Apogee ActiveX/COM API Specification

Applicable to

AP, KX, LISAA, and SPH Series Imaging Cameras Using Xilinx 4000 and Spartan series FPGA Engines

Supporting Parallel Port, ISA and PCI Interfaces

Also compatible with some AM (QRX firmware), AX (Q firmware) series cameras

Specification Version 1.0

Revision Date: September, 2001

Disclaimer

Apogee Instruments Inc. assumes no liability for the use of the information contained in this document or the software which it describes. The user assumes all risks. There is no warranty of fitness for a particular purpose, either express or implied.

The information contained in this document is assumed to be correct, but in no event shall Apogee Instruments Inc. be held responsible for typographical errors or changes in the software not reflected in this document.

The specifications contained in this document are subject to change without notice.

Support

The Apogee Camera Control development specification is provided as a courtesy to our customers, and comes without warranty of fitness for any purpose or application. The user assumes all risk for the use of the information contained in this document and the software it describes.

Apogee Instruments, Inc.
11760 Atwood Road, Suite #4
Auburn, CA 95603

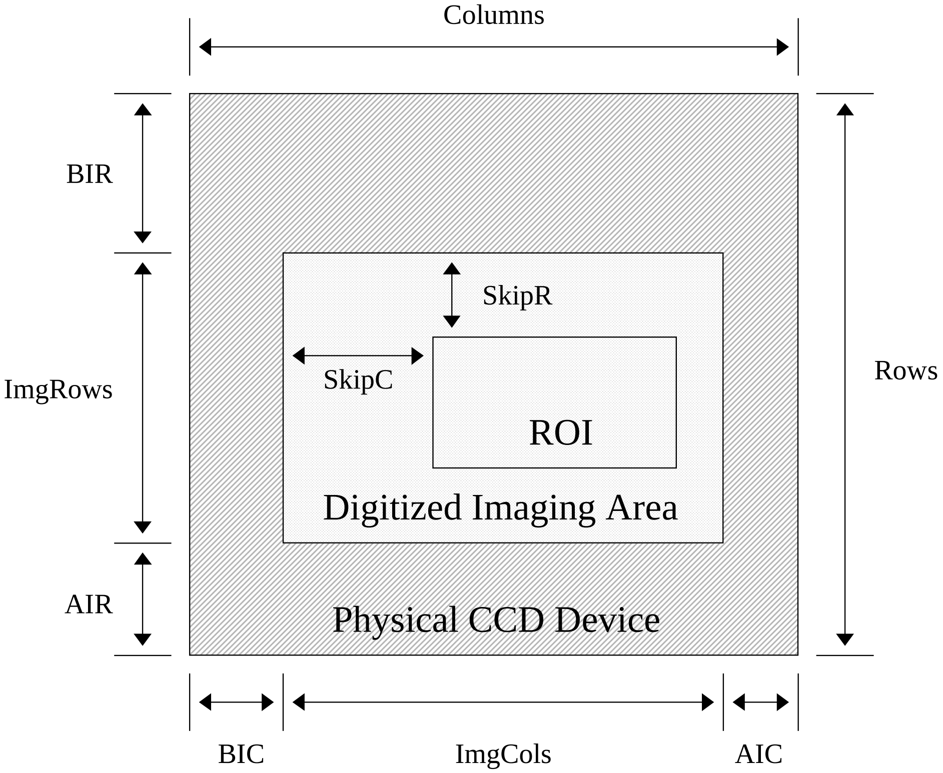
(530) 888-0500
(530) 888-0540 FAX

1 CCD Imaging Geometry

Apogee imaging cameras process and digitize pixels according to specified geometry parameters. In order to discuss these rules and algorithms, we will use the following definitions:

Parameter	Definition
Columns	Total physical count of columns on CCD device
Rows	Total physical count of rows on CCD device
Image Columns (ImgCols)	Number of columns within the Image Area (in unbinned pixels)
Image Rows (ImgRows)	Number of rows within the Imaging Area (in unbinned pixels)
BIC	Before Imaging Columns (BIC) count. Number of columns before Image Area. Specified in .INI file.
AIC	After Imaging Columns (AIC) count. Number of columns after Image Area. Internally calculated value.
BIR	Before Imaging Rows (BIR) count. Number of rows before Image Area. Specified in .INI file.
AIR	After Imaging Rows (AIR) count. Number of rows after Imag Area. Internally calculated value.
SkipC	Skip Columns. Number of columns to be digitized in the Image Area, but not actually part of the Region of Interest (ROI). Specified in .INI file.
SkipR	Skip Rows. Number of rows to be digitized in the Image Area, but not actually part of the Region of Interest (ROI). Specified in .INI file.

The following picture may be useful for visualizing the geometry:



Note in the image above that even though SkipC and SkipR are digitized pixels, they are not included as part of the final image that is presented to an application by the driver.

2 Camera Initialization Files

The API uses a configuration file to identify all characteristics unique to a camera. This eliminates the need to change driver or application software for each camera type. The industry standard .INI file format is used. It is assumed that the API or application will never write over the .INI file. Any changes to .INI settings within an application using the API will be saved elsewhere as defined by the application. The initialization file settings are not case sensitive. White space is allowed between tokens. Values of "off/0/false" or "on/1/true" are equivalent. A complete list of all .INI parameters and their descriptions is presented below.

2.1 Parameters

`[system]`

`Interface`	`Type of camera interface used`
`Base`	`CCD Controller card base address`
`Reg_Offset`	`Camera offset used for parallel port systems`
`PP_Repeat`	`Delay used for parallel port systems`
`Cable`	`Cable length`
`High_Priority`	`Thread set to high priority when dowlonding image`
`Data_Bits`	`Digitization resolution`
`Sensor`	`Type of sensor (CCD/CMOS) for future use.`
`Mode`	`Mode bits in decimal, determined by factory`
`Test`	`Test bits in decimal, determined by factory`
`Test2`	`Test2 bits in decimal, determined by factory`
`Shutter_Speed`	`Shutter time resolution (0.01 sec, 0.001 sec, dual)`
`Shutter_Bits`	`Mode and Test bits to toggle for dual speed shutters. The Mode mask is the low nibble, the Test mask is the high nibble.`
`MaxBinX`	`Maximum horizontal binning factor`
`MaxBinY`	`Maximum vertical binning factor`
`Guider_Relays`	`Camera can output to guider relays`
`Timeout`	`Maximum length of time the Frame Done bit is polled`

`[geometry]`

`Columns`	`Total columns on CCD (physical)`
`Rows`	`Total rows on CCD (physical)`
`ImgCols`	`Unbinned columns in imaging area`
`ImgRows`	`Unbinned rows count in imaging area`
`BIC`	`Before image column count (dark, non-imaging pixels)`
`BIR`	`Before image row count`
`SkipC`	`Deleted data columns`
`SkipR`	`Deleted data rows`
`HFlush`	`Horizontal flush binning`
`VFlush`	`Vertical flush binning`

`[temp]`

`Control`	`CCD temperature can be controlled`
`Target`	`CCD temperature set point`
`Cal`	`Temperature calibration factor`
`Scale`	`Temperature scaling factor`

`[ccd]`

`Sensor`	`Type of sensor installed in camera`
`Color`	`CCD sensor has color dyes`
`Noise`	`Typical readout noise in e- units`
`Gain`	`Typical camera gain in e-/ADU units`
`PixelXSize`	`Size of pixels in horizontal diraction (in micrometers)`
`PixelYSize`	`Size of pixels in vertical diraction (in micrometers)`

2.2 Ranges and Defaults

N.B. Parameters prefixed by 0x or sufixed by an H are assumed to be hexadecimal. Parameters suffixed by .0 are assumed to be floating point numbers. Decimal integer parameters can be used in their place.

Parameter	Range	Default
[system]		
Interface	ISA, PPI, PCI	required
Base	0x000 - 0xFFF	Required for PPI/ISA; Ignored for PCI cameras
Reg_Offset	0x0 - 0xF0	0x0
PP_Repeat	1 - 1000	1
Cable	short/long	short
Data_Bits	8 - 18	16
High_Priority	true/false	true
Sensor	CCD/CMOS	CCD
Mode	0x0 – 0xF	0x0
Test	0x0 – 0xF	0x0
Test2	0x0 – 0xF	0x0
Shutter_Speed	normal, fast, dual	normal
Shutter_Bits	0x0 - 0xFF	0x0
MaxBinX	1 - 8	8
MaxBinY	1 - 255	63
Guider_Relays	true/false	false
Timeout	0.0 - 10000.0	2.0
[geometry]		
Columns	1 - 65536	required
Rows	1 - 65536	required
ImgCols	1 - 4096	Columns – BIR - SkipR
ImgRows	1 - 4096	Rows – BIC - SkipC
BIC	1 - 4096	4
BIR	1 - 4096	4
SkipC	0 - 4096	0
SkipR	0 - 4096	0
HFlush	1 - 8	1
VFlush	1 - 255	1
[temp]		
Control	true/false	true
Target	-60 - +40	-10
Cal	1 - 255	160
Scale	1.0 - 10.0	2.1
[ccd]		
Sensor	Any text string	-
Color	true/false	false
Noise	Any floating point number	0.0
Gain	Any floating point number	0.0
PixelXSize	Any floating point number	0.0
PixelYSize	Any floating point number	0.0

3 Software Interface

The Apogee camera drivers provide access to all camera functions through a straightforward ActiveX (COM Automation) API. ActiveX is accessible from virtually any Windows programming or scripting language. The ActiveX driver resides in the file Apogee.dll, which can be installed anywhere on the user's system. Note, though, that the DLL must be registered with the operating system (this is done by software installers automatically, or can be done manually via the command line interface). Please see the installation README file for appropriate instruction on hardware and software installation of the Apogee system.

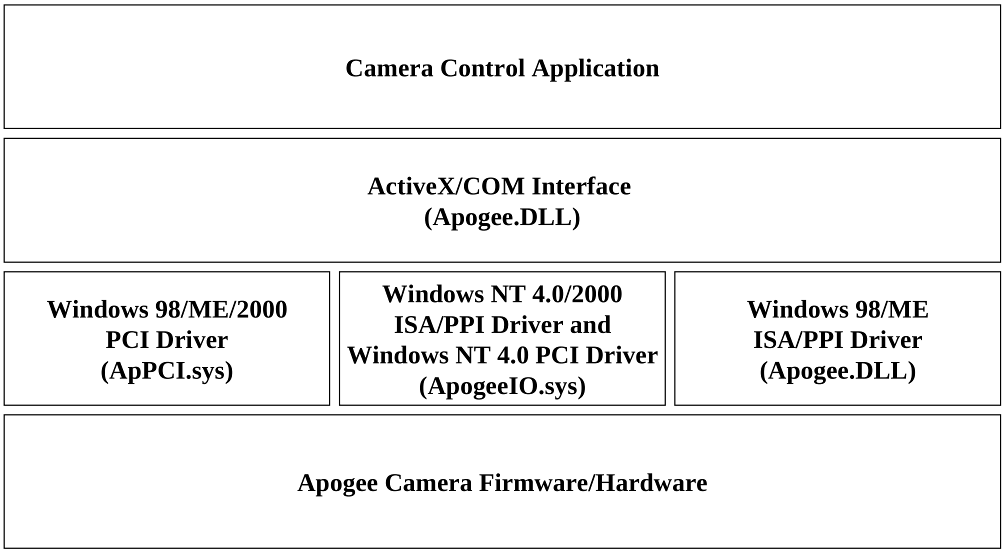
For applications where it is desirable to directly compile the driver into the program, to modify the driver for custom applications, or to use it under a different operating system, C++ source code has also been included. The interface for the C++ version is similar to the ActiveX interface described in this document, but is not identical. Please see the section Generic Class Interface for more information. Please note that the kernel-level driver must still be installed under Windows NT/2000.

3.1 ActiveX Driver

The ActiveX DLL can supply multiple Camera objects with an ICamera interface to any Windows application that can access COM objects. This includes applications written in Visual Basic, Visual C++, Delphi, Visual Java, Visual Interdev and other COM-aware languages, and scripting hosts such as Visual Basic for Applications, VBScript, JScript, PerlScript, Python etc.

The Apogee software stack for the ActiveX driver is illustrated below:

Apogee ActiveX/COM Software Stack


Camera Control Application
ActiveX/COM Interface (Apogee.DLL)
Windows 98/ME/2000 PCI Driver (ApPCI.sys)
Windows NT 4.0/2000 ISA/PPI Driver and Windows NT 4.0 PCI Driver (ApogeeIO.sys)
Windows 98/ME ISA/PPI Driver (Apogee.DLL)
Apogee Camera Firmware/Hardware

Note that PCI is only supported on Windows 98, Windows Millennium (ME), Windows NT 4.0, and Windows 2000. Apogee PCI controller cards are not supported on Windows 95.

As stated previously, the ActiveX/COM interface provides a level of indirection for the software and driver components lower in the stack. This allows the underlying Apogee architecture to change and grow, while still supporting previous products. For example, an application written using the ActiveX interface for ISA or Parallel Port (PPI) designs does not require changes or additions to support Apogee PCI controller cards.

Any API should be able to handle the requirements of a typical camera control application. A typical imaging session helps show the required, basic components, and might be structured as follows:

- Initialize the camera with an .INI file
- Load desired geometry, temperature and configuration data using various the camera properties
- Call the Expose method
- Poll camera Status property
- When ready call the GetImage method
- Poll temperature and cooler status periodically

The Apogee ActiveX/COM architecture supports this functionality, plus additional features, via the ICamera interface. The ICamera interface supports the following methods and properties.

3.1.1 Properties

The following table details the properties supported by the Apogee ActiveX driver.

Camera Settings			
Variable	R/W	Data Type	Notes
Status	Read Only	Short	Returns current camera state <0: Error codes 0: Idle 1: Waiting for trigger 2: Exposing 3: Downloading 4: Line ready 5: Image ready 6: Flushing BIR
Present	Read Only	Boolean	Returns TRUE if camera present; FALSE otherwise
Shutter	Read Only	Boolean	Returns TRUE if shutter is open; FALSE if closed
ForceShutterOpen	R/W	Boolean	TRUE forces shutter to open; FALSE allows normal shutter operation
Long Cable	R/W	Boolean	Returns/Sets long cable mode
PPRepeat	R/W	Short	Delay used on PPI camera systems
Mode	R/W	Short	Lower four bits map to Mode bits used for special camera functions or configurations
TestBits	R/W	Short	Lower four bits map to Test bits used for camera troubleshooting
Test2Bits	R/W	Short	Lower four bits map to Test2 bits used for special camera functions or configurations
DataBits	Read Only	Short	Digitization Resolution (8-18)
SensorType	Read Only	Short	Returns type of sensor used 0 or CCD: Charge Coupled Device 1 or CMOS: Complementary Metal-Oxide-Silicon
FastReadout	R/W	Boolean	Returns/Sets fast readout mode for focusing
Use Trigger	R/W	Boolean	Returns/Sets triggered exposure mode
TDI	R/W	Boolean	Returns/Sets drift scan integration mode
MaxExposure	Read Only	Double	Returns the maximum exposure duration
MinExposure	Read Only	Double	Returns the minimum exposure duration
MaxBinX	Read Only	Short	Returns the maximum horizontal binning factor
MaxBinY	Read Only	Short	Returns the maximum vertical binning factor
GuiderRelays	Read Only	Boolean	Returns TRUE if camera can output to guider relays; FALSE otherwise

Timeout	R/W	Double	Returns/Sets the maximum length of time the camera Frame Done bit is polled
Cooler Settings			
Variable	R/W	Data Type	Notes
CoolerControl	Read Only	Boolean	Returns TRUE if CCD temperature can be controlled; FALSE otherwise
CoolerSetPoint	R/W	Double	Returns/Sets the cooler set point temperature in degrees Celcius
CoolerStatus	Read Only	Short	Returns the current cooler status 0: Off 1: Ramp to set point 2: Correcting 3: Ramping to ambient 4: At ambient 5: Maximum cooling limit 6: Minimum cooling limit 7: At set point
CoolerMode	R/W	Short	Returns/Sets the current cooler operation mode 0: Off (Shutdown immediately) 1: On (Enable Cooler; Go to set point temperature) 2: Shutdown (Ramp to Ambient, then Shutdown)
Temperature	Read Only	Double	Returns the current temperature in degrees Celcius
TempCalibration	R/W	Short	Returns/Sets the temperature calibration factor (TempCelcius = (DAC units - Tcalibration) / Tscale)
TempScale	R/W	Double	Returns/Sets the temperature scaling factor (TempCelcius = (DAC units - Tcalibration) / Tscale)
Exposure Settings			
Variable	R/W	Data Type	Notes
BinX, BinY	R/W	Short	Returns/Sets the horizontal and vertical binning parameters
StartX, StartY	R/W	Short	Returns/Sets the subframe start position in terms of unbinned pixels
NumX, NumY	R/W	Short	Returns/Sets the subframe size in binned pixels
Geometry Settings			
Variable	R/W	Data Type	Notes
Columns, Rows	Read Only	Short	Returns the total number of physical columns or rows on the CCD
SkipC, SkipR	R/W	Short	Returns/Sets the number of deleted data columns that are not to be displayed
Hflush, Vflush	R/W	Short	Returns/Sets the horizontal and vertical flush binning parameters
BIC, BIR	R/W	Short	Returns/Sets the Before Imaging Columns/Rows (dark non-imaging pixels)
CCD Settings			
Variable	R/W	Data Type	Notes
Sensor	Read Only	String	Returns the sensor model installed in the camera (I.e. "SITe 502)
Color	Read Only	Boolean	Returns TRUE is CCD sensor has color dyes; FALSE otherwise

Noise	Read Only	Double	Returns the read-out noise in e-.
Gain	Read Only	Double	Returns the gain in e-/ADU units
PixelXSize, PixelYSize	Read Only	Double	Returns the size (X and Y) of the pixels in micrometers
Other			
Variable	R/W	Data Type	Notes
Image	Read Only	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 INT) returned is controlled by the associated property of ConvertShortToLong
Line	Read Only	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 INT) returned is controlled by the associated property of ConvertShortToLong
Snap (Duration as Double, Light as Boolean)	Read Only	Variant	Combination of the Expose Method and Image Property. Blocks the calling thread for the duration of the exposure and readout.
ConvertShortToLong	R/W	Boolean	Allows conversion of unsigned short (2 bytes per element) image data to long (4 bytes per element) when using the Image and Snap properties
OptionBase	R/W	Boolean	Returns/Sets the array base index for the Image and Snap properties. TRUE sets the base index to 1; FALSE sets the base index to 0.
HighPriority	R/W	Boolean	Returns/Sets whether the DLL thread is given high priority during image download (I.e. GetImage, Image and Snap).

3.1.2 Methods

The following table details the methods supported by the Apogee ActiveX driver.

System	
Function	Notes
Init (String iniFile, Short BaseAddress = -1, [Optional] Short RegOffset = -1, [Optional])	Initializes internal variables to their default value and reads the parameters in the specified INI file. If BaseAddress and RegOffset parmeters are non-negative, then these values are used instead of the INI settings for the BaseAddress and RegOffset properties. Note that PCI operation does not depend on a BaseAddress being specified. For PCI adapters, both the BaseAddress value in the INI file, as well as the BaseAddress parameter passed into this function, are ignored. An exception is thrown if the camera cannot be initialized. The error codes are: 0: No error detected 1: No config file (INI file) specified 2: Config missing or Config file missing required data 3: Loopback test failed; No camera detected 4: Memory allocation failed; System Error 5: NT I/O Driver not present

Reset()	Resets the camera to an idle state. Terminates current exposure, if exposure is in progress. Does not initiate flushing (use the Flush() method).
Flush (Short Rows = -1 [Optional])	Starts flushing the camera (the camera should be in an idle state). If Rows is a non-negative number, only the specified number of rows will be flushed. In this case, the method will return only when the flushing operation is complete
AuxOutput (Byte Value)	Outputs "Value" to an auxillary output port (e.g. for driving guider relays)
RegWrite (Short Register, Short Value)	Writes "Value" to the specified "Register". Registers 1-8 may be written to by the application.
RegRead (Short Register, Short Value)	Reads from the specified "Register". The result of the read operation is placed into the "Value" variable. Returns/Sets drift scan integration mode
FilterHome()	Move the filterwheel to the home position. Failure indicates that no filterwheel is attached or the filterwheel is broken.
FilterSlot (Short Slot)	Move the filterwheel to the position denoted by "Slot"
Normal Exposure	
Function	Notes
Expose (Double Duration, (Boolean Light)	Takes an exposure of a specified Duration (in seconds). The Light parameter controls the state of the shutter during the exposure. If Light is TRUE, the shutter opens. If Light is FALSE, the shutter will close. This method returns immediately after invocation. Poll the CameraStatus property to determine the start time of a triggered exposure, and the end of an exposure.
GetImage (Long pImageData)	Returns a pointer (pImageData) to unsigned short data in memory. The data will have (NumX * NumY) elements.
Drift Scan	
Function	Notes
DigitizeLine()	Begins clocking and digitization of a single line of data. Poll the CameraStatus property to determine when the data is ready for download.
GetLine (Long pImageData)	Returns a pointer (pImageData) to unsigned short data in memory. The data will have NumX elements.

3.1.3 Usage from Visual Basic

Accessing an ActiveX object from Visual Basic is very easy. Start Visual Basic and open the Project menu References command. In the list you will see "Apogee Camera Control Library". Turn on the check box and click OK. Now in the appropriate code section of a Form or Module enter the following:

3.1.3.1 Declaration and Initialization

```
Dim cam as Camera          ‘Declare camera object
Set cam = New Camera       ‘Create camera object
cam.Init “lisaa.ini”       ‘Initialize camera

‘Now you can then access all Apogee camera functions directly; for example:
```

3.1.3.2 Managing Temperature Control

```
‘Initializing the temperature control subsystem

cam.SetPoint = 10          ‘Set target temperature in degrees C
cam.CoolerMode = 1         ‘Turn on cooler

‘Updating temperature status (polling)

stat = cam.CoolerStatus
temp = cam.Temperature     ‘Poll temperature and status.  Space polls at least 1 second
                           ‘apart. Establish rolling average of temperature reads (16
                           ‘samples) to reduce read noise.

‘Shutting down temperature control subsystem (controlled ramp-up)

cam.CoolerMode = 2

‘When ramp-up complete (by polling cam.Coolerstatus):

cam.CoolerMode = 0         ‘Turn controller off

‘Shutting down temperature control subsystem (hard shutdown).  Only when really necessary.

cam.CoolerMode = 0
```

3.1.3.3 Take a normal exposure

```
cam.Expose 10, true        ‘10 sec exposure with shutter open

do
loop until cam.Status = Camera_Status_ImageReady

Dim ImageData as Variant
ImageData = cam.Image
‘Can now access image data as a 2D array (i.e. ImageData(100, 100) )
```

3.1.3.4 Take a dark frame

```
cam.Expose 10, false             ’10 sec exposure with shutter closed

do
loop until cam.Status = Camera_Status_ImageReady

Dim ImageData as Variant
ImageData = cam.Image
‘Can now access image data as a 2D array (i.e. ImageData(100, 100) )
```

3.1.3.5 Take a TDI (drift scan) exposure

```
Dim LineData( 1 to NumLines ) as Variant

cam.TDI = true
cam.Expose drift_rate, true      ‘specify drift_rate in seconds

for i = 1 to NumLines
       cam.DigitizeLine
       do
       loop until cam.Status = Camera_Status_LineReady
       LineData(i) = cam.Line
Next
```

3.1.3.6 Take an externally triggered exposure

```
cam.UseTrigger = true
cam.Expose 10, true              ’10 sec exposure with shutter open

do
loop while cam.Status = Camera_Status_Waiting
Print “Got Trigger”

do
loop until cam.Status = Camera_Status_ImageReady

Dim ImageData as Variant
ImageData = cam.Image
‘Can now access image data as a 2D array (i.e. ImageData(100, 100) )
```

3.1.4 Usage from Visual C++

An ActiveX object can be accessed from Visual C++ in many ways. The following example is perhaps the simplest way, taking advantage of VC++ wrapper classes.

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

ICameraPtr    cam;          // Declare a smart pointer to the camera interface
HRESULT       hr;           // Return code from ActiveX methods

CoInitialize(NULL);         // Initialize COM library

// Create the ActiveX object from the universally unique identifier
hr = cam.CreateInstance( __uuidof( Camera ) );
if ( FAILED( hr ) ) return ErrorCode;   // ErrorCode must be defined by the application

// Open the camera using an ini file
_bstr_t inifile( “lisaa.ini” );
hr = cam->Init( inifile, -1, -1 );
if ( FAILED( hr ) )
{
      cam = NULL;
      return hr & 0xFF;
}

// Access properties
short CameraXSize = cam->ImgColumns;
short CameraYSize = cam->ImgRows;

unsigned short* pBuffer = new unsigned short[ CameraXSize * CameraYSize ];
if  (pBuffer == NULL )
{
      cam = NULL;
      return ErrorCode;
}

// Take a 10 sec exposure with the shutter open
if ( FAILED( cam->Expose( 10, true ) ) )
{
      delete [] pBuffer;
      cam = NULL;
      return ErrorCode;
}

while ( true )
{     // Wait until the exposure is done and the image is ready
      if ( cam->Status == Camera_Status_ImageReady ) break;
}
```

```
// Get the image
if ( FAILED( cam->GetImage( (long) pBuffer ) ) )
{
       delete [] pBuffer;
       cam = NULL;
       return ErrorCode;
}

delete [] pBuffer;
cam = NULL;                // This will automatically release the ActiveX object

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

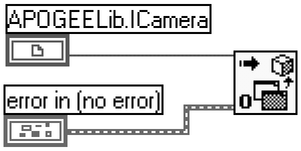
3.1.5 Usage from LabVIEW

The Apogee ActiveX DLL can be used within LabVIEW, a graphical programming environment from National Instruments. LabVIEW allows the user to control the camera system through the ActiveX DLL. Apogee does not provide an instrument driver for LabVIEW beyond the Apogee ActiveX DLL.

The easiest way to invoke the ActiveX capabilities within LabVIEW is to use LabView as an Automation Client. In this mode, LabVIEW acts as a client, and requests information from the Apogee ActiveX DLL, which is the automation server.

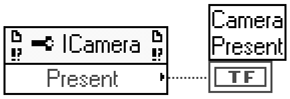
First, using your LabVIEW documentation, create an Automation Open Reference. This will allow the ActiveX DLL to be opened. The Automation Reference requires the user to select an ActiveX class in order to operate properly. Choose the option to "Select ActiveX Class" and look at the list of available ActiveX components on the machine. Note that it is not usual for many components to be registered. Select the component labeled "Apogee Camera Control Library." If the "Apogee Camera Control Library" is not present or shown as an ActiveX Class, then the Apogee.DLL has not been installed properly. Please see your installation instructions for proper installation before continuing. Once the reference has been opened, LabVIEW will refer to it in a shortened form, i.e. APOGEELib.ICamera.

The partial diagram below shows the Automation Open Reference for an ActiveX control, along with the selection of the Apogee Camera Control Library (APOGEELib.ICamera).



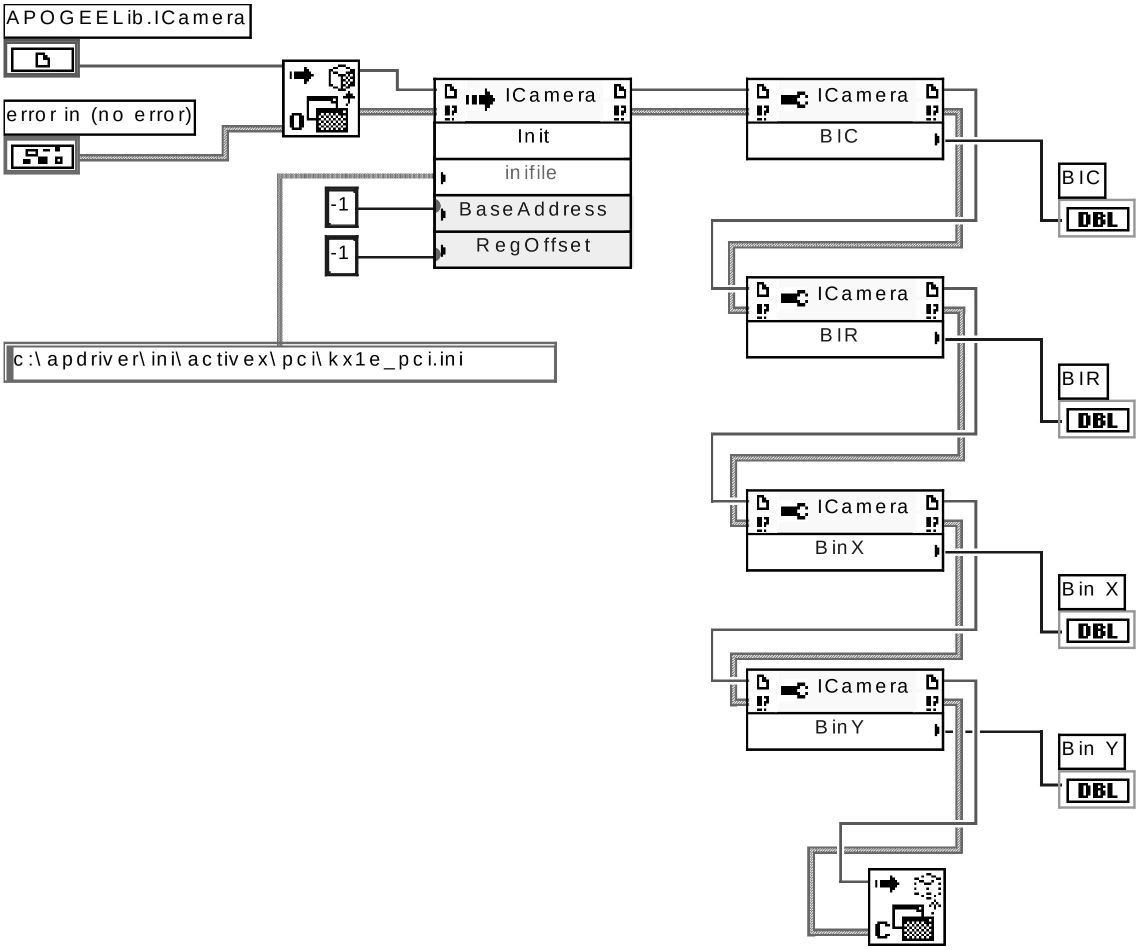
Once the Automation Reference has been opened with the Apogee ActiveX camera control, you can use the Properties and Methods available from the Automation Property Nodes and Automation Invoke Nodes. These Nodes also require an associated ActiveX Class, which should also be set to the Apogee Control (APOGEELib.ICamera). Once this is done, select the appropriate Method or Property you wish to use, and connect to the node to other LabVIEW components as appropriate.

The partial diagram below shows a Property Node (Present).



When finished with the Apogee ActiveX Control, make sure to complete operation with an Automation Close Reference.

The following diagram is a very simple LabVIEW virtual instrument, which opens an Automation Reference, initializes the camera with the Init method, and then uses the Icamera interfaces to display Before Image Columns/Rows (BIC/BIR) and the X and Y Binning values (BinX and BinY).



For more information regarding LabVIEW usage, as well as specifics of how to use LabVIEW as an Automation Client, please see the documentation provided by National Instruments.