(long ThisAsLong)**  
Ocx->ExportAsBitMap(long(this)) ;  
An utility that copies the current object (this) to a bitmap file.  
Example: See in the DataRay program menu item File->Save current screen as bitmap.

Method: ExportToPaint  
**void CGetDataOcx::ExportToPaint(long ThiPointer)**  
Ocx-> ExportToPaint (long(this)) ;  
An utility that copies the current object (this) to the Paint program.  
Example: See in the DataRay program menu item File->Export current screen to Paint.

Property: Exposure  
**double CGetDataOcx::GetExposure(long WhichCamera)**  
**void CGetDataOcx::SetExposure(long WhichCamera, double newValue)**  
exposure = Ocx->GetExposure(0) ; // get camera 1 exposure setting in ms  
Getdata1.SetExposure = 0.50 rem set camera 1 exposure setting to 500 us  
exposure = Getdata1.GetExposure(0) ; // get camera 1 exposure setting in ms  
WinCam only, sets the cameras shutter period.

Property: FastUpdate  
short CGetDataOcx::GetFastUpdate()  
void CGetDataOcx::SetFastUpdate(short propVal)  
fast = Ocx->GetFastUpdate() ;  
Ocx->SetFastUpdate = False rem default  
fast = Getdata1. GetFastUpdate ;  
WinCam only, turns off all calculations to decrease capture time.

Property: FilterValue  
double CGetDataOcx::GetFilterValue()  
void CGetDataOcx::SetFilterValue(double propVal)  
value = Ocx->GetFilterValue() ;  
Ocx->SetFilterValue(1.0) ; Set value to 1.0%  
value = Getdata1. GetFilterValue;  
Getdata1.SetFilterValue = 1.0 rem Set value to 1.0%  
Use to filter all the profiles, Range = 0 to 10%, default = 0.2%

Method: FindWinCamImage  
Obsolete, do not use.

Method: FingerToPosition  
BOOL CGetDataOcx::FingerToPosition(double Position)  
Success = Ocx->FingerToPosition(12000) ; move finger to 12000 um out  
Success = Getdata1.FingerToPosition(12000) rem move finger to 12000 um out  
BeamScope only, used in program to move finger to "AIM" and "SLITS" position.

Method: ForceCrosshairsTo45  
void CGetDataOcx::ForceCrosshairsTo45()  
Ocx->ForceCrosshairsTo45() ;  
Getdata1.ForceCrosshairsTo45()  
WinCam only, forces the crosshairs to 45 degrees.  
See also, ForceCrosshairsToZero() and AutoCrosshairs() ;

Method: ForceCrosshairsToZero  
void CGetDataOcx::ForceCrosshairsToZero()  
Ocx-> ForceCrosshairsTo45() ;  
Getdata1.ForceCrosshairsToZero ()  
WinCam only, forces the crosshairs to 45 degrees.  
See also, ForceCrosshairsTo45 and AutoCrosshairs() ;

Property: GeoClipLevel  
double CGetDataOcx::GetGeoClipLevel()  
void CGetDataOcx::SetGeoClipLevel(double propVal)  
clip = Ocx-> GeoClipLevel () ;  
Ocx->GeoClipLevel = .135 rem set to 13.5%  
clip = Getdata1.GeoClipLevel  
Getdata1.GeoClipLevel ( .135 ) ; // set to 13.5%  
The geometric clip level is used in the geometric center calculations to help suppress the influence of noise, it is not used directly.

Method: GetAverageNumber  
short CGetDataOcx::GetAverageNumber()  
averages = Ocx->GetAverageNumber() ;  
averages = Getdata1.GetAverageNumber();  
Returns the current averaging setting.

Method: GetBeamScopeIndex  
Obsolete

Method: GetBeamScopeState  
Obsolete

Method: GetCentroidXlocation  
**long CGetDataOcx::GetCentroidXlocation()**  
x = Ocx->GetCentroidXlocation() ;  
x = Getdata1.GetCentroidXlocation()  
WinCam only, Returns the current centroid x position in PIXELS.

Method: GetCentroidYlocation  
**long CGetDataOcx::GetCentroidYlocation()**  
y = Ocx->GetCentroidXlocation() ;  
y = Getdata1.GetCentroidXlocation()  
WinCam only, Returns the current centroid y position in PIXELS.

Method: GetClipLevel  
**double CGetDataOcx::GetClipLevel(short ClipOneOrTwo\_0\_1)**  
clip = Ocx->GetClipLevel(0) ; // get clip 1 (a)  
clip = Getdata1.GetClipLevel(1) rem get clip 2 (b)  
Returns the current clip levels.

Method: GetClipLevelMode  
**short CGetDataOcx::GetClipLevelMode(short ClipOneOrTwo\_0\_1)**  
mode = Ocx->GetClipLevelMode(0) ; // // get clip mode 1 (a)  
mode = Getdata1.GetClipLevelMode(1) rem get clip mode 2 (b)  
Returns the current clip level mode.  
enum WidthModes {  
    CLIP\_LEVEL\_METHOD=0,  
    SIGMA4\_METHOD=1,  
    KNIFE\_EDGE\_METHOD=2  
};

Method: GetCurrentDevice  
**short CGetDataOcx::GetCurrentDevice()**  
device = Ocx->GetCurrentDevice() ;  
device = Getdata1.GetCurrentDevice()  
enum \_Selected\_Device\_ {  
    IS\_BEAMSCOPE=1,  
    IS\_BEAMR=2,  
    IS\_BEAMMAP=3,  
    IS\_BEAMC=4,  
    IS\_WINCAM=5,  
    IS\_WINCAM\_DIV=6,  
    IS\_WINCAM\_LOG=7,  
    IS\_TWOD\_SCAN=8,  
    IS\_SPARE4=9,  
    LAST\_DEVICE=10,  
};

Method: GetCurrentIndex  
**short CGetDataOcx::GetCurrentIndex()**  
index = Ocx->GetCurrentIndex() ;  
index = Getdata1.GetCurrentIndex()  
Returns the current index of the data buffer, either live data or recalled data depending on the live/recall state. See GetCurrentState.

Method: GetCurrentState  
**short CGetDataOcx::GetCurrentState()**  
state = Ocx->GetCurrentState();  
state = Getdata1.GetCurrentState()  
#define \_LIVE\_ 0  
#define \_RECALL\_ 1

Method: GetCurrentWinCamData  
**BOOL CGetDataOcx::GetCurrentWinCamData(long\* ImageDataPt, long\* XSizePt, long\* YSizePtr)**  
**YSizePtr)**  
Success = Ocx->GetCurrentWinCamData(ImageDataPtr, XSizePtr, YSizePtr) ;

Example:

```

    long XsizePtr;
    long YsizePtr;
    long ImageDataPtr ;
    Success = Ocx->GetCurrentWinCamData(&ImageDataPtr, &XSizePtr, &YSizePtr)
int    w = XsizePtr ;
    int    h = YsizePtr ;
    long    *Data = (long *) ImageDataPtr ;
    double sum = 0 ;
    for(int y=0; y<h; y++) {
        for(int x=0; x<w; x++) Sum += *Data++ ;
    }
    // do not delete ImageDataPtr

```

Method: GetEffectiveCentroidX  
Obsolete, do not use.

Method: GetEffectiveCentroidY  
Obsolete, do not use.

Method: GetEffectiveGeoCenterX  
Obsolete, do not use.

Method: GetEffectiveGeoCenterY  
Obsolete, do not use.

Method: GetEffectiveWidthCliplevel  
Obsolete, do not use.

Method: GetErrorStatus  
Obsolete, do not use.

Method: GetFirmwareRevInfo  
Obsolete, do not use. (factory use)

Method: GetHeadAngle  
**double CGetDataOcx::GetHeadAngle()**  
angle = Ocx->GetHeadAngle() ;  
angle = Getdata1.GetHeadAngle()  
Returns the angle that the BeamMap/R/C.

Method: GetHelpString  
**CString CGetDataOcx::GetHelpString()**  
**CString string = Ocx->GetHelpString() ;**  
Return the help text associated with the cursor passing over a DataRay object.

Method: GetHorizontalPixels  
**short CGetDataOcx::GetHorizontalPixels()**  
width = Ocx->GetHorizontalPixels() ;  
width = Getdata1.GetHorizontalPixels()  
WinCam only, returns the horizontal image size in pixels.

Method: GetIncludedPowerPercentAtRadius  
**double CGetDataOcx::GetIncludedPowerPercentAtRadius(double RadiusInMicrons)**  
percent = Ocx->GetIncludedPowerPercentAtRadius(200) ; // 200 microns  
percent = Getdata1.GetIncludedPowerPercentAtRadius(200) rem 200 microns  
WinCam only, returns the percent of power within the given radius.  
See also: GetIncludedPowerTotal

Method: GetIncludedPowerTotal  
**double CGetDataOcx::GetIncludedPowerTotal()**  
power = Ocx->GetIncludedPowerTotal() ;  
power = Getdata1.GetIncludedPowerTotal()  
WinCam only, returns the total of all the pixels minus the baseline.  
See also: GetIncludedPowerPercentAtRadius

Method: GetLastError  
Factory use only.

Method: GetNonuniformityOnOff  
Factory use only.

\*\* Method: GetOcxResult  
**double CGetDataOcx::GetOcxResult(short IndexToValue)**  
result = Ocx->GetOcxResult(u\_BeamR\_Width\_at\_Clip\_1) ;  
result = Getdata1.GetOcxResult(u\_BeamR\_Width\_at\_Clip\_1)  
See Button Definitions: above for defintions of IndexToValue.

\*\* Method: GetOcxResultExt  
**double CGetDataOcx::GetOcxResultExt(long WhichResult, long WhichCamera)**  
result = Ocx-> GetOcxResultExt (u\_BeamR\_Width\_at\_Clip\_1, 0) ; // Camera 1  
result = Getdata1. GetOcxResultExt (u\_BeamR\_Width\_at\_Clip\_1, 1) rem Camera 2  
WinCam only.  
See Button Definitions: above for defintions of WhichResult.

Method: GetOcxResultName  
**CString CGetDataOcx::GetOcxResultName(short IndexToValue)**  
CString Name = Ocx->GetOcxResultName(u\_BeamR\_Width\_at\_Clip\_1) ;  
Returns the name used on the associated button object.  
See Button Definitions: above for defintions of IndexToValue.

Method: GetOcxRev  
**CString CGetDataOcx::GetOcxRev()**  
CString Rev = Ocx->GetOcxRev() ;  
Returns the attached ocx release number.

Method: GetParameter  
Obsolete use GetOcxResult or GetOcxResultExt.

Method: GetPeakXlocation  
**long CGetDataOcx::GetPeakXlocation()**  
x = Ocx-> GetPeakXlocation() ;  
x = Getdata1.GetPeakXlocation()  
WinCam only, returns the horizontal location of the current peak location in PIXELS.

Method: GetPeakYlocation  
**long CGetDataOcx::GetPeakYlocation()**  
x = Ocx->GetPeakYlocation() ;  
x = Getdata1.GetPeakYlocation()  
WinCam only, returns the vertical location of the current peak location in PIXELS.

Method: GetPixel  
**long CGetDataOcx::GetPixel(long x, long y)**  
intensity = Ocx->GetPixel(w, h) ;  
intensity = Getdata1.GetPixel(w, h)  
WinCam only, return the value of the pixel addressed.

Method: GetProfileTop  
Factory use only.

Method: GetRadiusAtPowerPercent  
**double CGetDataOcx::GetRadiusAtPowerPercent(double PowerPercent)**  
radius = Ocx->GetRadiusAtPowerPercent(86.5) ; // 86.5 % included  
radius = Getdata1.GetRadiusAtPowerPercent(86.5) rem 86.5 % included  
WinCam only, returns the effective radius that contains the specified power.  
The example above is equivalent to (Effective Diameter)/2 ;

Method: GetRealTimeLogging  
Factory use only.

Method: GetRecallFileVersion /sp  
Factory use only.

Method: GetROI  
**BOOL CGetDataOcx::GetROI(long\* LeftAsLongPointer, long\* TopAsLongPointer, long\* WidthAsLongPointer, long\* HeightAsLongPointer)**  
Success = Ocx-> CGetDataCtrl::GetROI(&LeftAsLongPointer, &TopAsLongPointer, &WidthAsLongPointer, &HeightAsLongPointer) ;  
WinCam only, returns the Region Of Interest (capture area).

Method: GetSampleCount  
**short CGetDataOcx::GetSampleCount(short Live\_Is\_0)**  
count = Ocx-> GetSampleCount(\_LIVE\_) ;  
count = Getdata1.GetSampleCount(\_LIVE\_)  
returns the number of samples in the buffer.

Method: GetSavedDataPointer  
Factory use only.

Method: GetSaveFileName  
Factory use only.

Method: GetShutterSetting  
Obsolete, use Property: Exposure

Method: GetSoftwareVersion  
Factory use only.

Method: GetVerticalPixels  
**short CGetDataOcx::GetVerticalPixels()**  
height = Ocx->GetVerticalPixels() ;  
height = Getdata1.GetVerticalPixels()  
WinCam only, returns the Region Of Interest height.

Method: GetWinCamDPixelSize  
`double CGetDataOcx::GetWinCamDPixelSize(short X_0_Y_1)`  
`Xpixel = Ocx->GetWinCamDPixelSize(0) ;`  
`Ypixel = Getdata1.GetWinCamDPixelSize(1)`  
WinCam only, returns the current pixel size.

Method: GetWinCamSingle  
`void CGetDataOcx::GetWinCamSingle()`  
`Ocx->GetWinCamSingle() ;`  
`Getdata1.GetWinCamSingle()`  
WinCam only, initiates a single capture, BUT it is NOT guaranteed to complete before the next call. See also GetWinCamSingleAndComplete.

Method: GetWinCamSingleAndComplete  
`void CGetDataOcx:: GetWinCamSingleAndComplete`  
`Ocx-> GetWinCamSingleAndComplete() ;`  
`Getdata1.GetWinCamSingleAndComplete()`  
WinCam only, initiates a single capture, AND IS guaranteed to complete before the next call.

Method: GetWinCamTraps  
Factory use only.

Method: HomeP7Head  
`BOOL CGetDataOcx::HomeP7Head()`  
`Success = Ocx->HomeP7Head() ;`  
`Success = Getdata1.HomeP7Head()`  
BeamScope only, this functions sends the finger to its reference point.

Method: HowManyBoardsOpened  
Factory use only.

Property: Index  
Obsolete do not use.

Property: InkSaverState  
`short CGetDataOcx::GetInkSaverState()`  
`On = Ocx->GetInkSaverState() ;`  
`On = Getdata1.GetInkSaverState()`  
Returns the state of the ink saver option.

Method: IsBeamScopeP7HeadThere  
`BOOL CGetDataOcx::IsBeamScopeP7HeadThere()`  
`There = Ocx-> IsBeamScopeP7HeadThere() ;`  
`There = Getdata1.IsBeamScopeP7HeadThere();`  
Returns true if P7 head detected.

Method: IsCameraThere  
`BOOL CGetDataOcx::IsCameraThere(short WhichCamera_0_1)`  
`Camera1There = Ocx-> IsCameraThere(0) ;`  
`Camera2There = Getdata1.IsCameraThere(1)`  
Returns true if camera N detected.

Method: IsDataReady  
Obsolete do not use.

Property: IsDivergenceOpen  
Factory use only.

Property: IsHeadRotationFixed  
Factory use only.

Property: IsHeadStalled  
**BOOL CGetDataOcx::IsHeadStalled()**  
Stalled = Ocx->IsHeadStalled() ;  
Stalled = Getdata1.IsHeadStalled()  
BeamMap, Beam'R, BeamMap-C only.

Property: IsMSquaredOpen  
Factory use only.

Property: JitterSuppression  
Obsolete do not use.

Method: KeyEvent  
Factory use only.

Method: LoadDefaults  
**void CGetDataOcx::LoadDefaults()**  
Ocx-> LoadDefaults() ;  
Getdata1.LoadDefaults()  
Loads most default settings place the ocx into a known state.

Method: LoadJobFile  
**long CGetDataOcx::LoadJobFile()**  
Success = Ocx-> LoadJobFile() ;  
Success = Getdata1.LoadJobFile();  
A dialog box will open to select the job file and location.

Method: LoadThisJobFile  
**BOOL CGetDataOcx::LoadThisJobFile(LPCTSTR JobFileNamePath)**  
Success = Ocx->LoadThisJobFile("C\\Path\\MyJobFile.OJF") ;  
Success = Getdata1.LoadThisJobFile("C\\Path\\MyJobFile.OJF")  
No dialog box.

Property: LockAll  
Obsolete do not use.

Property: Name  
Obsolete do not use.

Method: NextProfile  
**BOOL CGetDataOcx::NextProfile()**  
Ocx->NextProfile() ;  
Getdata1.NextProfile() ;  
Increments to the next data buffer index.

Property: Object  
Factory use only.

Method: OpenBeamMap  
Obsolete do not use.

Method: OpenBeamScopeP7  
Obsolete do not use.

Method: OpenClipLevelDlg  
**BOOL CGetDataOcx::OpenClipLevelDlg(short ClipOneOrTwo\_0\_1)**  
Success = Ocx->OpenClipLevelDlg(0) ; // clip a  
Success = Getdata1.OpenClipLevelDlg(1) rem clip b

Method: OpenFileDialog

```
short CGetDataOcx::OpenDialog(short IndexToDialog)  
success = Ocx->OpenDialog (CENTROID_CLIP_DLG) ;  
success = Getdata1.OpenDialog g(CENTROID_CLIP_DLG)  
Open a dialog.
```

```
enum OpenCloseDialogs {  
    M2_DEBUG_DLG=1,  
    M2_BEAMSCOPE_DLG=2,  
    M2_BEAMMAP_DLG=3,  
    M2_BEAMC_DLG=4,  
    DIV_BEAMMAP_DLG=5,  
    DIV_BEAMC_DLG=6,  
    WC_FLUENCE_DLG=7,  
    NUMERIC_DISPLAY_MODES=8,  
    EPROM_DATA=9,  
    LOGGING_DLG=10,  
    BS_PULSED_DLG=11,  
    WAVE_LENGTH_DLG=12,  
    CAPTURE_DLG=13,  
    PCI_EEPROM_DLG=14,  
    WANDER_DLG=15,  
    CENTROID_CLIP_DLG=16,  
    WC_IMAGE_LOG_SETUP_DLG=17,  
    WC_IMAGE_LOG_DLG=18,  
    WC_LOGGING_DLG=19,  
    BEAM_FIT_DLG=20,  
    TRIGGER_SETUP=21,  
    M_FACTOR_DLG=22,  
    GET_EWIDTH_CLIP=23,  
    UCM_TEST_DLG=24,  
};
```

Method: OpenFile

```
BOOL CGetDataOcx::OpenFile()  
Success = Ocx->OpenFile() ;  
Success = Getdata1.OpenFile()  
A dialog box will open.
```

Method: OpenPciCard

Obsolete do not use.

Method: OpenThisFile

```
long CGetDataOcx::OpenThisFile(LPCTSTR NameOfFile)  
Success = Ocx->OpenThisFile("C:\MyData\BeamScope.BSF") ;  
Success = Getdata1.OpenThisFile("C:\MyData\BeamScope.BSF")  
No dialog box will open.
```

Method: OpenWinCamD

Obsolete do not use.

Property: Palette  
short CGetDataOcx::GetPalette()  
void CGetDataOcx::SetPalette(short propVal)  
Ocx->SetPalette(PAL\_256) ; // Full color  
Getdata1.SetPalette(PAL\_GRAY) rem gray scale  
enum \_Palettes\_ {  
    PAL\_256=1,  
    PAL\_GRAY=2,  
    PAL\_16=3,  
    PAL\_10=4,  
    MAX\_PAL=5,  
};

Property: Parent  
Factory use only.

Property: PersistentData  
Factory use only.

Method: PreviousProfile  
BOOL CGetDataOcx:: PreviousProfile ()  
Ocx-> PreviousProfile () ;  
Getdata1.NextProfile()  
Decrements to the previous data buffer index.

Property: PrintNotes  
Factory use only.

Property: PurgeAllData  
Factory use only.

Method: PutToClipboard  
BOOL CGetDataOcx::PutToClipboard(long ThisAsLong)  
Ocx-> PutToClipboard (long(this)) ;  
An utility that copies the current object (this) to the clipboard.

Property: RangeLock  
Factory use only.

Method: ResetCamera  
Obsolete do not use.

Method: ResetCameras  
Obsolete do not use.

Method: SaveCurrentData  
BOOL CGetDataOcx:: SaveCurrentData(LPCTSTR FileNameAndPath)  
Success = Ocx-> SaveCurrentData("c:\\Program Files\\DataRay\\WinCamTest.txt") ;  
Success = Getdata1. SaveCurrentData("c:\\Program Files\\DataRay\\WinCamTest.txt")  
Note that the extensions how the file is saved:  
    .BIN = binary  
    .TXT = text  
    .WCF = wincam  
    .BCF = BeamMap-Collinate ect.

Method: SaveCurrentDataBuffer  
BOOL CGetDataOcx:: SaveCurrentDataBuffer(LPCTSTR NameOfFileWithPath)  
Success = Ocx->SaveCurrentDataBuffer ("c:\\Program Files\\DataRay\\WinCamTest") ;  
Success = Getdata1.SaveCurrentDataBuffer ("c:\\Program Files\\DataRay\\WinCamTest")  
Saved in device dependant manner. Ie WinCam = .WCF.

Method: SaveFile  
**BOOL CGetDataOcx::SaveFile()**  
Success = Ocx->SaveFile() ;  
Success = Getdata1.SaveFile()  
A dialog will open to complete the method.

Method: SaveJobFile  
**long CGetDataOcx::SaveJobFile()**  
Success = Ocx->SaveJobFile() ;  
Success = Getdata1. SaveJobFile()  
A dialog will open to complete the method.

Method: SelectProfile  
**BOOL CGetDataOcx::SelectProfile()**  
Success = Ocx->SelectProfile() ;  
Success = Getdata1.SelectProfile() ;  
A dialog will open to complete the method.

Method: SetAverage  
Obsolete do not use SetAverageNumber.

Method: SetAverageNumber  
**long CGetDataOcx::SetAverageNumber(long NumberToAverage)**  
averages = Ocx->SetAverageNumber(5) ;  
averages = Getdata1.SetAverageNumber(5)  
The return value is what was set.

Method: SetClipLevel  
**BOOL CGetDataOcx::SetClipLevel(double Clip1, double Clip2, short Mode1, short Mode2)**  
Success = Ocx->SetClipLevel(.135, 0.5, CLIP\_LEVEL\_METHOD, CLIP\_LEVEL\_METHOD) ;  
Success = Getdata1.SetClipLevel(.135, 0.5, CLIP\_LEVEL\_METHOD, CLIP\_LEVEL\_METHOD)  
enum WidthModes {  
    CLIP\_LEVEL\_METHOD=0,  
    SIGMA4\_METHOD=1,  
    KNIFE\_EDGE\_METHOD=2  
};  
No dialog will open.

Method: SetControlState  
Obsolete do not use StartDevice and StopDevice.

Method: SetCurrentDevice  
**short CGetDataOcx::SetCurrentDevice(short DeviceType)**  
device = Ocx->SetCurrentDevice(IS\_WINCAM) ;  
device = Getdata1.SetCurrentDevice(IS\_WINCAM) ;  
enum \_Selected\_Device\_ {  
    IS\_BEAMSCOPE=1,  
    IS\_BEAMR=2,  
    IS\_BEAMMAP=3,  
    IS\_BEAMC=4,  
    IS\_WINCAM=5,  
    IS\_WINCAM\_DIV=6,  
    IS\_WINCAM\_LOG=7,  
    IS\_TWOD\_SCAN=8,  
    IS\_SPARE4=9,  
    LAST\_DEVICE=10,  
};  
Selects the DataRay device class to enable.

Method: SetDefaultXcPlane  
**long CGetDataOcx::SetDefaultXcPlane(long DefaultXcPlane)**  
Ocx->SetDefaultXcPlane(2) ; // set to the THIRD plane  
Getdata1.SetDefaultXcPlane(0) rem set to the FIRST plane

Method: SetDisplayMode  
Factory use only.

Method: SetEffectiveWidthCliplevel  
**double CGetDataOcx::SetEffectiveWidthCliplevel(double NewClipLevel)**  
clip = Ocx->SetEffectiveWidthCliplevel(.135) ;  
clip = Getdata1.SetEffectiveWidthCliplevel(.135)  
Sets the clip level used to calculate Effective Width, 13.5% is default.

Method: SetGamma  
Obsolete do not use.

Method: SetHeadAngle  
Factory use only.

Method: SetHeadRotationAsFixed  
Factory use only.

Method: SetLiveRecallState  
**BOOL CGetDataOcx::SetLiveRecallState(short NewState\_0\_IS\_LIVE)**  
State = Ocx-> SetLiveRecallState(\_LIVE\_) ;  
Getdata1.SetLiveRecallState(\_LIVE\_)  
Method for switching from live (active) and recalled modes.  
#define           \_LIVE\_           0  
#define           \_RECALL\_        1

Method: SetNonuniformityOnOff  
Factory use only.

Method: SetRealTimeLogging  
Factory use only.

Method: SetROI  
**BOOL CGetDataOcx::SetROI(long Left, long Top, long Width, long Height)**  
Success = Ocx->SetROI(Left,Top, Width, Height) ;  
Success = Getdata1.SetROI(Left,Top, Width, Height)  
WinCam only, sets the capture region. Note: do not compesate for FAST vs FULL.

Method: SetToAbsolute  
Factory use only.

Method: SetToZero  
Factory use only.

Method: SetWorkingDirectory  
**BOOL CGetDataOcx::SetWorkingDirectory(LPCTSTR WorkingDirectory)**  
Success = Ocx->SetWorkingDirectory("C:\\MyDirectory") ;  
Success = Getdata1.SetWorkingDirectory("C:\\MyDirectory")  
Change program directory from default ("C:\\Program Files\\DataRay") ;

Property: ShowEffectiveSlits  
BOOL CGetDataOcx::GetShowEffectiveSlits()  
EffectiveShown = Ocx->GetShowEffectiveSlits() ;  
EffectiveShown = Getdata1.GetShowEffectiveSlits()  
WinCam only, (CCD view) returns whether effective slits are shown or not.

Property: SizeToggle  
Obsolete do not use.

Property: SlitsUsed  
long CGetDataOcx::GetSlitsUsed()  
void CGetDataOcx::SetSlitsUsed(long propVal)  
slits = Ocx->GetSlitsUsed() ;  
slits = Getdata1.GetSlitsUsed  
BeamScope only, 1 = X only, 2 = Y only and 3 = both X&Y.

\*\* Method: StartDevice  
BOOL CGetDataOcx::StartDevice()  
Success = Ocx->StartDevice() ;  
Success = Getdata1.StartDevice()  
Starts data collection for selected device (see SetCurrentDevice)

\*\* Method: StartDriver  
BOOL CGetDataOcx::StartDriver()  
Success = Ocx-> StartDriver () ;  
Success = Getdata1. StartDriver ()  
This MUST be the first method called, part of the initialization.

\*\* Method: StopDevice  
BOOL CGetDataOcx::StoptDevice()  
Success = Ocx->StopDevice() ;  
Success = Getdata1. StopDevice()  
Stops data collection for selected device (see SetCurrentDevice)

Property: StopMotorAtExit  
BOOL CGetDataOcx::GetStopMotorAtExit()  
void CGetDataOcx::SetStopMotorAtExit(BOOL propVal)  
StopsAtExit – Ocx-> GetStopMotorAtExit() ;  
Getdata1.SetStopMotorAtExit = True  
BeamMap. Beam'R and BeamMap-Collimate only.

Property: Tag  
Obsolete do not use.

Method: ToggleAnimation  
Factory use only.

Method: ToggleDialog  
Factory use only.

Method: Update  
void CGetDataOcx::Update()  
Ocx->Update() ;  
Getdata1.Update()  
Forces re-calculations and update all open DataRay OCX objects.

Method: UpdateAllButtons  
void CGetDataOcx::UpdateAllButtons()  
Ocx->UpdateAllButtons() ;  
Getdata1.UpdateAllButtons()  
Forces all open DataRay Button OCX objects to redraw.

Property: UseEffectiveSlits  
BOOL CGetDataOcx::GetUseEffectiveSlits()  
void CGetDataOcx::SetUseEffectiveSlits(BOOL propVal)  
UsingEffSlits = Ocx->GetUseEffectiveSlits() ;  
Getdata1.SetUseEffectiveSlits = False  
WinCam only.

Property: UseISO11146  
BOOL CGetDataOcx::GetUseISO11146()  
void CGetDataOcx::SetUseISO11146(BOOL propVal)  
UsingISO11146 = GetUseISO11146() ;  
Getdata1.SetUseISO11146 = False

Property: Wavelength  
double CGetDataOcx::GetWavelength()  
void CGetDataOcx::SetWavelength(double propVal)  
CurrentWl = Ocx->GetWavelength() ;  
Getdata1.SetWavelength = .672 rem set to 672 nm

Method: WinCamDDivergenceCameras  
Factory use only.

Property: WinCamFilter  
short CGetDataOcx::GetWinCamFilter()  
void CGetDataOcx::SetWinCamFilter(short propVal)  
FilterValue = Ocx->GetWinCamFilter() ;  
Getdata1.SetWinCamFilter = 0 rem No filter  
WinCam only. 0 or 1 = no filter, 2 = 2 by 2 and 3 = 3 by 3 filter.

Property: WinCamNormalized  
short CGetDataOcx::GetWinCamNormalized()  
void CGetDataOcx::SetWinCamNormalized(short propVal)  
IsNormalized = Ocx->GetWinCamNormalized() ;  
Getdata1.SetWinCamNormalized = False  
WinCam only, CCD view normalization on(TRUE) or off(FALSE).

New Methods added:

```
[id(159)] long GetCameraType();  
[id(160)] boolean PessButton(long Button_ID, int Left_Button);  
[id(161)] boolean StartBeamScopeM2();  
[id(162)] boolean StopBeamScopeM2();  
[id(163)] double BeamScopeSetStagePosition(double NewPositionInMicrons);  
[id(164)] double BeamScopeSetM2StartPosition(double StartPositionInMicons);  
[id(165)] boolean BeamScopeSetM2StoptPosition(short StopPositionInMicrons);  
[id(166)] boolean BeamScopeM2HomeStage();  
[id(167)] double GetWinCamGain(long WhichCamera);
```

Complete listing as of Feb 13, 2006

```
// DataRayOcx.odl : type library source for ActiveX Control project.  
  
// This file will be processed by the Make Type Library (mktyplib) tool to  
// produce the type library (DataRayOcx.tlb) that will become a resource in  
// DataRayOcx.ocx.
```

```
#include <olectl.h>  
#include <idispids.h>
```

```
[ uuid(43555BA2-3FE0-11D6-9F4A-00A0CC40A4D2), version(1.0),  
  helpfile("DataRayOcx.hlp"),  
  helpstring("DataRayOcx ActiveX Control module"),  
  control ]
```

```
library DATARAYOCXLib
```

```
{
```

```
    importlib(STDOLE_TLB);  
    importlib(STDTYPE_TLB);
```

```
    // Primary dispatch interface for CProfilesCtrl
```

```
    [ uuid(43555BA3-3FE0-11D6-9F4A-00A0CC40A4D2),  
      helpstring("Dispatch interface for Profiles Control"), hidden ]  
    dispinterface _DProfiles
```

```
    {
```

```
        properties:
```

```
            // NOTE - ClassWizard will maintain property information here.  
            // Use extreme caution when editing this section.  
            {{{AFX_ODL_PROP(CProfilesCtrl)  
            [id(2)] short MyID;  
            [id(1)] short ProfileID;  
            [id(3)] boolean NoButtonsPlease;  
            }}}AFX_ODL_PROP
```

```
        methods:
```

```
            // NOTE - ClassWizard will maintain method information here.  
            // Use extreme caution when editing this section.
```

```

        {{{AFX_ODL_METHOD(CProfilesCtrl)
        [id(4)] boolean GetProfileData(long* LongBuffer_32bit, long NumberOfLongs);
        }}}AFX_ODL_METHOD
};

// Event dispatch interface for CProfilesCtrl

[ uuid(43555BA4-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for Profiles Control") ]
dispinterface _DProfilesEvents
{
    properties:
        // Event interface has no properties

    methods:
        // NOTE - ClassWizard will maintain event information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_EVENT(CProfilesCtrl)
        }}}AFX_ODL_EVENT
};

// Class information for CProfilesCtrl

[ uuid(43555BA5-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Profiles Control"), control ]
coclass Profiles
{
    [default] dispinterface _DProfiles;
    [default, source] dispinterface _DProfilesEvents;
};

// Primary dispatch interface for CTwoDCtrl

[ uuid(43555BA7-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for TwoD Control"), hidden ]
dispinterface _DTwoD
{
    properties:

```

```

        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_PROP(CTwoDCtrl)
        //}}AFX_ODL_PROP

    methods:
        // NOTE - ClassWizard will maintain method information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_METHOD(CTwoDCtrl)
        //}}AFX_ODL_METHOD
};

// Event dispatch interface for CTwoDCtrl

[ uuid(43555BA8-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for TwoD Control") ]
dispinterface _DTwoDEvents
{
    properties:
        // Event interface has no properties

    methods:
        // NOTE - ClassWizard will maintain event information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_EVENT(CTwoDCtrl)
        //}}AFX_ODL_EVENT
};

// Class information for CTwoDCtrl

[ uuid(43555BA9-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("TwoD Control"), control ]
coclass TwoD
{
    [default] dispinterface _DTwoD;
    [default, source] dispinterface _DTwoDEvents;
};

```

```

// Primary dispatch interface for CCCDImageCtrl

[ uuid(43555BAB-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for CCDImage Control"), hidden ]
dispinterface _DCCDImage
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_PROP(CCCDImageCtrl)
        //}}AFX_ODL_PROP

    methods:
        // NOTE - ClassWizard will maintain method information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_METHOD(CCCDImageCtrl)
        [id(1)] boolean Apply();
        //}}AFX_ODL_METHOD
};

```

```

// Event dispatch interface for CCCDImageCtrl

```

```

[ uuid(43555BAC-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for CCDImage Control") ]
dispinterface _DCCDImageEvents
{
    properties:
        // Event interface has no properties

    methods:
        // NOTE - ClassWizard will maintain event information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_EVENT(CCCDImageCtrl)
        //}}AFX_ODL_EVENT
};

```

```

// Class information for CCCDImageCtrl

```

```

[ uuid(43555BAD-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("CCDImage Control"), control ]
coclass CCDImage
{
    [default] dispinterface _DCCDImage;
    [default, source] dispinterface _DCCDImageEvents;
};

// Primary dispatch interface for CThreeDviewCtrl

[ uuid(43555BAF-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for ThreeDview Control"), hidden ]
dispinterface _DThreeDview
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_PROP(CThreeDviewCtrl)
        //}}AFX_ODL_PROP

    methods:
        // NOTE - ClassWizard will maintain method information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_METHOD(CThreeDviewCtrl)
        //}}AFX_ODL_METHOD
};

// Event dispatch interface for CThreeDviewCtrl

[ uuid(43555BB0-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for ThreeDview Control") ]
dispinterface _DThreeDviewEvents
{
    properties:
        // Event interface has no properties

    methods:
        // NOTE - ClassWizard will maintain event information here.

```

```

        // Use extreme caution when editing this section.
        ///{AFX_ODL_EVENT(CThreeDviewCtrl)
        //}}AFX_ODL_EVENT
};

// Class information for CThreeDviewCtrl

[ uuid(43555BB1-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("ThreeDview Control"), control ]
coclass ThreeDview
{
    [default] dispinterface _DThreeDview;
    [default, source] dispinterface _DThreeDviewEvents;
};

// Primary dispatch interface for CPaletteBarCtrl

[ uuid(43555BB3-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for PaletteBar Control"), hidden ]
dispinterface _DPaletteBar
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        ///{AFX_ODL_PROP(CPaletteBarCtrl)
        //}}AFX_ODL_PROP

    methods:
        // NOTE - ClassWizard will maintain method information here.
        // Use extreme caution when editing this section.
        ///{AFX_ODL_METHOD(CPaletteBarCtrl)
        //}}AFX_ODL_METHOD
};

// Event dispatch interface for CPaletteBarCtrl

[ uuid(43555BB4-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for PaletteBar Control") ]

```

```

dispinterface _DPaletteBarEvents
{
    properties:
        // Event interface has no properties

    methods:
        // NOTE - ClassWizard will maintain event information here.
        // Use extreme caution when editing this section.
        //{AFX_ODL_EVENT(CPaletteBarCtrl)
        //}AFX_ODL_EVENT
};

// Class information for CPaletteBarCtrl

[ uuid(43555BB5-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("PaletteBar Control"), control ]
coclass PaletteBar
{
    [default] dispinterface _DPaletteBar;
    [default, source] dispinterface _DPaletteBarEvents;
};

// Primary dispatch interface for CGetDataCtrl

[ uuid(43555BB7-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for GetData Control"), hidden ]
dispinterface _DGetData
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        //{AFX_ODL_PROP(CGetDataCtrl)
        [id(1)] double FilterValue;
        [id(2)] long IsMSquaredOpen;
        [id(3)] short Palette;
        [id(4)] BSTR PrintNotes;
        [id(5)] short InkSaverState;
        [id(6)] short JitterSuppression;
};

```

```
[id(7)] long SlitsUsed;
[id(8)] double Wavelength;
[id(9)] short IsDivergenceOpen;
[id(10)] short WinCamFilter;
[id(11)] short FastUpdate;
[id(12)] short CameraSelect;
[id(13)] short CurrentCamera;
[id(14)] short AutoSnap;
[id(15)] short SizeToggle;
[id(16)] short BaselineLocked;
[id(17)] short WinCamNormalized;
[id(18)] short BackGroundSubtraction;
[id(19)] short AutoNaming;
[id(20)] long CustomSetup;
[id(21)] double CentroidClipLevel;
[id(22)] long CameraSequence;
[id(23)] double GeoClipLevel;
[id(24)] long EffectiveCentroidFilterInPixels;
[id(25)] boolean eTrapOn;
[id(26)] boolean AutoShutterOn;
[id(27)] boolean UseISO11146;
[id(28)] boolean RangeLock;
[id(29)] boolean LockAll;
[id(30)] boolean StopMotorAtExit;
[id(31)] boolean UseEffectiveSlits;
[id(32)] boolean ShowEffectiveSlits;
[id(33)] boolean AtAim;
[id(34)] long BeamMapCdefaultXc;
[id(35)] long WinCamDDivergenceCameras;
[id(36)] short CentroidType;
[id(37)] long DontUsePLS;
//}}AFX_ODL_PROP
```

methods:

```
// NOTE - ClassWizard will maintain method information here.
// Use extreme caution when editing this section.
//{{AFX_ODL_METHOD(CGetDataCtrl)
[id(38)] BSTR GetLastError();
```

[id(39)] boolean OpenBeamScopeP7(long Identity\_HWND, long WhichBoard);

[id(40)] boolean IsBeamScopeP7HeadThere();

[id(41)] boolean DoSearch();

[id(42)] boolean HomeP7Head();

[id(43)] boolean BeamScopeScanControl(long NumberOfScans, long FirstSlitON,  
long SecondSlitOn);

[id(44)] boolean SaveFile();

[id(45)] boolean OpenFile();

[id(46)] boolean PreviousProfile();

[id(47)] boolean NextProfile();

[id(48)] boolean SelectProfile();

[id(49)] void PurgeAllData();

[id(50)] double GetOcxResult(short IndexToValue);

[id(51)] BSTR GetOcxResultName(short IndexToValue);

[id(52)] boolean OpenClipLevelDlg(short ClipOneOrTwo\_0\_1);

[id(53)] double GetClipLevel(short ClipOneOrTwo\_0\_1);

[id(54)] short GetClipLevelMode(short ClipOneOrTwo\_0\_1);

[id(55)] long OpenPciCard(long BoardNumber, long TypeBoard, long VenderID,  
short DeviceID);

[id(56)] long GetFirmwareRevInfo(long BoardID, long Index);

[id(57)] void LoadDefaults();

[id(58)] long GetBeamScopeIndex();

[id(59)] long GetBeamScopeState();

[id(60)] long SetAverageNumber(long NumberToAverage);

[id(61)] long SaveJobFile();

[id(62)] long LoadJobFile();

[id(63)] boolean FingerToPosition(double Position);

[id(64)] long HowManyBoardsOpened();

[id(65)] long SetAverage(long NumberToAverage, long Reset, long  
ResetAtMircons);

[id(66)] boolean SetClipLevel(double Clip1, double Clip2, short Mode1, short  
Mode2);

[id(67)] boolean SetDisplayMode(short DisplayMode);

[id(68)] BSTR GetOcxRev();

[id(69)] boolean SetControlState(short WhichControl, short State\_0\_NOT0);

[id(70)] void Update();

[id(71)] boolean OpenBeamMap(short WhichBoard);

[id(72)] short SetCurrentDevice(short DeviceType);

[id(73)] long OpenThisFile(BSTR NameOfFile);

[id(74)] short GetCurrentDevice();  
[id(75)] short GetCurrentState();  
[id(76)] short GetCurrentIndex();  
[id(77)] boolean SetLiveRecallState(short NewState\_0\_IS\_LIVE);  
[id(78)] short GetSampleCount(short Live\_Is\_0);  
[id(79)] BSTR GetRecallFieName();  
[id(80)] long GetSavedDataPointer();  
[id(81)] short OpenDialog(short IndexToDialog);  
[id(82)] short CloseDialog(short IndexToDialog);  
[id(83)] boolean DeviceRunning();  
[id(84)] boolean StartDevice();  
[id(85)] boolean StopDevice();  
[id(86)] short GetAverageNumber();  
[id(87)] BSTR GetRecallFieVersion();  
[id(88)] boolean OpenWinCamD(short WhichBoard);  
[id(89)] short SetToZero();  
[id(90)] short SetToAbsolute();  
[id(91)] short GetHorizontalPixels();  
[id(92)] short GetVerticalPixels();  
[id(93)] boolean CaptureIsFullResolution();  
[id(94)] boolean IsCameraThere(short WhichCamera\_0\_1);  
[id(95)] BSTR GetHelpString();  
[id(96)] void FindWinCamImage();  
[id(97)] void GetWinCamSingle();  
[id(98)] long GetShutterSetting();  
[id(99)] void ExportToPaint(long ThiPointer);  
[id(100)] short ToggleDialog(short IndexOfDialog);  
[id(101)] boolean ExportAsBitMap(long ThisAsLong);  
[id(102)] boolean PutToClipboard(long ThisAsLong);  
[id(103)] BSTR GetSoftwareVersion();  
[id(104)] double GetWinCamDPixelSize(short X\_0\_Y\_1);  
[id(105)] double GetParameter(short IndexToValue);  
[id(106)] boolean StartDriver();  
[id(107)] void ResetCameras();  
[id(108)] short ResetCamera(short WhichCamera);  
[id(109)] BSTR GetSaveFileName();  
[id(110)] long GetProfileTop();  
[id(111)] double GetOcxResultExt(long WhichResult, long WhichCamera);

[id(112)] boolean IsDataReady(short Index);  
[id(168), propget] double PersistantData(short Index);  
[id(168), propput] void PersistantData(short Index, double newValue);  
[id(113)] long GetWinCamTraps();  
[id(114)] void ForceCrosshairsToZero();  
[id(115)] void ForceCrosshairsTo45();  
[id(116)] void SetGamma(double NewGamma);  
[id(117)] void EnableEffectiveCentroidsY();  
[id(118)] void EnableEffectiveCentroidsX();  
[id(119)] void DisableEffectiveCentroidsY();  
[id(120)] void DisableEffectiveCentroidsX();  
[id(121)] double GetEffectiveCentroidY(long WhichCamera);  
[id(122)] double GetEffectiveCentroidX(long WhichCamera);  
[id(123)] double GetEffectiveGeoCenterY(long WhichCamera);  
[id(124)] double GetEffectiveGeoCenterX(long WhichCamera);  
[id(169), propget] double Exposure(long WhichCamera);  
[id(169), propput] void Exposure(long WhichCamera, double newValue);  
[id(125)] void KeyEvent(short KeyCode, short KeyCount);  
[id(126)] long GetPixel(long x, long y);  
[id(127)] boolean SaveCurrentData(BSTR FileNameAndPath);  
[id(128)] long EnableUseEffectiveSlits(long Enable);  
[id(129)] boolean IsHeadRotationFixed();  
[id(130)] void SetHeadRotationAsFixed(long HeadRotationFixed\_0\_1);  
[id(131)] void SetHeadAngle(double AngleInRadians);  
[id(132)] boolean IsHeadStalled();  
[id(133)] double GetHeadAngle();  
[id(134)] boolean GetErrorStatus();  
[id(135)] void UpdateAllButtons();  
[id(136)] long GetPeakXlocation();  
[id(137)] long GetPeakYlocation();  
[id(138)] long GetCentroidXlocation();  
[id(139)] long GetCentroidYlocation();  
[id(140)] long SetDefaultXcPlane(long DefaultXcPlane);  
[id(141)] double SetEffectiveWidthCliplevel(double NewClipLevel);  
[id(142)] double GetEffectiveWidthCliplevel();  
[id(143)] long SetRealTimeLogging(long EnabledIsNotZero);  
[id(144)] long GetRealTimeLogging();  
[id(145)] long SetNonuniformityOnOff(long NonZeroIsOn);

```

        [id(146)] long GetNonuniformityOnOff();
        [id(147)] boolean GetCurrentWinCamData(long* ImageDataPt, long* XSizePt, long*
YSizePtr);
        [id(148)] boolean SetROI(long Left, long Top, long Width, long Height);
        [id(149)] boolean GetROI(long* LeftAsLongPointer, long* TopAsLongPointer, long*
WidthAsLongPointer, long* HeightAsLongPointer);
        [id(150)] boolean ToggleAnimation();
        [id(151)] boolean SetWorkingDirectory(BSTR WorkingDirectory);
        [id(152)] boolean LoadThisJobFile(BSTR JobFileNamePath);
        [id(153)] double GetIncludedPowerPercentAtRadius(double RadiusInMicrons);
        [id(154)] double GetIncludedPowerTotal();
        [id(155)] boolean SaveCurrentDataBuffer(BSTR NameOfFileWithPath);
        [id(156)] boolean GetWinCamSingleAndComplete();
        [id(157)] double GetRadiusAtPowerPercent(double PowerPercent);
        [id(158)] void AutoCrosshairs();
        [id(159)] long GetCameraType();
        [id(160)] boolean PressButton(long Button_ID, int Left_Button);
        [id(161)] boolean StartBeamScopeM2();
        [id(162)] boolean StopBeamScopeM2();
        [id(163)] double BeamScopeSetStagePosition(double NewPositionInMicrons);
        [id(164)] double BeamScopeSetM2StartPosition(double StartPositionInMicons);
        [id(165)] boolean BeamScopeSetM2StoptPosition(short StopPositionInMicrons);
        [id(166)] boolean BeamScopeM2HomeStage();
        [id(167)] double GetWinCamGain(long WhichCamera);
    //}}AFX_ODL_METHOD
};

```

```

// Event dispatch interface for CGetDataCtrl

```

```

[ uuid(43555BB8-3FE0-11D6-9F4A-00A0CC40A4D2),

```

```

    helpstring("Event interface for GetData Control") ]

```

```

dispinterface _DGetDataEvents

```

```

{

```

```

    properties:

```

```

        // Event interface has no properties

```

```

    methods:

```

```

        // NOTE - ClassWizard will maintain event information here.

```

```

        // Use extreme caution when editing this section.

```

```

        {{{AFX_ODL_EVENT(CGetDataCtrl)
        [id(1)] void SendMessage(long Message, long LongValue, double DoubleValue);
        [id(2)] void DataReady();
        //}}AFX_ODL_EVENT
};

// Class information for CGetDataCtrl

[ uuid(43555BB9-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("GetData Control"), control ]
coclass GetData
{
    [default] dispinterface _DGetData;
    [default, source] dispinterface _DGetDataEvents;
};

// Primary dispatch interface for CButtonCtrl

[ uuid(43555BBB-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for Button Control"), hidden ]
dispinterface _DButton
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_PROP(CButtonCtrl)
        [id(2)] long ButtonID;
        //}}AFX_ODL_PROP

    methods:
        // NOTE - ClassWizard will maintain method information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_METHOD(CButtonCtrl)
        //}}AFX_ODL_METHOD

        [id(DISPID_ABOUTBOX)] void AboutBox();
};

```

```

// Event dispatch interface for CButtonCtrl

[ uuid(43555BBC-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for Button Control") ]
dispinterface _DButtonEvents
{
    properties:
        // Event interface has no properties

    methods:
        // NOTE - ClassWizard will maintain event information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_EVENT(CButtonCtrl)
        //}}AFX_ODL_EVENT
};

// Class information for CButtonCtrl

[ uuid(43555BBD-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Button Control"), control ]
coclass Button
{
    [default] dispinterface _DButton;
    [default, source] dispinterface _DButtonEvents;
};

// Primary dispatch interface for CShuterControlCtrl

[ uuid(43555BBF-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for ShuterControl Control"), hidden ]
dispinterface _DShuterControl
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_PROP(CShuterControlCtrl)
        //}}AFX_ODL_PROP
};

```

```

methods:
    // NOTE - ClassWizard will maintain method information here.
    // Use extreme caution when editing this section.
    //{AFX_ODL_METHOD(CShuterControlCtrl)
[id(1)] boolean SetID(short ScrollID);
    //}AFX_ODL_METHOD

[id(DISPID_ABOUTBOX)] void AboutBox();
};

// Event dispatch interface for CShuterControlCtrl

[ uuid(43555BC0-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for ShuterControl Control") ]
dispinterface _DShuterControlEvents
{
    properties:
        // Event interface has no properties

    methods:
        // NOTE - ClassWizard will maintain event information here.
        // Use extreme caution when editing this section.
        //{AFX_ODL_EVENT(CShuterControlCtrl)
        //}AFX_ODL_EVENT
};

// Class information for CShuterControlCtrl

[ uuid(43555BC1-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("ShuterControl Control"), control ]
coclass ShuterControl
{
    [default] dispinterface _DShuterControl;
    [default, source] dispinterface _DShuterControlEvents;
};

// Primary dispatch interface for CTriggerControlCtrl

```

```

[ uuid(43555BC3-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for TriggerControl Control"), hidden ]
dispinterface _DTriggerControl
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_PROP(CTriggerControlCtrl)
        //}}AFX_ODL_PROP

    methods:
        // NOTE - ClassWizard will maintain method information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_METHOD(CTriggerControlCtrl)
        //}}AFX_ODL_METHOD

        [id(DISPID_ABOUTBOX)] void AboutBox();
};

```

```
// Event dispatch interface for CTriggerControlCtrl
```

```

[ uuid(43555BC4-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for TriggerControl Control") ]
dispinterface _DTriggerControlEvents
{
    properties:
        // Event interface has no properties

    methods:
        // NOTE - ClassWizard will maintain event information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_EVENT(CTriggerControlCtrl)
        //}}AFX_ODL_EVENT
};

```

```
// Class information for CTriggerControlCtrl
```

```
[ uuid(43555BC5-3FE0-11D6-9F4A-00A0CC40A4D2),
```

```

    helpstring("TriggerControl Control"), control ]
coclass TriggerControl
{
    [default] dispinterface _DTriggerControl;
    [default, source] dispinterface _DTriggerControlEvents;
};

// Primary dispatch interface for CDataRayOcx10Ctrl

[ uuid(43555BC7-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Dispatch interface for DataRayOcx10 Control"), hidden ]
dispinterface _DDataRayOcx10
{
    properties:
        // NOTE - ClassWizard will maintain property information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_PROP(CDataRayOcx10Ctrl)
        //}}AFX_ODL_PROP

    methods:
        // NOTE - ClassWizard will maintain method information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_METHOD(CDataRayOcx10Ctrl)
        //}}AFX_ODL_METHOD

        [id(DISPID_ABOUTBOX)] void AboutBox();
};

// Event dispatch interface for CDataRayOcx10Ctrl

[ uuid(43555BC8-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("Event interface for DataRayOcx10 Control") ]
dispinterface _DDataRayOcx10Events
{
    properties:
        // Event interface has no properties

    methods:

```

```

        // NOTE - ClassWizard will maintain event information here.
        // Use extreme caution when editing this section.
        {{{AFX_ODL_EVENT(CDataRayOcx10Ctrl)
        //}}AFX_ODL_EVENT
};

// Class information for CDataRayOcx10Ctrl

[ uuid(43555BC9-3FE0-11D6-9F4A-00A0CC40A4D2),
  helpstring("DataRayOcx10 Control"), control ]
coclass DataRayOcx10
{
    [default] dispinterface _DDataRayOcx10;
    [default, source] dispinterface _DDataRayOcx10Events;
};

{{{AFX_APPEND_ODL}}
//}}AFX_APPEND_ODL}}
};

```