

DataRay Inc.

Advancing the Technology of Laser Beam Analysis

DataRay OCX Documentation

Version 1.0
March 20th 2015

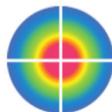


Table of Contents

Class List

Here is the list of supported classes; undocumented classes (eg. DataRayGetData) and methods are deprecated.

- [Button](#) (Dispatch interface for Button ActiveX controls; 'DATARAYOCX.ButtonCtrl.1')2
- [CCDImage](#) (Dispatch interface for CCD Image ActiveX control; 'DATARAYOCX.CCDImageCtrl.1')3
- [GetData](#) (Primary dispatch interface for GetData ActiveX control; 'DATARAYOCX.GetDataCtrl.1') ..4
- [GetDataEvents](#) (Event dispatch interface for GetData ActiveX control)29
- [PaletteBar](#) (Dispatch interface for PaletteBar ActiveX control; 'DATARAYOCX.PaletteBarCtrl.1')30
- [Profiles](#) (Dispatch interface for Profile ActiveX controls; 'DATARAYOCX.ProfilesCtrl.1')30
- [ShuterControl](#) (Dispatch interface for ShuterControl; 'DATARAYOCX.ShuterControlCtrl.1')32
- [ThreeDview](#) (Dispatch interface for ThreeDView ActiveX ctrl; 'DATARAYOCX.ThreeDviewCtrl.1') ...33
- [TriggerControl](#) (Dispatch interface for Trigger ActiveX ctrl; 'DATARAYOCX.TriggerControlCtrl.1') ...33
- [TwoD](#) (Dispatch interface for TwoD ActiveX control; 'DATARAYOCX.TwoDCtrl.1')34

Class Documentation

Button Interface Reference

Dispatch interface for Button ActiveX controls

Public Member Functions

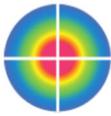
- boolean [PutImagetoClipboard](#) ()
Puts image to clipboard
- boolean [SaveImagetoFile](#) (BSTR FileNameWithPath)
Saves image to designated file
- double [GetParameter](#) ()
Returns the current value for what the button displays
- void [AboutBox](#) ()
Displays an about box

Properties

- long [ButtonID](#)
Sets button type as defined by ID number which must range from 1 to 440

Detailed Description

Dispatch interface for Button ActiveX controls



Member Function Documentation

boolean SaveImagetoFile (BSTR *FileNameWithPath*)

Saves image to designated file

Parameters:

<i>FileNameWithPath</i>	The filename and path combined
-------------------------	--------------------------------

CCDimage Interface Reference

Dispatch interface for CCD Image ActiveX control

Public Member Functions

- boolean [PutImagetoClipboard](#) ()
Puts image to clipboard
- boolean [SaveImagetoFile](#) (BSTR *FileNameWithPath*)
Saves image to designated file

Detailed Description

Dispatch interface for CCD Image ActiveX control

Member Function Documentation

boolean SaveImagetoFile (BSTR *FileNameWithPath*)

Saves image to designated file

Parameters:

<i>FileNameWithPath</i>	The filename and path combined
-------------------------	--------------------------------

GetData Interface Reference

Primary dispatch interface for GetData ActiveX control

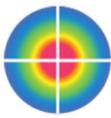


Public Member Functions

- BSTR [GetLastError](#) ()
Returns the last error
- boolean [IsBeamScopeP7HeadThere](#) ()
If it is a beam scope, it will return finding for head; otherwise, false
- boolean [DoSearch](#) ()
Searches for device and returns True if it is a BeamScope
- boolean [HomeP7Head](#) ()
Returns true if the homehead is there
- boolean [BeamScopeScanControl](#) (long NumberOfScans, long FirstSlitON, long SecondSlitOn)
Sets values for BeamScopeScan and returns true
- boolean [SaveFile](#) ()
Opens the save file dialog menu
- boolean [OpenFile](#) ()
Opens the open file dialog menu
- boolean [PreviousProfile](#) ()
Moves image and profiles back by one frame
- boolean [NextProfile](#) ()
Moves the image and profiles forward by one frame
- boolean [SelectProfile](#) ()
Opens the beam selection dialog to select a frame
- void [PurgeAllData](#) ()
Purges automatically recorded data from program; no frames will be available for selection
- double [GetOcxResult](#) (short IndexToValue)
Returns the current value for a button given its ID
- BSTR [GetOcxResultName](#) (short IndexToValue)
Returns the name for a button given its ID
- boolean [OpenClipLevelDlg](#) (short ClipOneOrTwo_0_1)
Opens the clip level dialog for the given clip
- double [GetClipLevel](#) (short ClipOneOrTwo_0_1)
Returns the current level for the given clip
- short [GetClipLevelMode](#) (short ClipOneOrTwo_0_1)
Returns the mode for the given clip
- long [OpenPciCard](#) (long BoardNumber, long TypeBoard, long VenderID, short DeviceID)
Opens the given PCI card with given information
- long [GetFirmwareRevInfo](#) (long BoardID, long Index)
Returns the firmware information
- void [LoadDefaults](#) ()
Load default settings for program
- long [GetBeamScopeIndex](#) ()
Returns the index (current frame) of the BeamScope
- long [GetBeamScopeState](#) ()
Returns 1 if the BeamScope is live and 0 otherwise
- long [SetAverageNumber](#) (long NumberToAverage)
Sets the number you want to average.



- long [SaveJobFile](#) ()
Opens the save job file dialog
- long [LoadJobFile](#) ()
Opens the load job file dialog
- boolean [FingerToPosition](#) (double Position)
Sets the aim to the position given as a double
- boolean [SetClipLevel](#) (double Clip1, double Clip2, short Mode1, short Mode2)
Sets parameters which affect the profile displays and measurements
- boolean [SetDisplayMode](#) (short DisplayMode)
Sets display mode in microns
- BSTR [GetOcxRev](#) ()
Returns the date of the last revision of OCX
- boolean [SetControlState](#) (short WhichControl, short State_0_NOT0)
Sets the state for a subset of controls
- short [SetCurrentDevice](#) (short DeviceType)
Set current device
- long [OpenThisFile](#) (BSTR NameOfFile)
Opens the given file
- short [GetCurrentDevice](#) ()
Returns current device as number; see table
- short [GetCurrentState](#) ()
Returns state of device; 0 is live and 1 is recall
- short [GetCurrentIndex](#) ()
Returns current index of device, the 0th to 64th frame
- boolean [SetLiveRecallState](#) (short NewState_0_IS_LIVE)
Toggle between live an recall state
- short [GetSampleCount](#) (short Live_Is_0)
Returns the sample count for given state; live is 0 and 1 is recall
- BSTR [GetRecallFieName](#) ()
Returns recall file name
- long [GetSavedDataPointer](#) ()
- short [OpenDialog](#) (short IndexToDialog)
Opens dialog defined by number; see list
- short [CloseDialog](#) (short IndexToDialog)
Closes dialog defined by number; see list
- boolean [DeviceRunning](#) ()
Returns device running status
- boolean [StartDevice](#) ()
Returns true on successful start of device
- boolean [StopDevice](#) ()
Returns true on successful stop of device
- short [GetAverageNumber](#) ()
Returns the number to average set by [SetAverageNumber](#)
- BSTR [GetRecallFieVersion](#) ()
Returns recall file version
- short [SetToZero](#) ()

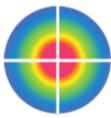


Based on current value, sets buttons to zero to display relative values

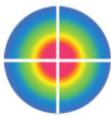
- short [SetToAbsolute](#) ()
Sets buttons to absolute setting () not absolute value*
- short [GetHorizontalPixels](#) ()
Returns the horizontal pixel size
- boolean [CaptureIsFullResolution](#) ()
Returns true if capture is set to full resolution
- boolean [IsCameraThere](#) (short WhichCamera_0_1)
Returns true if camera is there
- BSTR [GetHelpString](#) ()
Returns help string
- void [GetWinCamSingle](#) ()
Sets up 1st WinCam
- long [GetShutterSetting](#) ()
Returns shutter setting
- void [ExportToPaint](#) (long ThisPointer)
Exports to paint. Does not work in Windows 8 or later.
- short [ToggleDialog](#) (short IndexOfDialog)
Toggles dialog defined by number; see list
- boolean [PutToClipboard](#) (long ThisAsLong)
Takes screenshot regardless of input
- BSTR [GetSoftwareVersion](#) ()
Returns software version
- double [GetWinCamDPixelSize](#) (short X_0_Y_1)
Returns horizontal or vertical pixel size
- double [GetParameter](#) (short IndexToValue)
Gives value for button designated by ID number; similar to [GetOcxResult](#) but it must be a parameter only
- boolean [StartDriver](#) ()
Starts the driver
- void [ResetCameras](#) ()
Resets cameras
- short [ResetCamera](#) (short WhichCamera)
Resets target camera
- BSTR [GetSaveFileName](#) ()
Returns save file name
- long [GetProfileTop](#) ()
Returns profile display's height in pixels
- double [GetOcxResultExt](#) (long WhichResult, long WhichCamera)
Returns OCX result for given camera; see [GetOcxResult](#) for more information
- boolean [IsDataReady](#) (short Index)
Returns if data is ready
- double [PersistantData](#) (short Index)
Returns persistent data
- void [PersistantData](#) (short Index, double newValue)
Sets persistent data to given index
- void [ForceCrosshairsToZero](#) ()



- Forces crosshairs to zero instead of 45 degrees or auto orientation*
- void [ForceCrosshairsTo45](#) ()
Forces crosshairs to 45 instead of 0 degrees or auto orientation
- void [SetGamma](#) (double NewGamma)
Sets gamma value
- double [GetEffectiveCentroidY](#) (long WhichCamera)
Returns horizontal position of centroid for given camera
- double [GetEffectiveCentroidX](#) (long WhichCamera)
Returns vertical position of centroid for given camera
- double [GetEffectiveGeoCenterY](#) (long WhichCamera)
Returns horizontal position of geometric centroid for given camera
- double [GetEffectiveGeoCenterX](#) (long WhichCamera)
Returns vertical position of geometric centroid for given camera
- double [Exposure](#) (long WhichCamera)
Returns exposure for given camera
- void [Exposure](#) (long WhichCamera, double newValue)
Sets exposure for given camera
- void [KeyEvent](#) (short KeyCode, short KeyCount)
Relays events by keyboard input
- long [GetPixel](#) (long x, long y)
Returns pixel value for (x,y) coordinate position
- boolean [SaveCurrentData](#) (BSTR FileNameAndPath)
Saves data as given by filename and path variable
- long [EnableUseEffectiveSlits](#) (long Enable)
Enables use of effective slits
- boolean [IsHeadRotationFixed](#) ()
Returns status of fixed head rotation
- void [SetHeadRotationAsFixed](#) (long HeadRotationFixed_0_1)
Make head rotation fixed
- void [SetHeadAngle](#) (double AngleInRadians)
Set head angle in radians
- boolean [IsHeadStalled](#) ()
Returns if true if head is stalled
- double [GetHeadAngle](#) ()
Returns the active angle in radians
- boolean [GetErrorStatus](#) ()
Returns 1 if camera could have errors; eg. if it is in recall mode, it will return 0
- void [UpdateAllButtons](#) ()
Updates all buttons
- long [GetPeakXlocation](#) ()
Returns the peak in the horizontal direction, usually zero
- long [GetPeakYlocation](#) ()
Returns the peak in the vertical direction, usually zero
- long [GetCentroidXlocation](#) ()
Returns the horizontal coordinate of the centroid
- long [GetCentroidYlocation](#) ()



- Returns the vertical coordinate of the centroid*
- long [SetDefaultXcPlane](#) (long DefaultXcPlane)
For BeamMap, sets the default X plane
 - double [SetEffectiveWidthCliplevel](#) (double NewClipLevel)
Sets the effective clip width level
 - double [GetEffectiveWidthCliplevel](#) ()
Returns the effective clip level
 - long [SetRealTimeLogging](#) (long EnabledIsNotZero)
Enable real time logging
 - long [GetRealTimeLogging](#) ()
Returns real time logging status; 1 is on 0 is off
 - long [SetNonuniformityOnOff](#) (long NonZeroIsOn)
Toggle non-uniform;
 - long [GetNonuniformityOnOff](#) ()
Returns non-uniform status
 - boolean [GetCurrentWinCamData](#) (long *ImageDataPt, long *XSizePt, long *YSizePtr)
Upon successfully setting pointers to variables, returns true
 - boolean [SetROI](#) (long Left, long Top, long Width, long Height)
Sets the capture size and starting positions
 - boolean [GetROI](#) (long *LeftAsLongPointer, long *TopAsLongPointer, long *WidthAsLongPointer, long *HeightAsLongPointer)
Fills given pointers with capture size and starting position
 - boolean [SetWorkingDirectory](#) (BSTR WorkingDirectory)
Sets working directory for placement of DataRay files
 - boolean [LoadThisJobFile](#) (BSTR JobFileNamePath)
Loads a job file
 - double [GetIncludedPowerPercentAtRadius](#) (double RadiusInMicrons)
Gets the percentage of total power included at a given radius from the centroid
 - boolean [SaveCurrentDataBuffer](#) (BSTR NameOfFileWithPath)
Saves the current data buffer into one of the designated file types
 - double [GetRadiusAtPowerPercent](#) (double PowerPercent)
For a given percentage, returns the radius from centroid including that percentage of total power
 - void [AutoCrosshairs](#) ()
Forces crosshairs to be set automatically instead of 0 or 45 degrees
 - long [GetCameraType](#) ()
If WinCam, returns type of camera as defined by number in list, -1 otherwise; for full functionality, use [CameraType](#)
 - boolean [PressButton](#) (long Button_ID, int Left_Button)
Press button of given ID
 - boolean [StartBeamScopeM2](#) ()
Starts BeamScope M²; will open dialog menu
 - boolean [StopBeamScopeM2](#) ()
Stops M²
 - double [BeamScopeSetStagePosition](#) (double NewPositionInMicrons)
Sets the stage position in microns
 - double [BeamScopeSetM2StartPosition](#) (double StartPositionInMicons)

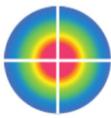


Sets the M² start position

- boolean [BeamScopeSetM2StoptPosition](#) (short StopPositionInMicrons)
Sets the M² stop position in microns
- boolean [BeamScopeM2HomeStage](#) ()
Sets the stage to home position
- double [GetWinCamGain](#) (long WhichCamera)
Returns the gain setting for given camera
- void [SetBackGroundSubtraction2](#) (short New_Remove, short Silent)
This sets the values for two background subtraction settings
- void [RestartMotor](#) ()
Restarts motor
- long [GetCameraIndex](#) ()
Returns the index of current camera
- long [GetBeamScopeHeadType](#) ()
Returns BeamScope head type as defined by number in list
- void [NudgeCrosshairs](#) (long Axis_X_Y, long SignedDirection)
Nudge the crosshairs by one
- boolean [EnableInclusion](#) (long Enable_Yes_No)
Enable inclusion
- short [GetVerticalPixels](#) ()
Returns vertical pixels
- long [CameraType](#) ()
Returns current camera type; see list
- VARIANT [GetWinCamDataAsVariant](#) ()
Returns WinCam data as a variant
- double [GetVSKOffset](#) ()
Returns VSK offset
- void [SetVSKOffset](#) (double newValue)
Sets VSK offset value
- VARIANT [GetTargetWinCamDataAsVariant](#) (short targetCamera)
Returns target WinCam data as a variant
- short [GetCameraImageIndex](#) (short targetCamera)
Gets the image index of designated camera
- short [GetBeamScopeGain](#) (short profileID)
Returns the BeamScope's gain for a given profile as defined by its ID
- bool [SetBeamScopeGain](#) (short newGain, short profileID)
Sets the BeamScope's gain for a given profile as defined by its ID
- long [FillVariantWithWinCamData](#) ([out]VARIANT *var)
Fills given pointer to variant data
- void [SetTargetCameraExposure](#) (long WhichCamera, double newValue)
Sets target camera exposure to value

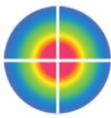
Public Attributes

- long [IsMSquaredOpen](#)
Gets and sets the value for the M2 dialog; 1 opens it and 0 closes it
- short [Palette](#)

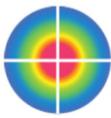


Getter and setter for palette selection; assigning a short sets it to one of the color palettes and accessing this property returns the current palette

- [BSTR PrintNotes](#)
Gets and sets note values as strings; assigning an empty string to it opens the note dialog and accessing this property returns stored notes
- short [InkSaverState](#)
Gets and sets the value for an ink saving option which removes black background from image to save ink; 1 is true and 0 is false
- short [JitterSuppression](#)
Gets and sets the value for jitter suppression; 1 is true and 0 is false
- long [SlitsUsed](#)
Gets and sets the long value for slits used
- double [Wavelength](#)
Gets and sets the double value for wavelength
- short [IsDivergenceOpen](#)
Gets and sets the value for open divergence; 1 is true and 0 is false
- short [WinCamFilter](#)
Gets and sets the short value for WinCam filter; 1 sets it to 1 pixel; 2 sets it to 3 pixels; 3, 5 pixels; 4, 7 pixels; 5, 9 pixels
- short [FastUpdate](#)
Toggles the fast update setting; 1 is true and 0 is false
- short [CameraSelect](#)
Gets and sets (switches) the camera; 0 is the first camera
- short [CurrentCamera](#)
Gets and sets (switches) the camera; 0 is the first camera. Unlike CameraSelect, this does not turn off the camera, so it is highly recommended to use [CameraSelect](#) to switch cameras instead
- short [AutoSnap](#)
Gets and sets AutoSnap; it should be set between 0 and 3
- short [SizeToggle](#)
Gets and sets the value for size toggle; 1 is true and 0 is false
- short [BaselineLocked](#)
Gets and sets the value for locked baseline; 1 is true and 0 is false
- short [WinCamNormalized](#)
Gets and sets the value for WinCam image normalization; 1 is true and 0 is false
- short [BackGroundSubtraction](#)
This gets and sets the background subtraction
- short [AutoNaming](#)
Gets and sets the value for automatically naming files; 1 is true and 0 is false
- double [CentroidClipLevel](#)
Gets and sets the centroid clip level as a percentage in decimal notation; this must be between 0 and 1.0
- double [GeoClipLevel](#)
Gets and sets the geometric centroid clip level as a percentage in decimal notation; this must be between 0 and 1.0
- long [EffectiveCentroidFilterInPixels](#)
Gets and sets the centroid filter size in pixels; the default is 5
- boolean [eTrapOn](#)
Gets and sets the value for eTrap/summary>



- boolean [AutoShutterOn](#)
Gets and sets the value for AutoShutter/summary>
- boolean [UseISO11146](#)
Make measurements based on ISO 11146/summary>
- boolean [RangeLock](#)
Gets and sets the value for RangeLock/summary>
- boolean [LockAll](#)
Gets and sets the value for LockAll/summary>
- boolean [StopMotorAtExit](#)
Gets and sets the value for StopMotorAtExit/summary>
- boolean [UseEffectiveSlits](#)
Gets and sets the value for using effective slits/summary>
- boolean [ShowEffectiveSlits](#)
Gets and sets the value for showing effective slits/summary>
- boolean [AtAim](#)
Gets and sets the value for at aim
- long [BeamMapCdefaultXc](#)
For BeamMap cameras, this gets and sets the default plane; setting a value opens a dialog box/summary>
- long [WinCamDDivergenceCameras](#)
Gets and sets the WinCamD divergent cameras; this must be between 1 and 3; a value outside of this range results in a setting of 3
- short [CentroidType](#)
Gets and sets the centroid type value corresponding to the listed centroid methods; this must be between 0 and 2; a value outside of this range results in a setting of 0
- short [UseAllUsbCameras](#)
Gets and sets the value for using multiple cameras; 1 is true and 0 is false
- long [AlternateDetector](#)
Gets and sets the value for using an alternate detector; 1 is true and 0 is false
- boolean [UseD63](#)
Gets and sets the value for using D63 method of calculating beam diameter
- double [ImagerGain](#)
Gets and sets the gain of the imager; this must be between 1 and 16; a value outside of this range results in the closest in range setting
- short [MajorMinorMethod](#)
Gets and sets the major minor method of the camera as number defined by list; must be range from 0 to 2
- boolean [TriggerEnabled](#)
Gets and sets the value for using the trigger/summary>
- boolean [WinCamDAutoTrigger](#)
Gets and sets the value for using the autotrigger for WinCAMD/summary>
- boolean [TriggerIsInput](#)
Gets and sets the value for using trigger is input/summary>
- boolean [TriggerOnPositive](#)
Gets and sets the value for trigger on positive feature/summary>
- double [AutoTrigMax](#)
Gets and sets the maximum for autotrigger in .1 second increments; this must be between 0.0 and 100; if maximum is bigger than minimum, max = 1.0 and min = 0.1



- double [AutoTrigMin](#)
Gets and sets the minimum for autotrigger in .1 second increments; this must be between 0 and 100. Must be less than maximum; see [AutoTrigMax](#) results in the closest in range setting
- boolean [EnableMultiBeams](#)
Multiple beams enabled
- boolean [EnableCTE](#)
Toggles Comet Tail Elimination and turns off HyperCal
- double [DoubleAlconRo](#)
Sets the radius in microns

Properties

- double [FilterValue](#)
Gets and sets double value for filter between 0 and 10.1

Detailed Description

Primary dispatch interface for GetData ActiveX control

Member Function Documentation

boolean BeamScopeM2HomeStage ()

Sets the stage to home position

Returns:

Returns true upon success

boolean BeamScopeScanControl (long *NumberOfScans*, long *FirstSlitON*, long *SecondSlitOn*)

Sets values for BeamScopeScan and returns true

Returns:

True upon success

double BeamScopeSetM2StartPosition (double *StartPositionInMicons*)

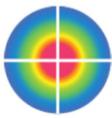
Sets the M² start position

Returns:

Returns the input value upon success

boolean BeamScopeSetM2StoptPosition (short *StopPositionInMicrons*)

Sets the M² stop position in microns



Returns:

Returns true upon success

double BeamScopeSetStagePosition (double *NewPositionInMicrons*)

Sets the stage position in microns

Returns:

Returns the input value upon success

long CameraType ()

Returns current camera type; see list

- 1: UCM_L
- 2: uHs_L
- 3: uHs_s
- 4: uHr_s
- 5: ucm_s
- 6: uccd_12
- 7: uccd_23
- 11: uBeamScope
- 19: uBeamScope9
- 12: uFir_1
- 14: xHr_s
- 22: uFir_2
- 13: uBlade
- 23: uHr_mini
- 26: uHr_mini2
- 24: uHs_mini
- 25: ucm_mini
- 28: uccd_15
- 30: xHr_mini
- 34: LCM_V1

short CloseDialog (short *IndexToDialog*)

Closes dialog defined by number; see list

- 7: WinCam fluence dialog
- 19: WinCam logging dialog
- 10: Logging dialog
- 2: Beamscope M2 dialog
- 15: Wander dialog

boolean DeviceRunning ()

Returns device running status



Returns:

True upon success

boolean DoSearch ()

Searches for device and returns True if it is a BeamScope

Returns:

True upon success

boolean EnableInclusion (long *Enable_Yes_No*)

Enable inclusion

Parameters:

<i>Enable_Yes_No</i>	1 means enable and 0 means disable
----------------------	------------------------------------

long EnableUseEffectiveSlits (long *Enable*)

Enables use of effective slits

Parameters:

<i>Enable</i>	Should be 1 for True (enable) and 0 for False (disable)
---------------	---

double Exposure (long *WhichCamera*)

Returns exposure for given camera

Parameters:

<i>WhichCamera</i>	Camera by index from 0 to 7
--------------------	-----------------------------

void Exposure (long *WhichCamera*, double *newValue*)

Sets exposure for given camera

Parameters:

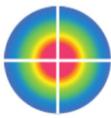
<i>WhichCamera</i>	Camera by index from 0 to 7
<i>newValue</i>	Sets the new exposure value

boolean FingerToPosition (double *Position*)

Sets the aim to the position given as a double

Returns:

True upon success



long GetBeamScopeHeadType ()

Returns BeamScope head type as defined by number in list

- Single slit
- Dual slit
- Dual single list
- Pin-hole slit

short GetCameraImageIndex (short *targetCamera*)

Gets the image index of designated camera

Parameters:

<i>targetCamera</i>	0 is the first
---------------------	----------------

long GetCameraType ()

If WinCam, returns type of camera as defined by number in list, -1 otherwise; for full functionality, use [CameraType](#)

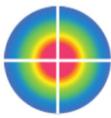
- 1: UCM_L
- 2: uHs_L
- 3: uHs_s
- 4: uHr_s
- 5: ucm_s
- 6: uccd_12
- 7: uccd_23
- 11: uBeamScope
- 19: uBeamScope9
- 12: uFir_1
- 14: xHr_s
- 22: uFir_2
- 13: uBlade
- 23: uHr_mini
- 26: uHr_mini2
- 24: uHs_mini
- 25: ucm_mini
- 28: uccd_15
- 30: xHr_mini
- 34: LCM_V1

double GetClipLevel (short *ClipOneOrTwo_0_1*)

Returns the current level for the given clip

Parameters:

<i>ClipOneOrTwo_0_1</i>	1 (A) or 2 (B)
-------------------------	----------------



short GetClipLevelMode (short *ClipOneOrTwo_0_1*)

Returns the mode for the given clip

Parameters:

<i>ClipOneOrTwo_0_1</i>	1 (A) or 2 (B)
-------------------------	----------------

short GetCurrentDevice ()

Returns current device as number; see table

- 1: BeamScope
- 2: BeamR
- 3: BeamMap
- 4: BeamMC
- 5: WinCam
- 6: WinCam Div
- 7: WinCam Log
- 8: TwoD Scan
- 9: WinCam Comp
- 10: WinCam Comp3
- 11: WinCam Comp4
- 12: WinCam Comp5

boolean GetCurrentWinCamData (long * *ImageDataPt*, long * *XSizePt*, long * *YSizePtr*)

Upon successfully setting pointers to variables, returns true

Parameters:

<i>ImageDataPt</i>	Pointer to be set to image data
<i>XSizePt</i>	Pointer to be set to horizontal size
<i>YSizePtr</i>	Pointer to be set to vertical size

double GetEffectiveCentroidX (long *WhichCamera*)

Returns vertical position of centroid for given camera

Parameters:

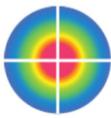
<i>WhichCamera</i>	Camera by index from 0 to 7
--------------------	-----------------------------

double GetEffectiveCentroidY (long *WhichCamera*)

Returns horizontal position of centroid for given camera

Parameters:

<i>WhichCamera</i>	Camera by index from 0 to 7
--------------------	-----------------------------



double GetEffectiveGeoCenterX (long WhichCamera)

Returns vertical position of geometric centroid for given camera

Parameters:

<i>WhichCamera</i>	Camera by index from 0 to 7
--------------------	-----------------------------

double GetEffectiveGeoCenterY (long WhichCamera)

Returns horizontal position of geometric centroid for given camera

Parameters:

<i>WhichCamera</i>	Camera by index from 0 to 7
--------------------	-----------------------------

double GetIncludedPowerPercentAtRadius (double RadiusInMicrons)

Gets the percentage of total power included at a given radius from the centroid

Parameters:

<i>RadiusInMicrons</i>	The radius from the centroid measured in microns
------------------------	--

long GetPixel (long x, long y)

Returns pixel value for (x,y) coordinate position

Parameters:

<i>x</i>	Must be smaller than image width
<i>y</i>	Must be smaller than image height

double GetRadiusAtPowerPercent (double PowerPercent)

For a given percentage, returns the radius from centroid including that percentage of total power

Parameters:

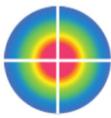
<i>PowerPercent</i>	A percentage 0 to 100 as a double
---------------------	-----------------------------------

boolean GetROI (long * LeftAsLongPointer, long * TopAsLongPointer, long * WidthAsLongPointer, long * HeightAsLongPointer)

Fills given pointers with capture size and starting position

Parameters:

<i>LeftAsLongPointer</i>	Long pointer to be set to horizontal position for capture
<i>TopAsLongPointer</i>	Long pointer to be set to starting vertical position for capture
<i>WidthAsLongPointer</i>	Long pointer to be set to capture width
<i>er</i>	



<i>HeightAsLongPointer</i>	Long pointer to be set to capture height
----------------------------	--

Returns:

Returns true upon success

VARIANT GetTargetWinCamDataAsVariant (short *targetCamera*)

Returns target WinCam data as a variant

Parameters:

<i>targetCamera</i>	The index of the camera from 0 to 7
---------------------	-------------------------------------

Returns:

If data is ready, the size of the variant will match the dimensions; otherwise, it will be a size of 1

VARIANT GetWinCamDataAsVariant ()

Returns WinCam data as a variant

Returns:

If data is ready, the size of the variant will match the dimensions; otherwise, it will be a size of 1

double GetWinCamDPixelSize (short *X_0_Y_1*)

Returns horizontal or vertical pixel size

Parameters:

<i>X_0_Y_1</i>	0 for horizontal and 1 for vertical
----------------	-------------------------------------

double GetWinCamGain (long *WhichCamera*)

Returns the gain setting for given camera

Parameters:

<i>WhichCamera</i>	The index of the camera from 0 to 7
--------------------	-------------------------------------

boolean IsCameraThere (short *WhichCamera_0_1*)

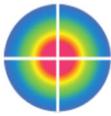
Returns true if camera is there

Parameters:

<i>WhichCamera_0_1</i>	The index of the camera from 0 to 7
------------------------	-------------------------------------

boolean IsDataReady (short *Index*)

Returns if data is ready



Parameters:

<i>Index</i>	Camera by index from 0 to 7
--------------	-----------------------------

void KeyEvent (short *KeyCode*, short *KeyCount*)

Relays events by keyboard input

Parameters:

<i>KeyCode</i>	The keyboard input as short
<i>KeyCount</i>	Deprecated

boolean LoadThisJobFile (BSTR *JobFilePath*)

Loads a job file

Parameters:

<i>JobFilePath</i>	The filename with its path
--------------------	----------------------------

boolean NextProfile ()

Moves the image and profiles forward by one frame

Returns:

True upon success

void NudgeCrosshairs (long *Axis_X_Y*, long *SignedDirection*)

Nudge the crosshairs by one

Parameters:

<i>Axis_X_Y</i>	0 means X and 1 means Y
<i>SignedDirection</i>	Positive means right or up; negative means down or left

boolean OpenClipLevelDlg (short *ClipOneOrTwo_0_1*)

Opens the clip level dialog for the given clip

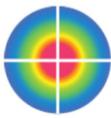
Parameters:

<i>ClipOneOrTwo_0_1</i>	1 (A) or 2 (B)
-------------------------	----------------

Returns:

True upon success

short OpenDialog (short *IndexToDialog*)



Opens dialog defined by number; see list

- 33: Firmware loading dialog
- 29: File browser dialog
- 28: PCD dialog
- 27: UCM calibration dialog
- 30: Test USB M2 stage dialog
- 32: ISO clip dialog
- 16: Centroid clip dialog
- 22: Geometric centroid clip dialog
- 26: UCM test dialog
- 25: Get e width clip dialog
- 12: Wavelength dialog
- 24: M factor dialog
- 14: PCI Eeprom dialog
- 15: Wander dialog
- 34: Old beam calibration dialog
- 23: Beam calibration dialog
- 31: UCM M2 dialog
- 13: Capture dialog
- 11: BS pulsed dialog
- 10: Logging dialog
- 9: Eeprom data dialog
- 2: M2 beamscope dialog
- 17: WinCam image log setup dialog
- 20: Beam fit dialog
- 7: WinCam fluence dialog
- 18: WinCam image log dialog
- 8: Numeric display dialog
- 21: Trigger display dialog
- 35: Fir hot adjust dialog
- 37: LCM registration dialog
- 36: UMap speed change dialog

boolean OpenFile ()

Opens the open file dialog menu

Returns:

True upon success

long OpenThisFile (BSTR *NameOfFile*)

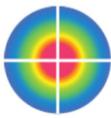
Opens the given file

Parameters:

<i>NameOfFile</i>	The full name of the file
-------------------	---------------------------

Returns:

Success as a boolean



double PersistentData (short *Index*)

Returns persistent data

Parameters:

<i>Index</i>	The index of the persistent data; usually 0 to 60
--------------	---

void PersistentData (short *Index*, double *newValue*)

Sets persistent data to given index

Parameters:

<i>Index</i>	The index of the persistent data; usually 0 to 60
<i>newValue</i>	A value to set to be set

boolean PressButton (long *Button_ID*, int *Left_Button*)

Press button of given ID

Parameters:

<i>Button_ID</i>	Must be between 0 and 440
<i>Left_Button</i>	For the equivalent of a left click; 1 = left click, 0 = right click

Returns:

Returns true upon success

boolean PreviousProfile ()

Moves image and profiles back by one frame

Returns:

True upon success

boolean PutToClipboard (long *ThisAsLong*)

Takes screenshot regardless of input

Returns:

True upon success

short ResetCamera (short *WhichCamera*)

Resets target camera

Parameters:

<i>WhichCamera</i>	The camera's index from 0 to 7
--------------------	--------------------------------



boolean SaveCurrentDataBuffer (BSTR *NameOfFileWithPath*)

Saves the current data buffer into one of the designated file types

Parameters:

<i>NameOfFileWithPath</i>	File extension must match corresponding camera type; see list
---------------------------	---

- BeamScope ".bsf"
- BeamMap ".bmf"
- BeamCamera ".bmc"
- BeamR ".bmr"
- WinCam ".wcf"

boolean SaveFile ()

Opens the save file dialog menu

Returns:

True upon success

boolean SelectProfile ()

Opens the beam selection dialog to select a frame

Returns:

True upon success

long SetAverageNumber (long *NumberToAverage*)

Sets the number you want to average.

Parameters:

<i>NumberToAverage</i>	The number to average
------------------------	-----------------------

void SetBackGroundSubtraction2 (short *New_Remove*, short *Silent*)

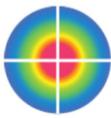
This sets the values for two background subtraction settings

Parameters:

<i>New_Remove</i>	This number should be 0, 1 or 100
<i>Silent</i>	1 is True and 0 is False; 1 prevents the opening of a window

New_Remove=1 and Silent=0 are the default settings of the standalone software and use some background subtraction, but not HyperCal. Setting New_Remove to 100 starts HyperCal.

bool SetBeamScopeGain (short *newGain*, short *profileID*)



Sets the BeamScope's gain for a given profile as defined by its ID

Returns:

Returns true upon success

boolean SetClipLevel (double Clip1, double Clip2, short Mode1, short Mode2)

Sets parameters which affect the profile displays and measurements

Parameters:

<i>Clip1</i>	Sets clip level A as percentage in decimal notation
<i>Clip2</i>	Sets clip level B as percentage in decimal notation
<i>Mode1</i>	Sets clip mode for A
<i>Mode2</i>	Sets clip level for B

As percentages in decimal notation, clip levels should be between 0 and 1. These impact the measurements displayed in the buttons above the profiles in the standalone program. Mode refers to whether you are using the clip level method (Mode=0), or the 4-sigma method (Mode=1). If Mode = 1, then the clip levels don't matter.

Returns:

True upon success

boolean SetControlState (short WhichControl, short State_0_NOT0)

Sets the state for a subset of controls

Parameters:

<i>WhichControl</i>	Must be one of the listed values (5-11)
<i>State_0_NOT0</i>	0 is false and 1 is true

- 5: Auto 3D update
- 6: Auto 2D update
- 7: Jitter control
- 8: Palette change
- 9: Ink saving
- 10: Auto naming
- 11: Live recall

Returns:

True upon success

short SetCurrentDevice (short DeviceType)

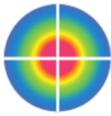
Set current device

Parameters:

<i>DeviceType</i>	Must be a number from 1 to 12; see list
-------------------	---

Returns:

Submitted number



- 1: BeamScope
- 2: BeamR
- 3: BeamMap
- 4: BeamMC
- 5: WinCam
- 6: WinCam Div
- 7: WinCam Log
- 8: TwoD Scan
- 9: WinCam Comp
- 10: WinCam Comp3
- 11: WinCam Comp4
- 12: WinCam Comp5

long SetDefaultXcPlane (long *DefaultXcPlane*)

For BeamMap, sets the default X plane

Parameters:

<i>DefaultXcPlane</i>	Must be in the range from 0 to 3
-----------------------	----------------------------------

boolean SetDisplayMode (short *DisplayMode*)

Sets display mode in microns

Parameters:

<i>DisplayMode</i>	Should be between 0 and 5
--------------------	---------------------------

Returns:

True upon success

void SetGamma (double *NewGamma*)

Sets gamma value

Parameters:

<i>NewGamma</i>	Must be between 0.2 and 5.0
-----------------	-----------------------------

void SetHeadAngle (double *AngleInRadians*)

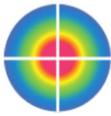
Set head angle in radians

Parameters:

<i>AngleInRadians</i>	The angle in radians
-----------------------	----------------------

void SetHeadRotationAsFixed (long *HeadRotationFixed_0_1*)

Make head rotation fixed



Parameters:

<i>HeadRotationFixed_0_1</i>	0 to disable and 1 to enable
------------------------------	------------------------------

boolean SetLiveRecallState (short *NewState_0_IS_LIVE*)

Toggle between live and recall state

Parameters:

<i>NewState_0_IS_LIVE</i>	0 is live and 1 is recall
---------------------------	---------------------------

Returns:

True upon success

long SetNonuniformityOnOff (long *NonZeroIsOn*)

Toggle non-uniform;

Parameters:

<i>NonZeroIsOn</i>	1 enables and 0 disables
--------------------	--------------------------

long SetRealTimeLogging (long *EnabledIsNotZero*)

Enable real time logging

Parameters:

<i>EnabledIsNotZero</i>	1 enables and 0 disables
-------------------------	--------------------------

boolean SetROI (long *Left*, long *Top*, long *Width*, long *Height*)

Sets the capture size and starting positions

Parameters:

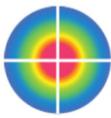
<i>Left</i>	The starting horizontal position for capture
<i>Top</i>	The starting vertical position for capture
<i>Width</i>	The capture width
<i>Height</i>	The capture height

void SetTargetCameraExposure (long *WhichCamera*, double *newValue*)

Sets target camera exposure to value

Parameters:

<i>WhichCamera</i>	The index of the camera from 0 to 7
<i>newValue</i>	Must be valid exposure setting between 0 to 1000



void SetVSKOffset (double *newValue*)

Sets VSK offset value

Parameters:

<i>newValue</i>	Usually between -1.75 and 1.75
-----------------	--------------------------------

boolean SetWorkingDirectory (BSTR *WorkingDirectory*)

Sets working directory for placement of DataRay files

Returns:

Returns true upon success

boolean StartBeamScopeM2 ()

Starts BeamScope M²; will open dialog menu

Returns:

Returns true upon success

boolean StartDevice ()

Returns true on successful start of device

Returns:

True upon success

boolean StartDriver ()

Starts the driver

Returns:

True upon success

boolean StopBeamScopeM2 ()

Stops M²

Returns:

Returns true upon success

boolean StopDevice ()

Returns true on successful stop of device



Returns:

True upon success

short ToggleDialog (short *IndexOfDialog*)

Toggles dialog defined by number; see list

- 34: Old beam calibration dialog
- 22: Beam calibration dialog
- 35: Fir hot adjust dialog
- 37: LCM registration dialog
- 38: OpenGL Test dialog
- 7: WinCam fluence dialog
- 15: Wander dialog
- 10: Logging dialog
- 19: WinCam logging dialog
- 20: Beam fit dialog

Member Data Documentation

short AutoNaming

Gets and sets the value for automatically naming files; 1 is true and 0 is false

Correct

short AutoSnap

Gets and sets AutoSnap; it should be set between 0 and 3

- 0: snap to centroid
- 1: snap to center
- 2: snap to peak
- 3: snap to user defined point

short BackGroundSubtraction

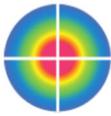
This gets and sets the background subtraction

It works the same as [SetBackGroundSubtraction2](#), except Silent is set to 1 (True). The default of the standalone software is 1 and it uses some background subtraction, but not HyperCal. Setting this to 100 starts HyperCal.

short CentroidType

Gets and sets the centroid type value corresponding to the listed centroid methods; this must be between 0 and 2; a value outside of this range results in a setting of 0

- 0: Centroid Method 0: $X_c = \frac{\sum[x.I(x,y)]}{\sum[I(x,y)]}$
- 1: Centroid Method 1: $X_c = \frac{\sum[x.I(x,y)^2]}{\sum[I(x,y)^2]}$



- 2: Centroid Method 2: $X_c = \frac{\sum[x \cdot I(x,y)^3]}{\sum[I(x,y)^3]}$

short MajorMinorMethod

Gets and sets the major minor method of the camera as number defined by list; must be range from 0 to 2

- 0: default major minor method
- 1: ISO 11146
- 2: D63

short Palette

Getter and setter for palette selection; assigning a short sets it to one of the color palettes and accessing this property returns the current palette

- 1: high color
- 2: monochrome (greyscale)
- 3: 32 colors
- 4: 10 colors

GetDataEvents Interface Reference

Event dispatch interface for GetData ActiveX control

Public Member Functions

- void [SendMessage](#) (long Message, long LongValue, double DoubleValue)
Event fired every time a message is sent
- void [DataReady](#) ()
Event fired every time new data becomes ready

Detailed Description

Event dispatch interface for GetData ActiveX control

Member Function Documentation

void SendMessage (long Message, long LongValue, double DoubleValue)

Event fired every time a message is sent



Parameters:

<i>Message</i>	The message as defined by a number
<i>LongValue</i>	Occasionally has value
<i>DoubleValue</i>	Occasionally has value

PaletteBar Interface Reference

Dispatch interface for PaletteBar ActiveX control

Detailed Description

Dispatch interface for PaletteBar ActiveX control

Profiles Interface Reference

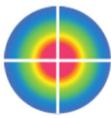
Dispatch interface for Profile ActiveX controls

Public Member Functions

- boolean [GetProfileData](#) (long *LongBuffer_32bit, long NumberOfLongs)
Upon successfully filling buffer, returns true
- boolean [PutImagetoClipboard](#) ()
Upon putting an image to clipboard, it returns true
- boolean [SaveImagetoFile](#) (BSTR FileNameWithPath)
Upon saving an image to the file, it returns true
- boolean [EnableTopHat](#) (void)
Upon enabling TopHat view of given profile, it returns true
- boolean [EnableGFit](#) (void)
Upon enabling Gaussian fit view of given profile, it returns true
- VARIANT [GetProfileDataAsVariant](#) (void)
Gets the profile data as a variant
- double [GetStepSize](#) (void)
Gets the step size
- int [GetBaseline](#) (void)
Gets the baseline

Public Attributes

- short [ProfileID](#)
Sets the profile type by ID number



Properties

- short [MyID](#)
MyID is the same as ProfileID; should use ProfileID

Detailed Description

Dispatch interface for Profile ActiveX controls

Member Function Documentation

boolean EnableGFit (void)

Upon enabling Gaussian fit view of given profile, it returns true

Either TopHat fit or Gaussian fit can be enabled; both cannot be enabled at the same time

boolean EnableTopHat (void)

Upon enabling TopHat view of given profile, it returns true

Either TopHat fit or Gaussian fit can be enabled; both cannot be enabled at the same time

int GetBaseline (void)

Gets the baseline

Returns:

An int representing the baseline; 0 if not live.

boolean GetProfileData (long * LongBuffer_32bit, long NumberOfLongs)

Upon successfully filling buffer, returns true

Parameters:

<i>LongBuffer_32bit</i>	The buffer of long
<i>NumberOfLongs</i>	The size of the buffer

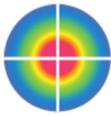
VARIANT GetProfileDataAsVariant (void)

Gets the profile data as a variant

Returns:

Variant data is 1D array of 4-byte integers of length 2048.

double GetStepSize (void)



Gets the step size

Returns:

A double representing the step size.

boolean SaveImageToFile (BSTR FileNameWithPath)

Upon saving an image to the file, it returns true

Parameters:

FileNameWithPath	The path and name of file to be saved
------------------	---------------------------------------

ShutterControl Interface Reference

Dispatch interface for ShutterControl ActiveX controls

Public Member Functions

- boolean [SetID](#) (short ScrollID)
Sets the type of shutter as defined by its ID in list
- void [AboutBox](#) ()
Displays an about box

Detailed Description

Dispatch interface for ShutterControl ActiveX controls

Member Function Documentation

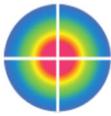
boolean SetID (short ScrollID)

Sets the type of shutter as defined by its ID in list

Parameters:

ScrollID	Must range from 0 to 2
----------	------------------------

- 0: Trigger
- 1: Gain
- 2: Shutter



ThreeDview Interface Reference

Dispatch interface for ThreeDView ActiveX control

Public Member Functions

- boolean [PutImagetoClipboard](#) ()
Puts image to clipboard
- boolean [SaveImagetoFile](#) (BSTR FileNameWithPath)
Saves image to designated file

Detailed Description

Dispatch interface for ThreeDView ActiveX control

Member Function Documentation

boolean SaveImagetoFile (BSTR *FileNameWithPath*)

Saves image to designated file

Parameters:

<i>FileNameWithPath</i>	The filename and path combined
-------------------------	--------------------------------

TriggerControl Interface Reference

Dispatch interface for Trigger ActiveX control

Public Member Functions

- void [AboutBox](#) ()
Displays an about box

Detailed Description

Dispatch interface for Trigger ActiveX control



TwoD Interface Reference

Dispatch interface for TwoD ActiveX control

Public Member Functions

- boolean [PutImagetoClipboard](#) ()
Puts image to clipboard
- boolean [SaveImagetoFile](#) (BSTR FileNameWithPath)
Saves image to designated file

Detailed Description

Dispatch interface for TwoD ActiveX control

Member Function Documentation

boolean [SaveImagetoFile](#) (BSTR *FileNameWithPath*)

Saves image to designated file

Parameters:

<i>FileNameWithPath</i>	The filename and path combined
-------------------------	--------------------------------