Apogee
INSTRUMENTS

Apogee Camera Control Development 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 4.0

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 Introduction

This specification outlines the hardware and software functionality of the Apogee imaging cameras. The specification is broken into three major sections: the Hardware Interface, Initialization Files, and the Software Interface.

The Hardware Interface details the firmware engine used to control Apogee camera systems. This includes an overview of the register set and associated bit descriptions of registers. Behavioral models of the hardware are also discussed in this section. For example, temperature control is now discussed in detail, and a more thorough description of the geometry variables used during image readout is provided.

Initialization files are key to providing the bridge between the Hardware and Software components of the camera. It is important to always properly configure the camera device to known safe state, before attempting any imaging operation. The initialization files provide a list of parameters to set on the camera before using the device for the first time.

The section on the Software Interface explains the Apogee software architecture. A key component of this architecture is an API library that is packaged as an ActiveX (COM) component. Apogee Instruments does provide source code support for the camera systems. However, wherever possible, third party vendors are encouraged to the use the ActiveX DLL. The ActiveX DLL provides probably the fastest method to producing a new imaging application. Furthermore, third party applications using the DLL will find it easy to support any changes or updates that might occur in the future, because while the internal implementation of the DLL functionality may change, the external interface will remain the same.

2 Hardware Interface

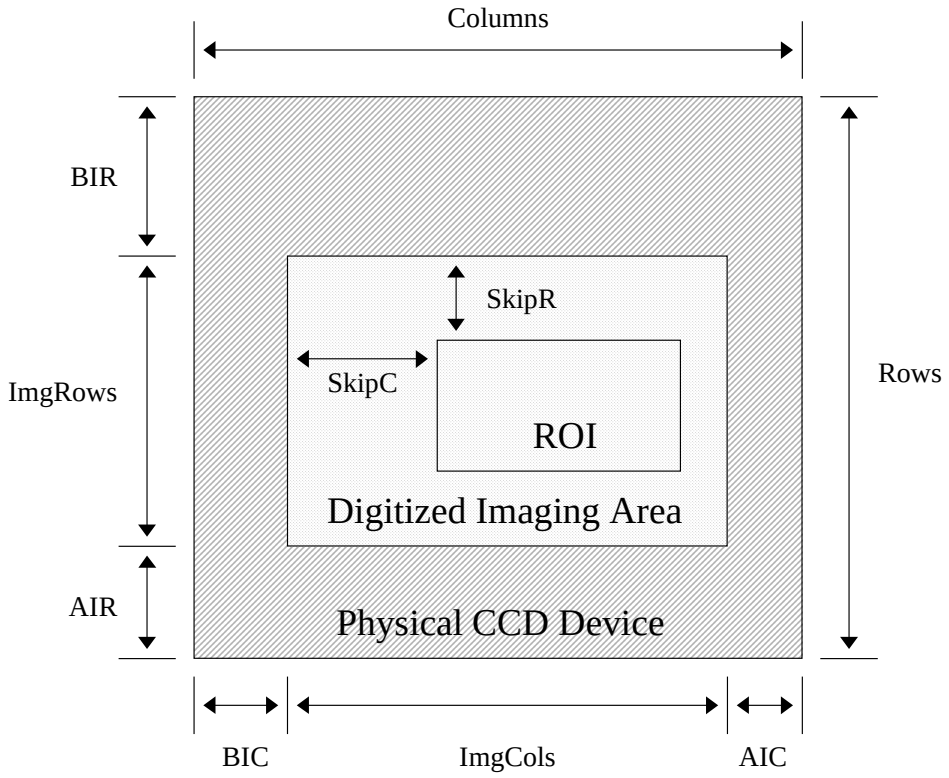
2.1 Firmware Behavior

2.1.1 Imaging Geometry

Pixels are processed and digitized 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. Also note that SkipC and SkipR are purely conventions used within the driver. The hardware firmware has no knowledge of these parameters.

2.1.1.1 Horizontal Geometry Rules

BIC	Digitized Imaging Area	AIC

Horizontally, the firmware automatically processes an entire CCD row. Each row is comprised of three sections which vary in size depending on the subframe location chosen. The first section is known as the "Before Imaging Columns" (BIC) and represents the column offset leading to the digitized area of interest. The BIC is counted on a 1:1 basis regardless of other binning parameters. For a full frame, the BIC value from the .INI file is used. The second section is the actual digitized imaging area "pixel count". The value loaded in the counter is based on groups of binned pixels. So if 100 columns within the imaging area are being binned by a factor of 4, then a value of 100/4 = 25 would be loaded into the Pixel_count variable. The third section is the "After Imaging Columns" (AIC) and represents the remaining columns of the physical CCD that still require clocking. It is calculated using the "Columns" value from the .INI file as follows:

AIC = Columns (Specified in .INI) – BIC – "Unbinned (physical) Digitized Imaging Area pixels"

2.1.1.2 Horizontal Geometry examples:

INI files settings:
[geometry]
columns=530
imgcols=512
bic=4

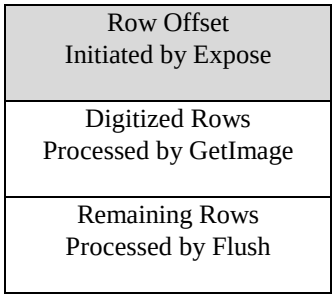
a. For a full frame image binned 1:1, the values of BIC_count, Pixel_count and AIC_count are as follows;
BIC_count = 4 (from ini file)

Pixel_count = 512 (from ini file)
AIC_count = 530 – 4 – 512 = 14

b. For a sub-frame image 50 pixels wide located at a column offset= 100 ccd columns, binned 2:2:
BIC_count = 4 (from ini file) + 100 = 104
Pixel_count = 50 / 2 = 25
AIC_count = 530 – 104 – 50 = 376

That having been said, the horizontal and vertical binning factors loaded during the Expose method are based on the .INI file setting "hflush=" and not the binning factors desired for the region of interest data. The Hflush and Vflush parameters were designed to provide a different (higher) value of binning for flushing so that faster flush cycles were possible.

2.1.1.3 Vertical Geometry Rules



Since the Expose method causes the firmware to automatically skip (clock) to the region of interest after the exposure is complete, the geometry and vertical binning for this skipped region (row offset) must be considered. The factors to be considered are BIR and vflush binning in order to determine the proper value for line_count. In addition, a residual line_count number is generated and passed to GetImage. This is explained below.

Line_count is based on unbinned rows, hence with a row offset of 4 with a vertical binning of 2 would result in a value of 4 being written to line_count. However, the complication comes apparent if the vertical binning used for the row offset exceeds the row offset itself, or if the number of skipped rows is not evenly divisible by the vertical binning factor. These two cases will be described as follows.

1. The vertical binning used for the row offset exceeds the row offset

The default vertical binning usedby the Expose method is based on "vflush=" from the INI file. If the number of offset rows (BIR) is less then vflush, then the value of BIR should be used for binning and a value of BIR written to the line_counter.

2. If the number of skipped rows is not evenly divisible by the vertical binning factor

This will occur most of the time. In this case, the greatest possible number of binned rows are assigned to line_count, and the remainder (m_ExposureRemainingLines) is passed to GetImage. GetImage will first skip the remainder before it does it's normal processing by setting the vertical binning = remainder and calling next_line one time.

2.1.1.4 Vertical Geometry Examples

INI files settings:
[geometry]
bir=4
vflush=8

a. For a full frame image, the values of vertical_binning and line_count are;

Vertical_binning = 4
Line_count = 1

Value of remaining_lines returned = 0

b. For a sub-frame image with a row offset of 150, the values of vertical_binning and line_count are;

Vertical_binning = 8 (from vflush INI setting)
Line_count = ((bir of 4) + (row offset of 150)) / 8 = 19 (rounded down)
Value of remaining_lines returned = 2

2.1.2 Temperature Calculations

The correct temperature setting values are derived from proper programming of the temperature setting registers. See the register descriptions for specific details on calculating the correct register values.

Proper cooler operation involves setting the desired temperature, and then enabling the cooler operation. Once the set temperature has been reached, R11.7 (At-temp) will go high. From this point forward, assuming no other changes to the settings, the cooler can safely be assumed to be at the desired temperature setting. Some fluctuation may exist beyond this point, which an application can take into account by conducting a rolling average of the temperature value. Again, please note that there should be no expectation that once the set temperature has been reached, R11.7 will remain high consistently—it is only to be used as an indication of the first time the set temperature was reached.

As noted above, because of temperature fluctuations, the temperature readings should be taken once or twice a second, with a rolling average of 15 to 20 values used to derive the current temperature. An application can take the temperature reading more frequently, but should not attempt more than eight readings per second (the number of times the temperature reading is updated within the firmware itself).

The following table lists various states for the cooling system.

	CommandReadback and Status Register Bits					
Status String	**R11.4**	**R11.5**	**R11.6**	**R11.7**	**R12.8**	**R12.15**
Cooler off	X	X	X	X	X	0
Ramping to temp	0	0	0	0	0	1
Correcting	0	0	0	0*	0	1
Ramping to ambient	X	X	0	X	1	1
Ramp-up Complete	X	X	1	X	1	1
Max cooling limit	0	1	0	X	0	1
Min cooling limit	1	0	0	X	0	1
At temp	0	0	0	1	0	1

* Correcting can be assumed if R11.7 (at-temp bit) was once a one but now a zero. After temperature ramp down, keep track of this bit in order to derive the “correcting” status. When first determining temp status following the opening of a camera, “correcting” can be assumed, if no other status can be derived.

2.2 Register Descriptions

Apogee cameras are line-readout devices, not frame-capture cameras. No interrupts or DMA operations are used. This means that the CCD data is read from the camera one line at a time, with polling to determine the camera states. Internal state control logic in the camera head and/or the interface card handles camera status, exposure timing, row and column counting, flushing, and temperature control. The driver is responsible for reading the correct number of rows from the camera and organizing the resulting data for display and processing.

2.2.1 Register Summary

The firmware is organized as a group of twelve 16 bit registers. Eight registers can be written to with the following (note that these registers can also be read from on PCI-based systems):

- Exposure time
- Horizontal and vertical binning
- Geometry information such as column offsets, #pixels per ROI, row offsets
- Desired Temperature
- Command bits which when toggled begin image acquisition, start/stop flushing, reset the system, or digitize and store entire rows of image data.
- Misc. control bits

Four 16 bit registers can be read from and contain the following data:

- Image data
- CCD Temperature
- Status bits used for temperature control and image readout
- Mirror of command register

Register	**Mnemonic**	**Function**
1	Reg_Command	Command bits
2	Reg_Timer	Timer bits 0-15
3	Reg_Vbinning	Timer bits 16-20, Vertical Binning
4	Reg_AICCounter	After Imaging Column (AIC), Test2 bits
5	Reg_TempSetPoint	Desired Temperature, Digital port
6	Reg_PixelCounter	Pixel Counter, Horizontal binning
7	Reg_LineCounter	Line Counter, Mode bits
8	Reg_BICCounter	Before Imaging Column (BIC), Test bits
9	Reg_ImageData	16 bit Image Data
10	Reg_TempData	CCD Temperature
11	Reg_Status	Status Register
12	Reg_CommandReadback	Command bits (Register 1) mirror

2.2.2 Reg_Command

Register Number: 1

Access:

ISA/PPI	PCI
Write Only	Read/Write

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	N/A	0x0
PCI	0x20	0x0

Programming Notes:

None.

Register Description:

Bits															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Cooler Enable	Cable Length	Focus Mode	Start Flushing	Start Next Line	Timer Load Enable	Done Reading	Cooler Shut-down	Shutter Enable	Stop Flushing	Trigger Enable	FIFO Cache	Reset System	Shutter Over-ride	Start Timer	TDI Mode

Bit	Function	Usage
0	TDI Mode	Enables drift scan operation (if camera firmware supports). If the firmware does not support Drift Scan, this bit has no effect on camera operation.
1	Start Timer with Offset	Transition this bit (1 to 0) to begin exposure, skip lines according to image row offset, and digitize and store the first line of data in FIFO buffer.
2	Shutter Override	Enabling this bit forces the shutter open. Disabling this bit has no effect on shutter control. Note that the Shutter Override bit is not used in normal operation. Normal operation should use bit 7 (Shutter Enable) to control shutter operation during an exposure.
3	Reset System	Transition this bit (1 to 0) to reset firmware.
4	FIFO Cache	Enable this bit during readout of an image line from the camera.
5	Trigger Enable	Enables the start of an exposure from an external hardware trigger source. Disable this bit when not using an external hardware trigger.
6	Stop Flushing	Transition this bit (1 to 0) to discontinue flushing.
7	Shutter Enable	0 = Dark Frame or Bias; 1 = Shuttered Exposure. Controls whether or not shutter is open while timer is active. Used to differentiate Dark and Bias frames from Images and Flat Fields.
8	Cooler Shutdown	Setting this bit will ramp the cooler to Ambient temperature at a controlled rate.
9	Done Reading	Transition this bit (1 to 0) to indicate that the current line has been read.
10	Timer Load Enable	Enable this bit while writing timer values to Reg2 and Reg3. After values are written, transition this bit back to 0 in order to use the timer. (Timer does not function when the Timer Load Enable bit is set.)
11	Start Next Line	Transition this bit (1 to 0) to initiate next line acquisition. Used after initial Start Timer command to read each successive line.
12	Start Flushing	Transition this bit (1 to 0) to initiate flushing. Flushing causes the CCD to continuous readout pixels, as if an image were being generated. This keeps the charge on the pixels to a minimum.
13	Focus Mode	Enabling this bit allows faster “lower quality” subframing. Use when speed is preferred over image quality.

14	Cable Length	0 = Short Cable; 1 = Long Cable. The following indicates appropriate values for cable length:
15	Cooler Enable	Enables cooler operation. When enabled, the internal cooler control system will seek desired temperature defined by Reg5. Disabling this bit causes the hardware to stop regulating temperature control. The camera will eventually reach Ambient temperature. Note that this is not the recommended method of turning off the cooler. For controller cooler shutdown, use the “Cooler Shutdown” bit.

2.2.3 Reg_Timer

Register Number: 2

Access:

ISA/PPI	PCI
Write Only	Read/Write

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	N/A	0x2
PCI	0x24	0x4

Programming Notes:

The camera timer is a hardware timer, capable of measuring exposure duration to 0.01 second increments. The full timer is 20 bits wide, yielding a maximum exposure time of approximately 10,485.75 seconds or roughly 2.9 hours.

A camera reset is suggested before using the internal camera timer. The reset operation will insure that the camera is not already running or in an unknown state when the timer values are loaded.

The timer is set by enabling bit 10 (Timer Load Enable) of the Command register (Reg_Command). The 20 bit timer value is then written to Reg_Timer (lower 16 bits) and Reg_VBinning (upper four bits). Once those values are written, the Timer Load Enable bit should be brought low. The timer will not function with the Timer Load Enable bit set high. Also, the Timer will not be loaded with the correct timing value if the Timer Load Enable bit is not set before writing to the Reg_Timer and Reg_VBinning registers.

For PCI camera operation, note that a read from this register will return the current value of the timer. If the timer is currently in operation,it should not be assumed that the current value of the timer register is the original (set) value of the timer.

Register Description:

Bits
15 : 0
Timer [15 : 0]

Bits	Function	Usage
15:0	Timer	The lower 16 bits of the camera's internal 20 bit timer.

2.2.4 Reg_VBinning

Register Number: 3

Access:

ISA/PPI	PCI
Write Only	Read/Write

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	N/A	0x4
PCI	0x28	0x8

Programming Notes:

None.

Register Description:

Bits			
15 : 14	13 : 8	7 : 4	3 : 0
Reserved	Vertical Binning [5 : 0]	Reserved	Timer [19 : 16]

Bits	Function	Usage
3:0	Timer	The upper four bits of the camera's internal 20 bit timer.
7:4	Reserved	Reserved for future use.
13:8	Vertical Binning	Sets the Vertical Binning count to a value in the range 0-63.
15:14	Reserved	Reserved for future use. (These bits may be used to set an 8 bit vertical binning value in future products and/or firmware.)

2.2.5 Reg_AICCounter

Register Number: 4

Access:

ISA/PPI	PCI
Write Only	Read/Write

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	N/A	0x6
PCI	0x2C	0xC

Programming Notes:

None.

Register Description:

Bits	
15 : 12	11 : 0
Test2	AIC Counter [11 : 0]

Bits	Function	Usage
11:0	AIC Counter	Defines the number of columns to be skipped after the imaging columns in a subframe have been digitized. The valid range of values is 1-4096, inclusive.
15:12	Test2	Defines special camera functions or configurations. Specified via .INI file, and loaded when the device is initialized. Test2 bits are used for camera settings set at the Factory.

2.2.6 Reg_TempSetPoint

Register Number: 5

Access:

ISA/PPI	PCI
Write Only	Read/Write

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	N/A	0x8
PCI	0x30	0x10

Programming Notes:

This register is used for writing temperature data to the device. A conversion must be applied in order to write the correct value, given a certain temperature specified in degrees Celsius. This conversion is determined by using a zero-point and scale factor from the configuration file.

Given the following variables:

ZP = Zero-Point (Specified in .INI file)
T = Temperature in degrees Celcius (Desired temperature to set)
SF = Scale Factor (Specified in .INI file)
Value = Raw Temperature (Value we will write to the device)

We can determine the temperature in degrees Celsius using the following formula:

Value = ZP + (T * SF)

So, supposing we want to set a temperature of –10 degrees Celsius, and have specified a zero-point of 0x115 and a scale factor of 2.5 in the .INI file, we get

Value = 0x115 + (-10 * 2.5) = 0x90

Register Description:

Bits	
15 : 8	7 : 0
Relay Control	Temperature Value

Bits	Function	Usage
7:0	Temperature Value	Current temperature value, as determined from a set zero-point and scaling factor. An application must convert this value into degrees Celsius.
15:8	Relay Control	Relay control.

2.2.7 Reg_PixelCounter

Register Number: 6

Access:

ISA/PPI	PCI
Write Only	Read/Write

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	N/A	0xA
PCI	0x34	0x14

Programming Notes:

The pixel counter is a 12 bit counter. This defines the limit (4096) to the number of pixels which can be digitized by the Apogee camera. Note that this does not include AIC or BIC pixels, only those pixels to be digitized. See the previous section on hardware geometry parameters for more information.

Register Description:

Bits		
15	14 : 12	11 : 0
Reserved	Horizontal Binning	Pixel Counter

Bits	Function	Usage
11:0	Pixel Counter	Denotes the number of pixels to be digitized on a given line of the CCD.
14:12	Horizontal Binning	Sets the Horizontal Binning count to a value in the range 1-8. (A value of “000” in these bits is treated as a binning value of “1”)
15	Reserved	Reserved for future use.

2.2.8 Reg_LineCounter

Register Number: 7

Access:

ISA/PPI	PCI
Write Only	Read/Write

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	N/A	0xC
PCI	0x38	0x18

Programming Notes:

None.

Register Description:

Bits	
15 : 12	11 : 0
Mode	Line Counter

Bits	Function	Usage
11:0	Line Counter	The number of rows to be flushed before the Frame Done Status (Reg11.11) goes high.
15:12	Mode	Defines special camera functions or configurations. Specified via .INI file, and loaded when the device is initialized. Mode bits are used for camera settings set at the Factory

2.2.9 Reg_BICCounter

Register Number: 8

Access:

ISA/PPI	PCI
Write Only	Read/Write

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	N/A	0xE
PCI	0x3C	0x1C

Programming Notes:

The BIC count is always given in terms of unbinned pixels..

Register Description:

Bits	
15 : 12	11 : 0
Test	BIC Counter

Bits	Function	Usage
11:0	BIC Counter	Defines the number of columns to be skipped before the imaging columns in a subframe have been digitized. The valid range of values is 1-4096, inclusive.
15:12	Test	Defines special camera functions or configurations. Specified via .INI file, and loaded when the device is initialized. Test bits are used for camera settings set at the Factory

2.2.10 Reg_ImageData

Register Number: 9

Access:

ISA/PPI	PCI
Read Only	Read Only

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	0x0	N/A
PCI	0x0	N/A

Programming Notes:

None.

Register Description:

Bits
15 : 0
16 bit Image Data

Bits	Function	Usage
15:0	Image Data	16 bit image data from a single pixel.

2.2.11 Reg_TempData

Register Number: 10

Access:

ISA/PPI	PCI
Read Only	Read Only

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	0x2	N/A
PCI	0x4	N/A

Programming Notes:

This register is used for reading temperature data from the device. A conversion must be applied in order to convert the value into degrees Celsius. This conversion is determined by using a zero-point and scale factor from the configuration file.

Given the following variables:

ZP = Zero-Point (Specified in .INI file)
T = Temperature in degrees Celcius (Unknown)
SF = Scale Factor (Specified in .INI file)
Value = Raw Temperature (Read from lower byte of this register)

We can determine the temperature in degrees Celsius using the following formula:

$T = (Value – ZP) / SF$

So, supposing we read a value of 0x90 from this register, and have specified a zero-point of 0x115 and a scale factor of 2.5 in the .INI file, we get

$T = (0x90 – 0x115) / 2.5 = -10$ degrees Celsius

Register Description:

Bits	
15 : 8	7 : 0
Reserved	Temperature Value

Bits	Function	Usage
7:0	Temperature Value	Current temperature value, as determined from a set zero-point and scaling factor. An application must convert this value into degrees Celsius.
15:8	Reserved	Reserved for future use.

2.2.12 Reg_Status

Register Number: 11

Access:

ISA/PPI	PCI
Read Only	Read Only

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	0x6	N/A
PCI	0x8	N/A

Programming Notes:

None.

Register Description:

Bits												
15 : 12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	Frame Done	Got Trigger	Rsvd	Rsvd	At Temp	Shut-down Done	Temp Max	Temp Min	Rsvd	Cache Read OK	Line Done	Expose In Progress

Bit	Function	Usage
0	Expose In Progress	Bit is set to indicate that an exposure is in progress. While high, poll the Frame Done bit (Reg11.11) to determine when the image is complete.
1	Line Done	Bit is set when imaging line has been processed and is ready for readout.
2	Cache Read OK	Unused in driver. Goes high when cached FIFOs will allow reading to begin on the next line.
3	Reserved	Reserved for future use.
4	Temp Min	Bit is set when the temperature cannot be driven more positive.
5	Temp Max	Bit is set when the temperature cannot be driven more negative.
6	Shutdown Done	Bit is set to indicates that a controlled cooler shutdown has finished. This is in response to the “Cooler Shutdown” bit being set in the Command register (Reg1.8).
7	At Temp	Indicates the point when the cooler has reached the desired temperature. Temperature can be assumed to reach the set point the first time “At Temp” goes high (even if the bit subsequently goes low).
8	Reserved	Reserved for future use.
9	Reserved	Reserved for future use.
10	Got Trigger	Bit is set when an external trigger has been received.
11	Frame Done	Bit is set when the number of rows indicated by the line counter (Reg_LineCounter (Reg7)) have been flushed.
15:12	Reserved	Reserved for future use.

2.2.13 Reg_CommandReadback

Register Number: 12

Access:

ISA/PPI	PCI
Read Only	Read Only

Location (Offset from Base Address):

Interface	Read	Write
ISA/PPI	0x8	N/A
PCI	0x10	N/A

Programming Notes:

None.

Register Description:

Bits															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Cooler Enable	Cable Length	Focus Mode	Start Flushing	Start Next Line	Timer Load Enable	Done Reading	Cooler Shut-down	Shutter Enable	Stop Flushing	Trigger Enable	FIFO Cache	Reset System	Shutter Over-ride	Start Timer	TDI Mode

The description and usage of these bits is the same as for the Command (Reg_Command) register. Please see that register for a detailed explanation of the bit fields.

3 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.

3.1 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)
```

3.2 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

4 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.

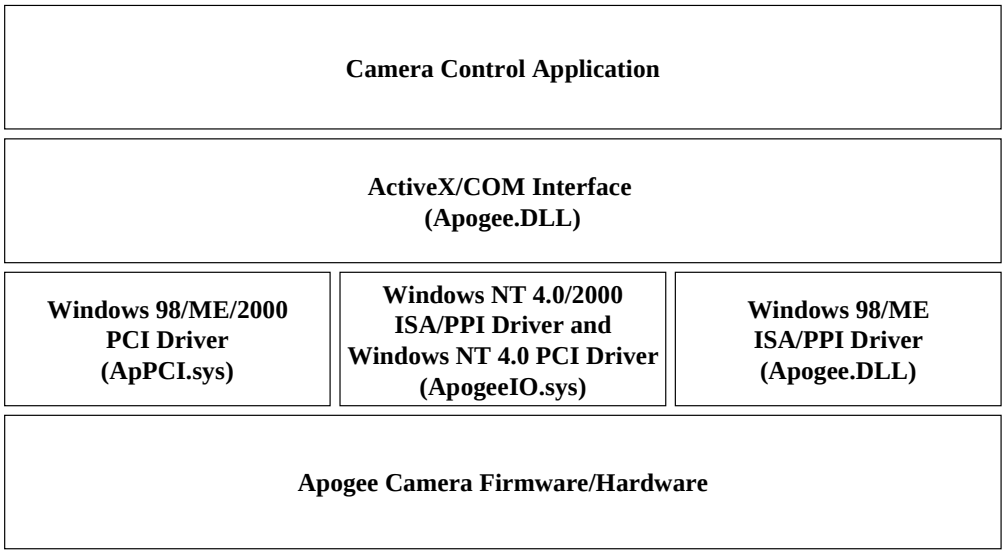
For applications where it is desirable to directly compile the driver into the program, to modify the driver for custom applications, or to use it under a different operating system, C++ source code has also been included. The interface for the C++ version is similar to the ActiveX interface described 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.

4.1 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



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.

4.1.1 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).

4.1.2 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.

4.1.3 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:

4.1.3.1 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:
```

4.1.3.2 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
```

4.1.3.3 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) )
```

4.1.3.4 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) )
```

4.1.3.5 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
```

4.1.3.6 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) )
```

4.1.4 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
```

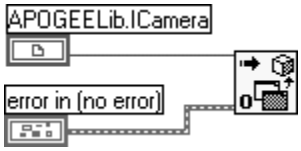
4.1.5 Usage from LabVIEW

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).

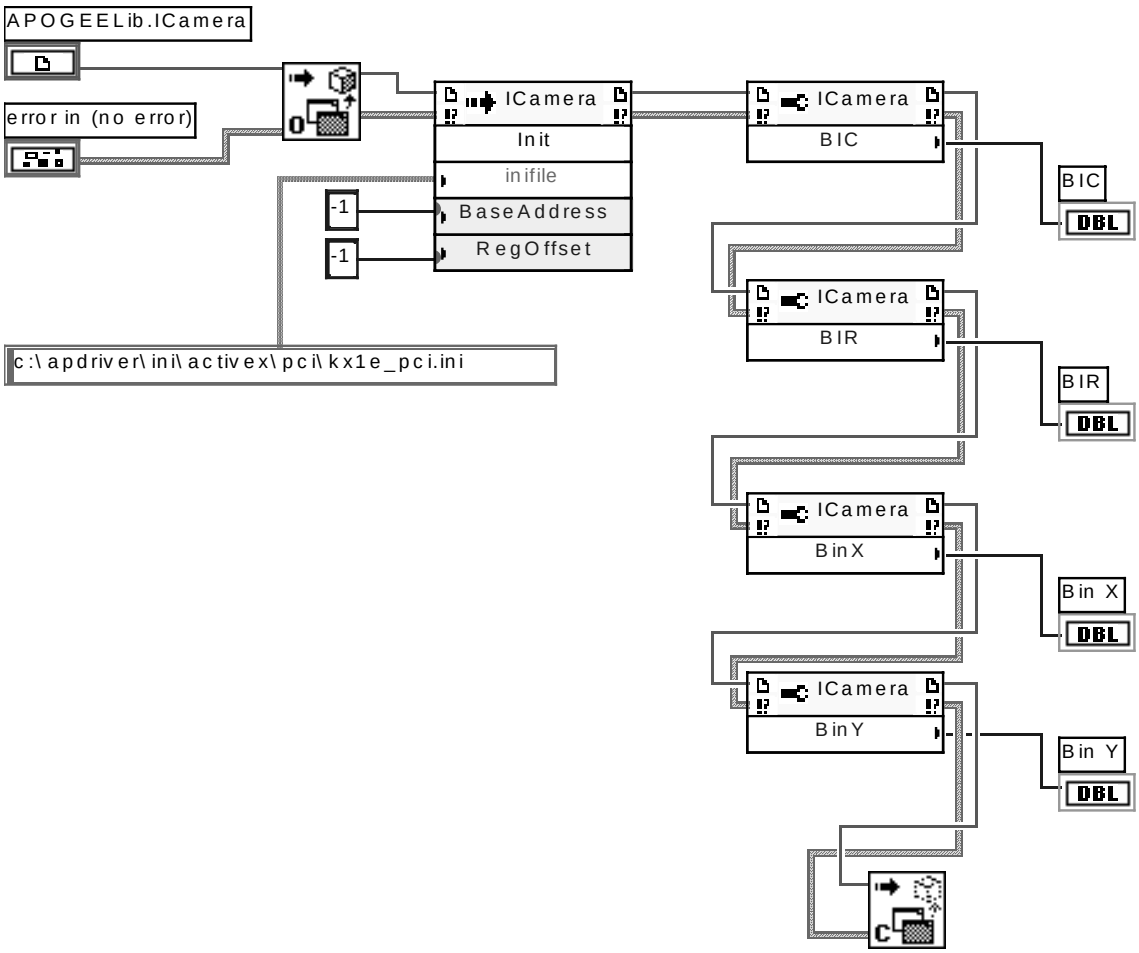


Once the Automation Reference has been opened with the Apogee ActiveX camera control, you can use the Properties and Methods available from the Automation Property Nodes and Automation Invoke Nodes. These Nodes also require an associated ActiveX Class, which should also be set to the Apogee Control (APOGEELib.ICamera). 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 from National Instruments.

4.2 Generic Class Interface

Normally it is recommended that the included ActiveX driver be used. In cases where custom modification is required to the driver, or ActiveX cannot be used (e.g. non-Microsoft operating systems), the source code can be compiled directly into your application.

The "Apogee Driver Source" directory contains the source code. It has been developed under Microsoft Visual C++, but should be portable to other environments with minimal modification.

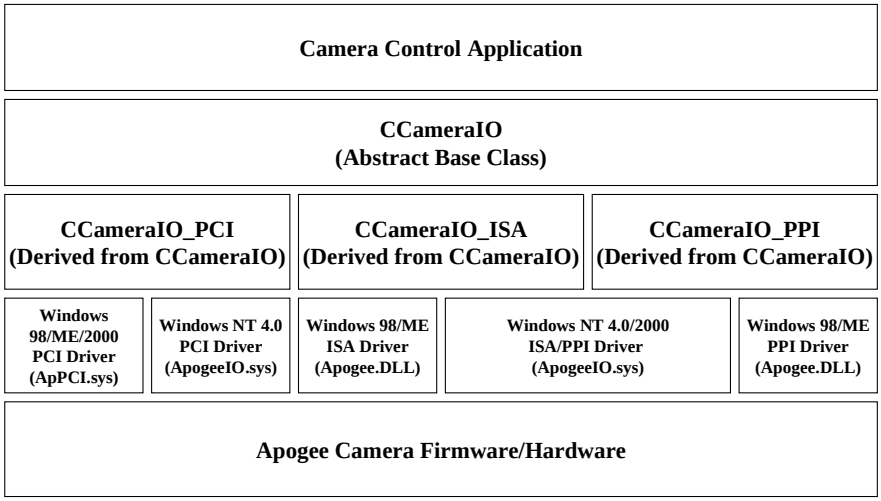
Instead of an ActiveX interface, a set of C++ classes are available:

Interface	Windows 95/98/ME	Windows NT/2000
ISA	CCameraIO_ISA_9x	CCameraIO_ISA_NT
Parallel Port	CCameraIO_PPI_9x	CCameraIO_PPI_NT
PCI	CCameraIO_PCI	CCameraIO_PCI

These classes are all derived from the base class CCameraIO, which contains code that is common to all cameras. This is a virtual class and cannot be created directly. Also note that the PCI control interface is not supported on Windows 95.

The following diagram illustrates the software stack for the C++ class interfaces:

Apogee Class Library Software Stack



The class interface is very similar but not identical to the ActiveX interface. Please review the CCameraIO.h header file for a detailed explanation of the interface.

The ActiveX Init method is not part of the class interface since it instantiates the derived classes. A generic replacement routine for the Init method, called InitCam is provided along with the actual helper functions (config_load, CfgGet, hextoi and trimstr) called by the Init method to read from the INI file and set the camera default values. These routines can be found in the Camera_Example.cpp file in the Apogee Driver Source directory.

4.3 Architectural Notes

4.3.1 Read/Write Substitutions for Parallel Port Operation

4.3.1.1 Settings

Parallel ports have particular settings, defined within the .INI initialization files.

4.3.1.1.1 Parallel port type (ECP verses standard Bi-directional)

For ECP ports, an additional port setup must be done in order to place the ECP port in standard bi-directional mode. It is suggested to perform this step every time, since it will not hurt a bi-directional port. Prior to communication with the port, the I/O must be converted to bi-directional mode by writing the following value to the port address (defined by .INI file): base + 0x402h, 0x034h.

4.3.1.1.2 PP_Repeat parameter

The "pp_repeat" function was created as a .INI file setting to allow for signal settling on long parallel port cables. This function is used in the detailed sequences below to place a time space at critical points.

4.3.1.2 Usage

This API specification assumes that reads and writes are done to a single address. For parallel port operation where we allow multiple cameras on the same parallel port, substitutions should be used to as a replacement to the read and write commands. The following describes the operation of the parallel port interface. The parallel port operation requires a simple 8 bit bi-directional bus. THERE ARE NO CHANGES TO CAMERA OPERATION BEYOND THE READ/WRITE REPLACEMENT ROUTINES DESCRIBED HERE and base address/register locations.

The camera uses the port signals as follows;

Data Register (Port Base Address)

> **Bits 0-7 – 8 bit data** to and from camera. All INP and OUTP instructions send 16 bit data to the registers via this port. First Low byte and then High byte.

Control Register (Port Base Address + 2)

> **Bit 0** – Latch: Used to latch data to camera or bring data from camera.
> **Bit 1** – Select/Transfer: When high, data written to port selects a camera register. When low, data written to the port makes data available to the selected register.
> **Bit 2** – Read/Write: When high, data is written to registers. When Low, data is read from registers.
> **Bit 3** – Low/High: Selects byte to be written or read. When low, high byte is selected. When high, low byte is selected.
> **Bits 5 and 7** – Port Output Disable: When high, disables parallel port outputs and allows read functions.

The preceding port assignments are used in three routines. These include REGISTER_SELECT, INP AND OUTP. It is assumed that unless explicitly stated, all bits not specifically changing are left at their previous value.

4.3.1.3 REGISTER_SELECT Description

Common to both the **INP** and **OUTP** functions, **REGISTER_SELECT** tells the camera which 16 bit register will be operated on by the INP and OUTP instructions. It also identifies the camera on the bus as defined by the reg_offset parameter in the .INI file (see INI file description). The firmware is preset for a particular offset on the parallel port bus. The Register_select function always precedes a inp or outp function. The byte passed during a register select is defined as follows:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Offset bit128	Offset bit64	Offset bit32	Offset bit16	Register bit3	Register bit2	Register bit1	Register bit0

Register values selected by the register_select function and described in the bitmap later in this document are as follows:

DATA(hex)	REGISTER	FUNCTION
x0	1	Misc Commands – write only
x2	2	Lower Timer – write only
x4	3	Upper Timer/Vbin – write only
x6	4	AIC Counter – write only
x8	5	Temp/Relay – write only
xa	6	Pixel Counter – write only
xc	7	Line Counter – write only
xe	8	BIC Counter – write only
x0	9	FIFO Data – read only
x2	10	Desired Temperature – read only
x6	11	Status Register – read only
x8	12	Register 1 Mirror – read only

Offset values selected by the register_select function and defined by the reg_offset command in the .INI file are as follows:

DATA(hex)	FUNCTION
1x	Reg_offset=16
2x	Reg_offset=32
4x	Reg_offset=64
8x	Reg_offset=128

4.3.1.4 Detailed Parallel I/O sequences

4.3.1.4.1 Register_Select

a. Write the following bits to the control register (Base +2). C0=low, C1=high, C2=high, C3=high, C5=low, C7=low.
b. Write 8 bit camera register selection to Data port (Base Address). Valid values are:

Repeat “c” as many times as **“pp_repeat=”** specifies
c. Write the desired register_select bits to the control register (Base +2). C0=high

d. Write the following bits to the control register (Base +2). C0=low
e. Write the following bits to the control register (Base +2). C1=low.

4.3.1.4.2 OUTP

a. Write the following bits to the control register (Base +2). C0=low, C1=low, C2=high, C3=high, C5=low, C7=low.
b. Write low byte to Data port (Base Address).

Repeat “c” as many times as **“pp_repeat=”** specifies
c. Write the following bits to the control register (Base +2). C0=high

d. Write the following bits to the control register (Base +2). C0=low
e. Write the following bits to the control register (Base +2). C3=low
f. Write high byte to Data port (Base Address).

Repeat “g” as many times as **“pp_repeat=”** specifies
g. Write the following bits to the control register (Base +2). C0=high

h. Write the following bits to the control register (Base +2). C0=low

4.3.1.4.3 INP

a. Write the following bits to the control register (Base +2). C0=low, C1=low, C2=high, C3=high, C5=high, C7=high.
b. Write the following bits to the control register (Base +2). C2=low

Repeat "c" as many times as **"pp_repeat="** specifies
c. Write the following bits to the control register (Base +2). C0=high

d. Read low byte from Data port (Base Address).
e. Write the following bits to the control register (Base +2). C0=low
f. Write the following bits to the control register (Base +2). C3=low

Repeat "g" as many times as **"pp_repeat="** specifies
g. Write the following bits to the control register (Base +2). C0=high

h. Read high byte from Data port (Base Address).
i. Write the following bits to the control register (Base +2). C0=low,
j. Write the following bits to the control register (Base +2). C2=high
k. Write the following bits to the control register (Base +2). C3=high.