ICamera2 and ICamDiscover Reference Documentation

September 22, 2003
Version 0.9

1 ICamera2 Methods

1.1 Init

1.1.1 Format:

```
Init( [in] Apn_Interface Interface,
      [in] long CamIdOne,
      [in] long CamIdTwo,
      [in] long Option )
```

1.1.2 Parameters:

Interface: The interface type requested by the application. Valid values are *Apn_Interface_NET*, for Ethernet cameras, and *Apn_Interface_USB*, for USB 2.0 camera systems.

CamIdOne: The first of three camera identifiers. For camera systems using *Apn_Interface_NET*, this identifier is the camera IP address. The IP address is written in standard little endian byte order, so an address of 192.168.0.3 has the value 0xC0A80003. For camera systems using the *Apn_Interface_USB*, this identifier is used to identifying a particular camera, as enumerated by the operating system.

CamIdTwo: The second of three camera identifiers. For camera systems using the *Apn_Interface_NET*, this identifier is the IP address port number of the camera. For camera systems using the *Apn_Interface_USB*, this identifier is not used and should be set to zero (0x0).

Option: Reserved for future use. In the future, this parameter may be used for passing interface-specific information to the driver during Initialization. Should be set to zero (0x0).

1.1.3 Description:

The *Init()* method is used for initializing the APn camera system and loading firmware to the device.

1.2 ResetSystem

1.2.1 Format:

```
ResetSystem()
```

1.2.2 Parameters:

None.

1.2.3 Description:

The *ResetSystem()* method resets the camera's internal pixel processing engines, and then starts the system flushing again.

This method may be used by an application to attempt to clear an error condition from the device, instead of re-initializing the complete system. This method is not destructive. Programmed camera settings will remain intact after the method is called. An application using *ResetSystem()* as an attempt to clear an error condition should query status after this method is called to check the current state of the camera.

The *ResetSystem()* method does *not* return the camera to the initial state it was in after the *Init()* method was called. Applications wishing to re-initialize the camera system to known state should call the *Init()* method.

1.3 Expose

1.3.1 Format:

```
Expose( [in] double Duration,
        [in] Boolean Light )
```

1.3.2 Parameters:

Duration: Length of the exposure(s), in seconds. The valid range for this parameter is 0.00000256s to 10994.4s

Light: Determines whether the exposure is a light or dark/bias frame. A light frame requires this parameter to be set to "TRUE", while a dark frame requires this parameter to be "FALSE".

1.3.3 Description:

The *Expose()* method begins the imaging process. The following types of imaging categories are begun with this method:

1) Light (Nominal) Frames
2) Dark and Bias Frames
3) TDI Images

4) Image Sequences
5) Triggered Images

The type of exposure taken is dependent on various state variables, which are properties of the ICamera2 interface—these include *TdiMode* and *TriggerMode*.

1.4 PauseTimer

1.4.1 Format:

```
PauseTimer( Boolean PauseState )
```

1.4.2 Parameters:

PauseState: A state variable that controls the pausing of the exposure timer. A value of "TRUE" will issue a command to pause the timer. A value of "FALSE" will issue a command to unpause the timer. Multiple calls with this parameter set consistently to either state (i.e. back-to-back "TRUE" states) have no effect.

1.4.3 Description:

The *PauseTimer()* method pauses the current exposure by closing the shutter and pausing the exposure timer.

There is no limit to the length of time that the exposure timer may be paused.

1.5 StopExposure

1.5.1 Format:

```
StopExposure( Boolean Digitize )
```

1.5.2 Parameters:

Digitize: A state variable that controls whether the stopped exposure data will be digitized or discarded by the application. A value of "TRUE" indicates that the application wishes to download the data in the future. A value of "FALSE" indicates the application will not try to retrieve the image data.

1.5.3 Description:

The *StopExposure()* method halts/stops an exposure already in progress, and the hardware begins digitizing the image.

An application should follow a call to the *StopExposure()* method with a call to retrieve the image data (i.e. using *GetImage()*). The application then has the option of discarding the image data entirely, or displaying the data from the shortened exposure.

If *StopExposure()* is called, and there is no exposure in progress, the method has no effect.

1.6 GetImage

1.6.1 Format:

```
GetImage(long pImageBuffer)
```

1.6.2 Parameters:

pImageBuffer: Returns a pointer to 16 bit, unsigned short data located in memory. The image data region should be allocated by the application prior to calling this method.

1.6.3 Description:

The *GetImage()* method returns a pointer to a previously-allocated region of memory (allocated by the calling application) that will be filled with image data.

The application must take care to assure that it properly allocates the image memory region before calling this method.

1.7 GetLine

1.7.1 Format:

```
GetLine(long pLineBuffer)
```

1.7.2 Parameters:

pLineBuffer: Returns a pointer to 16 bit, unsigned short data located in memory. The image data region should be allocated by the application prior to calling this method.

1.7.3 Description:

The *GetLine()* method returns a pointer to a previously-allocated region of memory that will be filled with line data.

The application must take care to assure that it properly allocates the image memory region before calling this method.

This method should not be used with the *Apn_Interface_NET* interface type. If it is used with this interface, it will fail.

2 ICamera2 Helper-Dialog Methods

The ICamera2 interface also includes three methods to assist application writers in getting their applications up and running as quickly as possible. These methods invoke modal dialog boxes for encapsulating some of the Alta functionality. This allows application writers to concentrate on features specific to their software, instead of creating dialog boxes to display camera features. Of course, many application writers will choose not to use these generic dialog boxes, and any functionality in these dialogs can also be queried through the ICamera2 properties.

2.1 ShowIoDialog

2.1.1 Format:

```
ShowIoDialog()
```

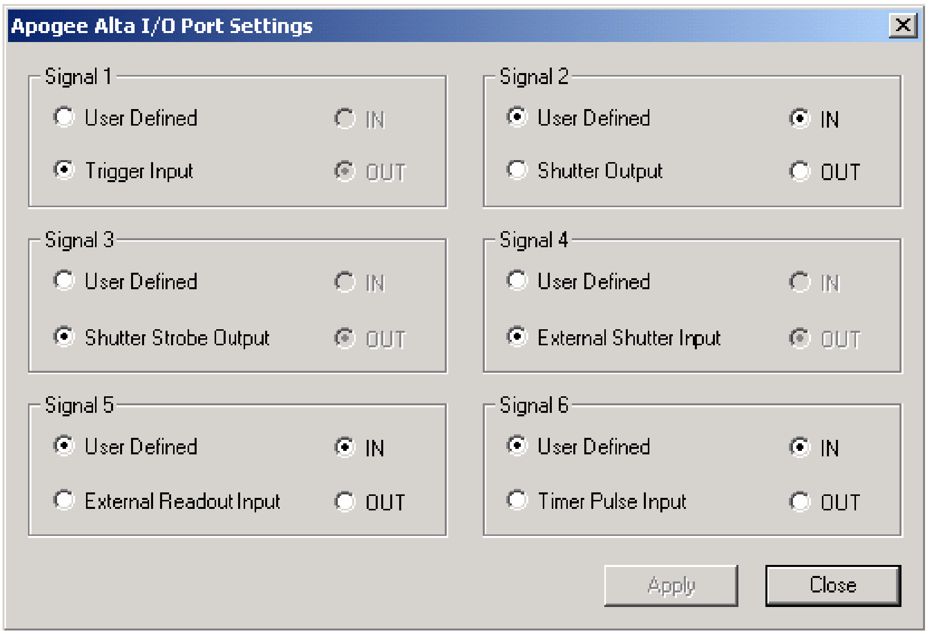
2.1.2 Parameters:

None.

2.1.3 Description:

The *ShowIoDialog()* method can be used to display an I/O selection dialog to the user. The dialog box is a modal dialog. The *ShowIoDialog()* method is not required to access the camera I/O features—please see the various properties relating to camera I/O for that information. This method is intended as a convenience for application writers who do not wish to write their own dialog box to encapsulate this functionality.

The following graphic shows the I/O selection dialog:



Apogee Alta I/O Port Settings

Signal 1: User Defined [✓ Trigger Input] — IN [✓ OUT]

Signal 2: [✓ User Defined] Shutter Output — [✓ IN] OUT

Signal 3: User Defined [✓ Shutter Strobe Output] — IN [✓ OUT]

Signal 4: User Defined [✓ External Shutter Input] — IN [✓ OUT]

Signal 5: [✓ User Defined] External Readout Input — [✓ IN] OUT

Signal 6: [✓ User Defined] Timer Pulse Input — [✓ IN] OUT

2.2 ShowLedDialog

2.2.1 Format:

```
ShowLedDialog()
```

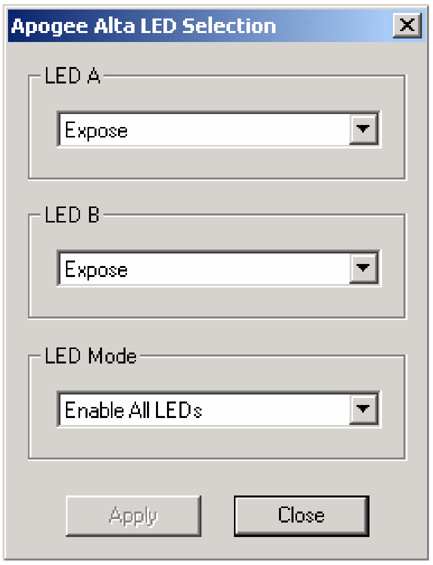
2.2.2 Parameters:

None.

2.2.3 Description:

The *ShowLedDialog()* method can be used to display an LED selection dialog to the user. The dialog box is a modal dialog. The *ShowLedDialog()* method is not required to access the camera LED features—please see the various properties relating to camera LED control for that information. This method is intended as a convenience for application writers who do not wish to write their own dialog box to encapsulate this functionality.

The following graphic shows the LED selection dialog:



2.3 ShowTempDialog

2.3.1 Format:

```
ShowTempDialog()
```

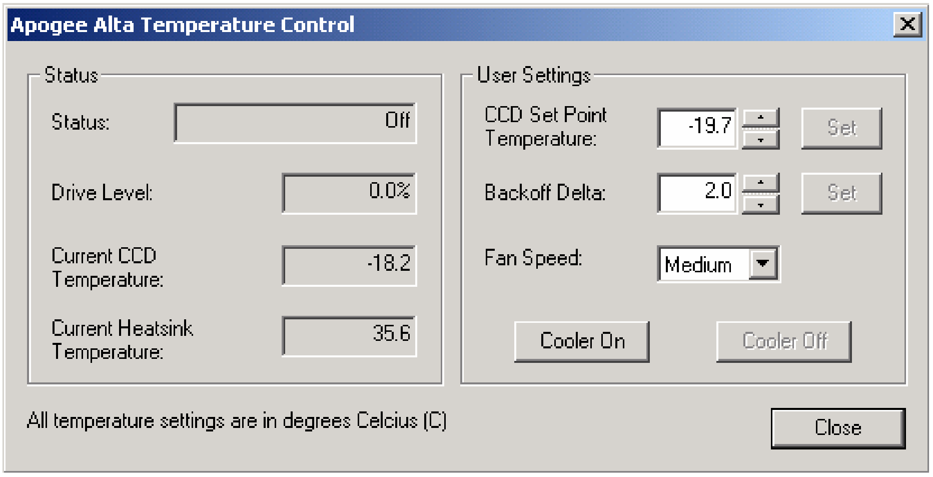
2.3.2 Parameters:

None.

2.3.3 Description:

The *ShowTempDialog()* method can be used to display a temperature control dialog to the user. The dialog box is a modal dialog. The *ShowTempDialog()* method is not required to access the camera temperature control features—please see the various properties relating to camera I/O for that information. This method is intended as a convenience for application writers who do not wish to write their own dialog box to encapsulate this functionality.

The following graphic shows the temperature control dialog:



Apogee Alta Temperature Control

Status

Status: Off

Drive Level: 0.0%

Current CCD Temperature: -18.2

Current Heatsink Temperature: 35.6

User Settings

CCD Set Point Temperature: -19.7

Backoff Delta: 2.0

Fan Speed: Medium

All temperature settings are in degrees Celcius (C)

3 ICamera2 Properties

Camera Settings			
Variable	R/W	Data Type	Notes
AvailableMemory	RO	Long	Returns the amount of available memory for storing images in terms of kilobytes (KB).
CameraInterface	RO	Apn_Interface	Returns the interface type supported by the camera. Valid values are listed below.
			0x0 (Apn_Interface_NET): Ethernet interface.
			0x1 (Apn_Interface_USB): USB 2.0 interface.
CameraMode	R/W	Apn_CameraMode	Returns/Sets the operational mode of the camera. The default value for this variable after initialization is Apn_CameraMode_Normal.
			0x0 (Apn_CameraMode_Normal): Specifies nominal camera operation for exposure control. Single exposures, or sequences of exposures, are completely initiated by software control.
			0x1 (Apn_CameraMode_TDI): Specifies camera operation using time delayed integration (drift scan) mode. Used in conjunction with TDIRows and TDIRate. The actual TDI exposure is started with the Expose method, but the "Duration" parameter of Expose is not used.
			0x2 (Apn_CameraMode_Test): Specifies that the camera operation should be defined using simulated data for image parameters.
			0x3 (Apn_CameraMode_ExternalTrigger): Specifies camera operation using an external trigger to control the exposure.
			0x4 (Apn_CameraMode_ExternalShutter): Specifies camera operation using an external shutter to control the exposure.
CameraModel	RO	String	Returns a camera model identifier for the device.
DataBits	R/W	Apn_Resolution	Digitization Resolution. Valid values are listed below. The default value for this variable after initialization is Apn_Resolution_SixteenBit.
			0x0 (Apn_Resolution_SixteenBit): Selects resolution of 16 bits per pixel.
			0x1 (Apn_Resolution_TwelveBit): Selects resolution of 12 bits per pixel.
DriverVersion	RO	String	Version number of the camera driver. This is the version of the driver stack as released by Apogee Instruments, and not specifically the file version of Apogee.DLL. A return value of <=0 indicates that the driver stack version could not be recognized, and should be treated as an error code by the application. Note that the use of this property does not require a connection to the camera system.
FastSequence	R/W	Boolean	Enables/Disables very fast back to back exposures. Interline CCDs only. (Also referred to as Ratio Mode.) The default value for this variable after initialization is FALSE.
FirmwareVersion	RO	Long	Version number of the camera control firmware.
ImagingStatus	RO	Apn_Status	Returns the current imaging state of the camera. Error conditions are noted by negative numbers. The Apn_Status_Idle is a unique state that the camera should

			never be in once initialization has occurred (the normal camera state is for the camera to be flushing).
			-2: Apn_Status_DataError - Error. An internal error was generated by the camera during image readout and the internal FIFO was hung. Using the Reset() or Init() methods may return the camera to a known, good state.
			-1: Apn_Status_PatternError - Error. An internal error was generated by the camera during pixel processing. Using the Reset() or Init() methods may return the camera to a known, good state.
			0: Apn_Status_Idle - Idle. The camera system is completely idle. Flushing operations have not been started. Applications should typically never see this state after the Init() method has been called.
			1: Apn_Status_Exposing - Exposing. An exposure is in progress.
			2: Apn_Status_ImagingActive - Imaging Active. The camera is reading out an image, or waiting for an image to begin. While an image is actually being exposed, the status returned will be Apn_Status_Exposing.
			3: Apn_Status_ImagingDone - Imaging Complete. The camera has completed an exposure and digitized the image data. Applications should poll this flag before retrieving the image data. The status flag Apn_Status_ImagingDone will be sent only one time after an image is complete, subsequent queries for status will (in the nominal case) return Apn_Status_Flushing.
			4: Apn_Status_Flushing - Flushing. The camera system is flushing the sensor. No other operations are in effect.
			5: Apn_Status_WaitingOnTrigger - Waiting on Trigger. The camera is waiting for a trigger event to start an exposure.
InputVoltage	RO	Double	Returns the operating input voltage to the camera.
MaxExposure	RO	Double	Returns the maximum exposure duration. This is a hard value. Exposure times sent to the "Expose" method, which are greater than MaxExposure, will be truncated to the value specified by MaxExposure.
MinExposure	RO	Double	Returns the *suggested* minimum exposure duration, based on the camera model's sensor type, shutter size, et cetera. As this is a suggested duration, the actual exposure time sent to the "Expose" method may be less than the value specified in MinExposure.
NetworkTransferMode	R/W	Apn_NetworkMode	***This mode is temporarily disabled.*** Used only with Ethernet camera systems. Valid values are listed below. This variable should only be changed when the interface of the camera is Ethernet. Modifying this variable while controlling a USB camera has no effect. The default value of this variable after initialization is Apn_NetworkMode_Tcp. 0x0 (Apn_NetworkMode_Tcp): Selects transfer of the image data via TCP/IP. While the slowest of the transfer types, it can be used over the Internet itself, and is highly reliable. 0x1 (Apn_NetworkMode_Udp): A faster transfer type for use within lightly-loaded local area networks (LANs).
SerialASupport	RO	Boolean	Returns whether the camera supports Serial Port A.

SerialBSupport	RO	Boolean	Returns whether the camera supports Serial Port B.
Shutter Settings			
Variable	R/W	Data Type	Notes
DisableShutter	R/W	Boolean	TRUE forces shutter closed and disabled during an exposure; FALSE allows normal operation. Overridden by the value of ForceShutterOpen. The default value of this variable after initialization is FALSE.
ExternalIoReadout	R/W	Boolean	When TRUE, the readout of the camera is no longer started by the external shutter. Instead, a separate I/O pin is used to start the readout. The default value of this variable after initialization is FALSE.
ForceShutterOpen	R/W	Boolean	TRUE forces shutter to open; FALSE allows normal shutter operation (if shutter was previously opened with this command, FALSE will then close the shutter). This property overrides the DisableShutter property. The default value of this variable after initialization is FALSE.
ShutterAmpControl	R/W	Boolean	TRUE disables the CCD voltage while the shutter strobe is high. The default value of this variable after initialization is FALSE.
ShutterState	RO	Boolean	Returns TRUE if shutter is open; FALSE if closed.
ShutterStrobePeriod	R/W	Double	Sets the period of the shutter strobe appearing on a pin at the experiment interface. The minimum valid value is 45ns and maximum value is 2.6ms. (40ns/bit resolution)
ShutterStrobePosition	R/W	Double	Sets the delay from the time the exposure begins to the time the rising edge of the shutter strobe period appears on a pin at the experiment interface. The minimum valid value is 3.31us and the maximum value is 167ms. (2.56us/bit resolution)
LED Settings			
Variable	R/W	Data Type	Notes
LedMode	R/W	Apn_LedMode	Format of the status LED lights. Must be one of following (Default is Apn_LedMode_EnableAll):
			0x0 (Apn_LedMode_DisableAll): Disable all LEDs
			0x1 (Apn_LedMode_DisableWhileExpose): Disable LEDs during exposure only
			0x2 (Apn_LedMode_EnableAll): Enable LEDs at all times
LedA	R/W	Apn_LedState	Indicates the usage of LED A, which is user-defined by the table below. The default value of this variable after initialization is Apn_LedState_Expose.
			0x0 (Apn_LedState_Expose): Expose
			0x1 (Apn_LedState_ImageActive): Image Active
			0x2 (Apn_LedState_Flushing): Flushing
			0x3 (Apn_LedState_ExtTriggerWaiting): Waiting for external trigger
			0x4 (Apn_LedState_ExtTriggerReceived): External Trigger Received
			0x5 (Apn_LedState_ExtShutterInput): External Shutter Input
			0x6 (Apn_LedState_ExtStartReadout): External Start Readout
			0x7 (Apn_LedState_AtTemp): At Temperature
LedB	R/W	Apn_LedState	Indicates the usage of LED B, as defined by the user. (See

			table for LedA, above.) The default value of this variable after initialization is Apn_LedState_Expose.
TestLedBrightness	R/W	Double	Controls the brightness/intensity level of the test LED light within the cap of the camera head. Expressed as a percentage from 0% to 100%. The default value of this variable after initialization is 0%.
I/O Port Settings			
Variable	R/W	Data Type	Notes
IoPortAssignment	R/W	Long	Defines the signal usage for the I/O port. Valid range is for the 6 LSBs, 0x0 to 0x3F. The default value for this variable after initialization is 0x0.
			Bit 0 (I/O Signal 1): A value of zero (0) indicates that the I/O bit is user defined according to the specified IoPortDirection. A value of one (1) indicates that this I/O will be used as a trigger input.
			Bit 1 (I/O Signal 2): A value of zero (0) indicates that the I/O bit is user defined according to the specified IoPortDirection. A value of one (1) indicates that this I/O will be used as a shutter output.
			Bit 2 (I/O Signal 3): A value of zero (0) indicates that the I/O bit is user defined according to the specified IoPortDirection. A value of one (1) indicates that this I/O will be used as a shutter strobe output.
			Bit 3 (I/O Signal 4): A value of zero (0) indicates that the I/O bit is user defined according to the specified IoPortDirection. A value of one (1) indicates that this I/O will be used as an external shutter input.
			Bit 4 (I/O Signal 5): A value of zero (0) indicates that the I/O bit is user defined according to the specified IoPortDirection. A value of one (1) indicates that this I/O will be used for starting readout via an external signal.
			Bit 5 (I/O Signal 6): A value of zero (0) indicates that the I/O bit is user defined according to the specified IoPortDirection. A value of one (1) indicates that this I/O will be used for an input timer pulse.
IoPortDirection	R/W	Long	Defines I/O port signal selection. Valid range is for the 6 LSBs, 0x0 to 0x3F. This property defines user-selected I/O port definitions. The I/O signals must have been marked specifically as user defined by the IoPortAssignment property.
			Bit 0: I/O Signal 1 (0=IN and 1=OUT)
			Bit 1: I/O Signal 2 (0=IN and 1=OUT)
			Bit 2: I/O Signal 3 (0=IN and 1=OUT)
			Bit 3: I/O Signal 4 (0=IN and 1=OUT)
			Bit 4: I/O Signal 5 (0=IN and 1=OUT)
			Bit 5: I/O Signal 6 (0=IN and 1=OUT)
IoPortData	R/W	Long	Data sent to/from the I/O port. Dependent on the I/O port assignment and direction (IoPortAssignment/IoPortDirection). Applications are responsible for toggling bits, i.e., if Bit 2 of the I/O port is specified as an OUT signal, and a 0x1 is written to this bit, it will remain 0x1 until 0x0 is written to the same bit. Valid range of this property is for the 6 LSBs, 0x0 to 0x3F.
Cooler/Fan Settings			

Variable	R/W	Data Type	Notes
CoolerControl	RO	Boolean	Returns TRUE if the camera supports cooling, FALSE if no cooling is available.
CoolerRegulated	RO	Boolean	Returns TRUE if the camera supports regulated cooling, FALSE if regulated cooling is not available.
CoolerEnable	R/W	Boolean	Returns/Sets the Cooler operation. A value of TRUE will enable Cooler operation, and FALSE will turn the cooler off.
CoolerStatus	RO	Apn_CoolerStatus	Returns the current cooler status
			0x0 (Apn_CoolerStatus_Off): Off (At or near Ambient). No drive applied to the Cooler.
			0x1 (Apn_CoolerStatus_RampingToSetPoint): Ramp to the Set Point specified by the CoolerSetPoint property.
			0x2 (Apn_CoolerStatus_AtSetPoint): At Set Point specified by the CoolerSetPoint property.
			0x3 (Apn_CoolerStatus_Revision): Controller generated temp revision. If the temperature Set Point is revised, the system will continue to return this status code until the next read of the CoolerSetPoint property.
CoolerSetPoint	R/W	Double	Returns/Sets the desired temperature in degrees Celsius. If the Set Point cannot be reached, the Cooler will determine a new Set Point based on the Backoff Point, and change the status to Apn_CoolerStatus_Revision. An application should reread this property to see the new Set Point that the system is using. Once the application rereads this property, the status of Apn_CoolerStatus_Revision will be cleared.
CoolerBackoffPoint	R/W	Double	Returns/Sets the Backoff temperature of the cooler subsystem. The Backoff Point is given in degrees Celsius. If the cooler is unable to reach the Set Point, the Backoff Point is number of degrees up from the lowest point reached. Used to prevent the cooler from being constant driven with max power to an unreachable temperature. The default value of this variable after initialization can vary depending on camera model, but is typically set at 2.0 degrees Celsius.
CoolerDrive	RO	Double	Drive level applied to the temp controller. Expressed as a percentage from 0% to 100%.
FanMode	R/W	Apn_FanMode	Returns/Sets the current fan speed. The default value of this variable after initialization is Apn_FanMode_Medium.
			0x0 (Apn_FanMode_Off): Off
			0x1 (Apn_FanMode_Low): Low
			0x2 (Apn_FanMode_Medium): Medium
			0x3 (Apn_FanMode_High): High
TempCCD	RO	Double	Returns the current CCD temperature in degrees Celsius.
TempHeatsink	RO	Double	Returns the current Heatsink temperature in degrees Celsius.
Geometry Settings			
Variable	R/W	Data Type	Notes
PhysicalColumns, PhysicalRows	RO	Long	Returns the total number of physical columns or rows on the CCD. These variables depend upon the particular geometry of the sensor used within the camera.
ImagingColumns, ImagingRows	RO	Long	Returns the imaging area size in terms of unbinned pixels. These variables depend upon the particular geometry of the sensor used within the camera.
OverscanColumns	RO	Long	Returns the number of overscan columns in terms of unbinned pixels. This variable depends upon the particular

			sensor used within the camera.
DigitizeOverscan	R/W	Boolean	Determines whether the overscan region will ignored or digitized. Only valid when RoiBinningH is set to 1. The default value for this variable after initialization is FALSE.
RoiPixelsH, RoiPixelsV	R/W	Long	Returns/Sets the image/subframe size in terms of binned pixels. The variables are indexed from one (1). When DigitizeOverscan is FALSE, the valid range for RoiPixelsH is from 1 to ImagingColumns, and when DigitizeOverscan is TRUE, the valid range for RoiPixelsH is from 1 to ImagingColumns+OverscanColumns. The valid range for RoiPixelsV is from 1 to ImagingRows. The default value of RoiPixelsH after initialization is ImagingColumns, and the default value of RoiPixelsV after initialization is ImagingRows.
RoiStartX, RoiStartY	R/W	Long	Returns/Sets the subframe start position in terms of unbinned pixels. The variables are indexed from zero (0). When DigitizeOverscan is FALSE, the valid range for StartX is from 0 to ImagingColumns-1, and when DigitizeOverscan is TRUE, the valid range for StartX is from 0 to ImagingColumns+OverscanColumns-1. The valid range for StartY is from 0 to ImagingRows-1. The default value of both variables after initialization is 0.
Binning Parameters			
Variable	R/W	Data Type	Notes
MaxBinningH, MaxBinningV	RO	Long	Returns the maximum horizontal and vertical binning factors of the device.
RoiBinningH, RoiBinningV	R/W	Long	Returns/Sets the horizontal and vertical binning parameters for an exposure. The valid range for these properties is between 1 and the corresponding value of MaxBinningH (for RoiBinningH) or MaxBinningV (for RoiBinningV). The default value for both variables after initialization is 1.
TDI Parameters			
Variable	R/W	Data Type	Notes
TDICounter	RO	Long	Dynamically incrementing count during a TDI image. The final value of TDICounter equals TDIRows. Valid range is between 1 and 65535.
TDIRate	R/W	Double	Incremental rate between TDI rows. Range is from 5.12us to 336ms.
TDIRows	R/W	Long	Total number of rows in the TDI image. Range is between 1 and 65535.
Sequence Parameters			
Variable	R/W	Data Type	Notes
ImageCount	R/W	Long	Number of images in an image sequence. For single exposures, this property is simply set to 1. Valid range is between 1 and 65535. The default value of this variable after initialization is 1.
SequenceCounter	RO	Long	Dynamically incrementing count during an image sequence. The final value of SequenceCounter equals ImageCount. Valid range is between 1 and 65535.
SequenceDelay	R/W	Long	Time delay between images of the sequence. Range is from 327us to 21.42s. Dependent on VariableSequenceDelay.
VariableSequenceDelay	R/W	Boolean	If TRUE, SequenceDelay is from end of last readout to binning of next image. If FALSE, SequenceDelay is a

			constant time interval from the beginning of the last exposure to the beginning of the next exposure.
CCD Settings			
Variable	R/W	Data Type	Notes
Color	RO	Boolean	Returns TRUE is CCD sensor has color dyes; FALSE otherwise
GainSixteenBit	RO	Double	Returns the 16bit gain in e-/ADU units. The 16bit gain is for informational purposes only. It is not a programmable value. It should be noted that 16bit gain values will have slight deviation from camera model to camera model. The gain number given here is a generic approximation, based on the sensor within a particular camera model. Applications or users who wish to use the camera gain in some meaningful way, should measure the gain for their particular system, or use the value provided by Apogee Instruments in the camera data sheet.
GainTwelveBit	R/W	Long	A programmable value to select the actual gain of the camera being used. The default value is based on the particular sensor used with a camera model. The valid range of this property is from 0-1023. Applications or user who wish to use this property to change the camera gain, should experiment and test different values in order to determine the gain for their particular camera system.
PixelSizeX, PixelSizeY	RO	Double	Returns the size (width and height) of the pixels in micrometers
Sensor	RO	String	Returns the sensor model installed in the camera (I.e. "KAF401E")
SensorTypeCCD	RO	Boolean	Returns TRUE if the sensor is a CCD; FALSE if CMOS
Other			
Variable	R/W	Data Type	Notes
CameraRegister[*Index*]	R/W	Long	Reads or writes data to the camera register specified by *Index*. Applications should rarely (if ever) require use of this property. Also note that not every camera register can be read.
Image	RO	Variant	Returns a 2D SAFEARRAY, of type LONG (4 bytes per element) or INTEGER (2 bytes per element), which contains the image data. The type of data (LONG or INTEGER) returned is controlled by the associated property of ConvertShortToLong.
Line	RO	Variant	Returns a 1D SAFEARRAY of type LONG (4 bytes per element) or INTEGER (2 bytes per element) which contains the image data. The type of data (LONG or INTEGER) returned is controlled by the associated property of ConvertShortToLong.
ConvertShortToLong	R/W	Boolean	If TRUE, converts unsigned short (2 bytes per element) image data to LONG (4 bytes per element) when using the Image and Line properties. The default value of this variable after initialization is FALSE.
OptionBase	R/W	Boolean	Returns/Sets the array base index for the Image and Line properties. TRUE sets the base index to 1; FALSE sets the base index to 0. The default value of this variable after initialization is FALSE.

4 ICamDiscover

ICamDiscover provides a simple dialog box within the driver, to assist in the user's camera selection. It is a generic component designed to be quickly inserted into an application, and providing most of the arguments to the ICamera2 Init() method.

ICamDiscover contains only a single method, ShowDialog(). This method causes a modal dialog box to appear. The following description applies to this single method of the ICamDiscover interface.

Format:

```
ShowDialog( Boolean Interactive )
```

Parameters:

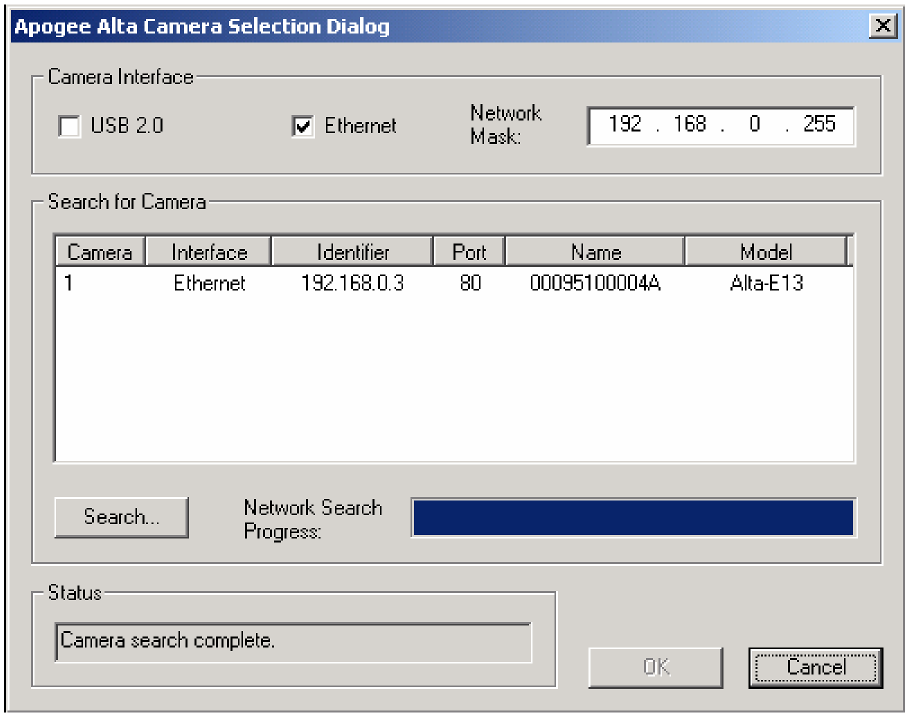
Interactive: A state variable that controls whether the displayed dialog box is interactive or not. A value of "TRUE" indicates that the dialog should be interactive, and the user will be able to select one of the cameras found during the discovery process. A value of "FALSE" indicates that the dialog box will be used only to locate cameras. Applications will almost always set this variable to "TRUE", and use the user's selection to call the ICamera2 Init() method.

Description:

The *ShowDialog()* method displays a dialog box that allows the user to query cameras either directly attached to the computer (USB2) or locally on a network (Ethernet).

The dialog has properties that may be queried or configured to provide a more "custom" look and feel to the dialog box. See the list of ICamDiscover properties for more information.

The current design of this dialog box is shown below. Note that this is a screenshot image of an interactive dialog box, and was invoked using ShowDialog(TRUE).



Network Mask: 192 . 168 . 0 . 255

Camera	Interface	Identifier	Port	Name	Model
1	Ethernet	192.168.0.3	80	00095100004A	Alta-E13

Camera search complete.

The "Search" window shows the cameras located. The user selects which interface type should be included in the search (USB 2.0 or Ethernet). If an Ethernet search is request, the user is also queried for a network mask. The default mask is 192.168.0.255, meaning that any host with an address of 192.168.0.*X* will be located.

Properties define the default state of the dialog. The USB 2.0 and Ethernet check boxes may be checked or unchecked before the dialog is display. In addition, the network mask may be changed as well.

ICamDiscover Settings			
Variable	R/W	Data Type	Notes
DlgTitleBarText	R/W	String	Sets the Title Bar of the ICamDiscover dialog box. Default value is "Apogee Alta Camera Selection Dialog".
DlgCheckEthernet	R/W	Boolean	If TRUE, sets the default state of the Ethernet check box to be enabled (checked). FALSE disables. Setting or unsetting the Ethernet check box will also enable or disable the IP Address field for the network mask. The default value is FALSE.
DlgCheckUsb	R/W	Boolean	If TRUE, sets the default state of the USB 2.0 check box to be enabled (checked). FALSE disables. The default value is FALSE.
DlgNetworkMask	R/W	Long	Returns or sets the value of the network mask field. The default value is 0xC0A800FF, which corresponds to 192.168.0.255.
DlgShowEthernet	R/W	Boolean	If TRUE, the Ethernet check box is displayed in the dialog box. A setting of FALSE will hide the Ethernet check box, and remove it from the user interface. This could be used if a particular application is specifically written for a particular

			type of camera interface. The default value is TRUE.
DlgShowUsb	R/W	Boolean	If TRUE, the USB check box is displayed in the dialog box. A setting of FALSE will hide the USB check box, and remove it from the user interface. This could be used if a particular application is specifically written for a particular type of camera interface. The default value is TRUE.
ValidSelection	RO	Boolean	If TRUE, the user has pressed the "OK" button, and selected one of the cameras in the search box. If TRUE, the values of the Selected* properties (SelectedInterface, SelectedCamIdOne, SelectedCamIdTwo) are valid. On creation of the interface, the default value is FALSE.
SelectedInterface	RO	Apn_InterfaceType	If a valid selection was made, returns the interface type as either Apn_Interface_NET or Apn_Interface_USB. Default value is Apn_Interface_NONE.
SelectedCamIdOne	RO	Long	If a valid selection was made, returns the first camera identifier. For cameras of Apn_Interface_NET, this value is the IP Address of the camera. For Apn_Interface_USB, this value is the order in which the camera was enumerated by the operating system, out of the number of cameras that were detected by the operating system.
SelectedCamIdTwo	RO	Long	If a valid selection was made, returns the second camera identifier. For cameras of Apn_Interface_NET, this value is the Port number of the camera. For Apn_Interface_USB, this value is zero (0x0).
SelectedModel	RO	String	If a valid selection was made, returns a string with the model name of the camera. If the user did not make a valid selection, this property will contain the string "No Model".

5 Example

The following is meant to provide a very simple example of usage of the ICamera2 and ICamDiscover objects with the Microsoft Visual C++ environment. This example simply creates the ICamera2 and ICamDiscover objects, attempts to locate a usable Alta camera, and then takes a test image using the camera's native data simulator. The test image is retrieved and the objects are released.

```
#include <stdio.h>

// Import the type library to create an easy to use wrapper class
#import "apogee.dll" no_namespace


void main()
{
      ICamera2Ptr       AltaCamera;       // Camera interface
      ICamDiscoverPtr   discover;         // Discovery interface
      HRESULT           hr;               // Return code

      CoInitialize( NULL );               // Initialize COM library

      // Create the ActiveX objects from the universally unique identifier
      hr = AltaCamera.CreateInstance( __uuidof( Camera2 ) );
      if ( FAILED( hr ) )
      {
            printf("Failed to create the ICamera2 object\n");
            return;
      }
      else
      {
            printf("Successfully created the ICamera2 object\n");
      }

      hr = discover.CreateInstance( __uuidof( CamDiscover ) );
      if ( FAILED( hr ) )
      {
            printf("Failed to create the ICamDiscover object\n");
            return;
      }
      else
      {
            printf("Successfully created the ICamDiscover object\n");
      }


      discover->DlgCheckEthernet = true;
      discover->ShowDialog( true );

      if ( discover->ValidSelection )
      {
            hr = AltaCamera->Init( discover->SelectedInterface,
                                   discover->SelectedCamIdOne,
                                   discover->SelectedCamIdTwo,
                                   0x0 );
```

```
	}
	else
	{
		printf( "No Valid Selection made\n");
		return;
	}

	if ( FAILED(hr) )
	{
		printf("Failed to connect to camera");
	}
	else
	{
		printf("Connected.  Camera ID:  %u\n", AltaCamera->CameraID );
	}

	long ImgXSize = AltaCamera->ImagingColumns;
	long ImgYSize = AltaCamera->ImagingRows;

	unsigned short *pBuffer = new unsigned short[ ImgXSize * ImgYSize ];

	AltaCamera->DigitizeOverscan = false;
	AltaCamera->TestMode = true;

	AltaCamera->Expose( 0.001, true );

	AltaCamera->GetImage( (long)pBuffer );

	delete [] pBuffer;      // Free buffer memory

	discover = NULL;        // Release the object
	AltaCamera = NULL;      // Release the object

	CoUninitialize();       // Close the COM library
}
```