SBIG
ASTRONOMICAL
INSTRUMENTS

SANTA BARBARA INSTRUMENT GROUP
147-A Castilian Drive
Santa Barbara, CA 93117
Phone (805) 571-SBIG (571-7244) FAX (805) 571-1147
e-mail:sbig@sbig.com home page:www.sbig.com

SBIG Class Library

January 17, 2005

Introduction

In order to help customers write custom software for SBIG cameras we are publishing a set of C++ classes in source form. These classes link with the SBIG Universal Driver/Library and remove the burden of having to write a lot of software to perform basic camera control and image acquisition with SBIG's cameras. Highlights include:

- ***CSBIGCam*** class encapsulates camera behavior and supports all Parallel and USB based cameras. It supports up to four active cameras at one time.
- ***CSBIGImg*** class encapsulates image acquisition and supports reading and writing of SBIG Compressed and Uncompressed images (Type 3).
- ***CSBIGImg*** class has compile time option that can write FITS files in conjunction with NASA-Goddard's CFITSIO library.
- Both classes are platform agnostic using all standard library functions and have been tested under Windows, Mac OSX and Linux.

The CSBIGImg Class

The CSBIGImg class is stand-alone, having no dependencies on other classes other than the Standard Library **string** class. The class is implemented in the **csbigimg.cpp** and **csbigimg.h** files. All class member variables are **private** and accessed through accessor functions. The class methods are discussed individually below:

Constructors/Destructors

CSBIGImg() – Standard constructor. Initializes all member variables but allocates no image buffer.
CSBIGImg(height, width) – Alternate constructor that allocates an image buffer of the passed size.
~CSBIGImg() – Standard destructor, frees any allocated image buffer.

Buffer Access

GetImagePointer() – Returns a pointer to the 16-bit (unsigned short) image buffer. The buffer is exactly height * width pixels in length. Images are stored along a row whereby pixel(column, row) is at **GetImagePointer()**[row*width + column];
GetHeight()/GetWidth() – Return the size of the image buffer.
AllocateImageBuffer(height, width) – Deletes then allocates an image buffer of the passed size.
DeleteImageData() – Deletes the image buffer. Typically you can allow the destructor to do this or if reallocating image buffers the **AllocateImageBuffer()** to do it.

Image Storage/Retrieval

SaveImage(path, format) – Saves image in the passed path and format. Supports SBIG Compressed, SBIG Uncompressed and if INCLUDE_FITSIO is set supports FITS with CFITSIO Library. Returns an error code. SBIG Images are in conformance with the Type 3 specification. FITS files are 16-bit and written in accordance with the SBIGFITSEXT Specification, Version 1.0.
OpenImage(path) – Reads SBIG Compressed and SBIG Uncompressed format images from the passed path, returning an error code.
Get/SetImageModified() – Indicates whether the image data has been modified since it was last saved.

Image Processing

AutoBackgroundAndRange() – Sets the background and range member variables based upon the histogram of the pixel values. Does not modify pixel values. This is the same algorithm used by CCDOps and works well with most astronomical images.

HorizontalFlip() – Flips the image about the center column.

VerticalFlip() – Flips the image about the center row.

Accessor Functions

These functions allow access to the private class member variables. You typically don't need to call them yourself as the CSBIGCam class sets them for you when acquiring data.

Get/SetApertureArea() – The aperture area in square inches as saved in SBIG header.

Get/SetBackground/Range() – The background or black level and range as saved in SBIG header.

Get/SetBinning() – Horizontal and vertical binning used to acquire the image.

Get/SetCameraModel() – Name of the camera model used to acquire the image.

Get/SetCCDTemperature() – CCD Temperature in Degrees C when image acquired.

Get/SetEachExposure() – For multiple exposures the length of each exposure in seconds.

Get/SetEGain() – The electronic gain of the camera in electrons/ADU.

Get/SetExposureState() – The exposure state (ABG, Correlated Double Sampling, etc.) for the image.

Get/SetExposureTime() – The total exposure length in seconds.

Get/SetFilter() – The name of the optical filter used.

Get/SetFocalLength() – The camera lens or telescope focal length in inches.

Get/Set/AddHistory() – A string of characters codes representing modifications made to the image.

Get/SetImageNote() – A text based comment attached to the image.

Get/SetImageStartTime() – GMT when the light frame was started. Several overloaded Set methods are available. Set with no parameters sets it to the current time.

Get/SetNumberExposures() – For multiple exposures the total number of exposures.

Get/SetObserver() – The observer's name.

Get/SetPedestal() – Any constant pedestal value removed from each pixel to stop overflow.

Get/SetPixelHeight/Width() – Pixel dimensions in mm (not microns).

Get/SetReadoutMode() – The readout modes used to acquire the data.

Get/SetResponseFactor() – CCD calibration factor between electrons and 0^{th} magnitude.

Get/SetSaturationLevel() – The maximum ADU that the camera can produce.

Get/SetSoftware() – Name of the software acquiring the images.

Get/SetSubFrame() – For sub-frame images the upper-left corner of the image, 0 based.

Get/SetTrackExposure() – For self-guided images, the tracking exposure in seconds.

FITS Only Accessor Functions

These are only available when built with the INCLUDE_FITSIO compile time option and are for creating header entries for the FITS files. The class is implemented in the **csbigcam.cpp** and **csbigcam.h** files.

Get/SetApertureDiameter() – The telescope aperture diameter in inches.

Get/SetFITSObject() – The name of the object.

Get/SetFITSTelescope() – Description of the telescope.

The CSBIGCam Class

The CSBIGCam class depends on the CSBIGImg class for image acquisition and also uses the Standard Libraries **string** class. All class member variables are **private** and accessed through accessor functions. The class methods are discussed individually below:

Constructors/Destructors

CSBIGCam() – The standard constructor, initializes all member variables but doesn't assign a port (LPT, USB, etc) to the camera.

CSBIGCam(device type) – Alternate constructor whereby the SBIG Universal Driver/Library is opened and the passed device is opened. This works with USB and LPT ports where you don't have to specify the port address but wont work with Ethernet. Use the other alternate constructor below for Ethernet. As constructors return no values use the **GetError()** method to see if the Open Driver and Open Device calls were successful. This does not try to establish a link to the camera which is done separately with the **EstablishLink()** method described below.

CSBIGCam(OpenDeviceParams) – Second alternate constructor like the one above except by passing the OpenDeviceParams you can specify the LPT base address or Ethernet IP address. Don't forget to use the **GetError()** method to see if the opens worked.

~CSBIGCam() – Standard destructor. Closes the device and the driver in case they were left open.

Error Reporting Functions

GetError() – Returns an error code for the last error that occurred in the calling the class methods.

GetErrorString(err) – Returns an error string for the passed error code.

GetErrorString() – Returns an error string for the last error that occurred.

High Level Data Acquisition Functions

GrabImage(image pointer, dark frame) – The main data acquisition routine that acquires an entire image. Allocates the image buffer and acquires the image from the active CCD using the set readout mode, exposure, etc. Based upon the passed parameter acquires a dark frame, a light frame or a dark subtracted light frame. This routine is synchronous meaning it hangs until the image acquisition is complete or an error occurs. Returns an error code.

Mid Level Data Acquisition Functions

DumpLines(noLines) – Dumps the passed number of lines on the active CCD in the set readout mode, returning an error code.

IsExposureComplete(&complete) – Checks to see whether an exposure on the active CCD is complete, returning an error code.

ReadoutLine(ReadoutLineParams, dark subtract, data pointer) – Reads out a row of data based upon the passed parameters, returning an error code.

StartExposute(shutter state)/EndExposure() – Starts or stops an exposure on the active CCD of the set exposure time and passed shutter state, returning an error code.

StartReadout(StartReadoutParams)/EndReadout() – Starts or stops the readout of the active CCD after the exposure is complete, returning an error code.

General Purpose Functions

ActivateRelay(relay, time) – Activates the passed relay for the passed time or cancel all relay activations of the time is zero, returning an error code.

ADToDegreesC(ad, ccd=TRUE) – Converts the passed temperature sensor A/D reading to degrees C for the CCD or Ambient thermistor.

AOTipTilt(AOTipTiltParams) – Passes the passed parameters on to the AO, returning an error code.

CFWCommand(CFWParams, &CFWResults) – Passes the passed parameters on to the CFW, returning an error code.

CheckLink() – Returns TRUE if a link has been established or can be establish to a camera.

EstablishLink() – Tries to establish a link to a camera, returning an error code. Use the **GetCameraType()** method to see what type of camera was found.

IsRelayActive(relay, &active) – Sets the active parameter TRUE if the passed relay is active, returning an error code.

QueryTemperatureStatus(&enabled, &ccdTemp, &setpointTemp, &percentTE) – Queries the status of the temperature control in the camera, setting the passed parameters accordingly and returning an error code.

Accessor Functions

These functions allow access to the private class member variables and the camera state data. You'll typically call these to configure the camera prior to data acquisition.

Get/SetABGState() – For cameras with variable ABG this method controls it or reports its setting.

Get/SetActiveCCD() – Select the Imaging, Tracking or External Tracking CCD for cameras that support it.

GetCameraType() – Returns a camera type code for the camera found on the **EstablishLink()** method.

GetCameraTypeString() – Returns a string describing the type of camera found.

GetCCDTemperature(&temperature) – Returns the CCD temperature in the passed parameter, returning an error code.

GetCommand() – Returns the last command issued to the SBIG Universal Driver/Library.

GetDriverInfo(request, GetDriverInfoResults0) – Returns the results of the passed request in the passed results parameter, returning an error code.

Get/SetExposureTime() – Controls the exposure time in seconds.

GetFullFrame(&width, &height) – Returns in the parameters the full size of the active CCD in the current readout mode, returning an error code.

Get/SetReadoutMode() – Controls the readout mode used in the **GrabImage()** and other methods.

Get/SetSubFrame(left, top, width, height) – Controls the coordinates of sub-frames used in the **GrabImage()** method. Setting the height or width to zero uses the full frame.

Low Level SBIG Universal Driver/Library Functions

OpenDevice(OpenDeviceParams)/CloseDevice() – Opens or closes a device, returning an error code. Can be called automatically by the alternate constructors and if called manually must be after a call to the **OpenDriver()** method.

Open/CloseDriver() – Opens or closes the SBIG Universal Driver/Library, returning an error code. Can be called automatically by the alternate constructors.

SBIGUnivDrvCommand(command, *params, *results) – A catch-all method to make calls directly to the SBIG Universal Driver/Library, returning an error code.

Sample Programs

See the **main.cpp** file for a small sample program using the Class Library. Once you get used to the Class Library you'll be able to write fairly complicated image acquisition programs in a short period of time. In fact the following 5 lines is all you need to acquire an image and save it on disk:

```
pCam = new CSBIGCam(DEV_USB);
pImg = new CSBIGImg();
pCam->EstablishLink();
pCam->GrabImage(pImg, SBDF_LIGHT_ONLY);
pImg->SaveImage("MyImage.sbig", SBIF_COMPRESSED);
```

Also, the **sbig2fits.cpp** program is an incredibly simple yet powerful user contributed application of the Class Library where in only a few lines you can write a SBIG to FITS file format converter.

Finally, if you have any suggestions about how we can improve the Class Library please send us your comments. We always like to hear from you and we take making your life easier as part of our job.

References

- From the SBIG Developers Web Page <http://www.sbig.com/sbwhtmls/devswframe.htm>
  - "**SBIG Type 3 Image File Format**", December 1, 2004
  - "**SBIGFITSEXT Version 1.0**", March 19, 2003
  - "**SBIG Universal Driver/Library**", Version 4.43, January 11, 2005
- NASA-Goddard
  - <http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html>