Using SBIG Cameras in Igor Pro

Daniel A. Briotta, Jr.

01/08/08

0 Why?. . . And What. . .

The Santa Barbara Instruments Group (SBIG) provides a number of different CCD cameras and accessories for astronomical imaging. This package adds operations and functions to Igor that enable it to control SBIG cameras through SBIG's low-level driver routines. This allows you to write `Igor Pro` macros which can script the behavior of an SBIG camera for whatever application you may have. Most available software for these cameras, *e.g.* `Equinox` and SBIG's own `CCDOps`, provide point-and-click operation for basic camera functions such as taking a picture, moving the filter wheel, and so on. Some provide limited scripting capability. Some will even allow you to control two or more cameras simultaneously. So why this XOP? This XOP is for people who need to control a camera or cameras in ways that go beyond the limited functionality of the available programs. Do you want to use the tracker CCD to guide your telescope during a long exposure? Do you want your computer to do other processing during a long exposure? Do you need to control up to four different cameras simultaneously? With this XOP, the low-level routines in SBIG's camera driver are available to you as high-level calls from within `Igor Pro` macros and functions. What you can do with these calls is limited only by the capabilities of the camera, SBIG's driver, and your own ingenuity.

Up to four cameras may be connected, either through a USB connection, or through SBIG's Ethernet-to-Parallel (E2P) adapter. This package supports the following cameras and accessories:

ST-7E/8E/9E/10E
ST-5C/237/237A (PixCel255/237)
ST-1K, ST-2K
ST-L Large Format Camera
ST-402 Camera
AO-7, AOL
CFW-8, CFW-9, CFW-10, CFW-L

This package is in universal form, so it will run on either a PowerPC or an Intel Macintosh running OS-X v10.4 or later. It also requires the installation of the Universal Binary SBIG Driver (Version 4.47), which should have come with your camera. If not, you can dowload it from SBIG's web site at `http://www.sbig.com/sbwhtmls/softpage.htm` (scroll down to "Section V. Macintosh Users").

1 Checking out the Hardware

Before doing anything else, you should make sure that the camera hardware and driver are working properly. Follow the instructions in SBIG's camera manual to install the driver. (You may need to reboot after installing.) Then, turn the camera on. If the camera has a filter wheel, you may notice it turning after the power is applied. Finally, plug the camera into the USB port. If the driver has been installed correctly, the LED on the camera should blink a bit and, in a couple of seconds, the fan should start. This tells you that the driver you just installed is speaking to the camera.

Now you should run the CCDOPS program that SBIG supplied with the camera. Following the manual, try getting the camera to do a few things: open/close the shutter, move the filter wheel, take an image, etc. This will tell you that the SBIG software is speaking properly to the driver and the camera. Now, you are ready to try it inside `Igor Pro`.

2 Installing the Software

To enable `Igor Pro` to speak to SBIG's camera, you need to copy two files to subfolders in the `Igor Pro` folder. Put the files "`SBIG.xop`" and "`SBIG Help.ihf`" somewhere safe on your disk. (They should not be placed directly in `Igor Pro`'s directories where they could be discarded when you update `Igor Pro`.) Make aliases of the two files, and place them in `Igor Pro`'s "Igor Extensions" folder. (The descriptions in this document assume you have done this.) You then need to restart `Igor Pro`. When `Igor Pro` starts up, it will now include the `SBIG XOP` and its help file. (`Igor Pro` may ask to compile the help file the first time it loads.)

Now you can check to see if the help file was loaded into `Igor Pro` properly. Open the help window and select the "`Command Help`" tab. Make sure the `Functions` and `Operations` selectors either say "`All`" or "`External`". Scroll down to the s's and you should see all of the XOP's commands. (They all begin with the letters '`sbig`'.) Clicking on them should show the description in the right hand pane of the help window.

3 Connecting to the Camera

Turn the camera on and plug it into the USB port. After the fan starts, you can scan the USB bus to see if the camera is there. In `Igor Pro`'s command window, type: "`sbigScanUSB`" and hit return. This will print to the history area a list of all the cameras, and their serial numbers, that the driver has found on the bus. SBIG's driver is capable of handling up to four cameras. In what follows, I will assume only one camera has been connected. See below ("Using Multiple Cameras") if you want to control more than one camera. Also note, due to a limitation in SBIG's driver, the `sbigScanUSB` command only works properly if you are not currently connected to any SBIG cameras. Finally, connect to the camera by issuing the `sbigConnect` command.

While `sbigConnect` is connecting to the camera, it will print some information dowloaded from camera in the history area. It will also create a directory "`root:sbig0:`" which will contain all of the variables, strings, and waves that describe the capabilities of the camera. You should scan through this directory, using `Igor Pro`'s Data Browser. The data in this directory is described in the help file and in Table 1 below.

There are many different options for controlling the camera. I will divide them into 4 sections:

controlling the temperature, controlling the filter wheel, taking exposures, and miscellaneous peripheral controls.

4 Controlling the Temperature

The first thing I usually do, after connecting to the camera, is to turn on the thermoelectric cooler. The cooler the CCD chip gets, the lower the thermal noise will be and the better your images will look.

The "sbigSetCooler" command allows you to control all aspects of the thermoelectric cooler. The cooler can run at either a fixed power level or it can, within limits, adjust its power to regulate the imaging CCD's temperature to match your set temperature. I usually turn the cooler on at 100% power and see how cold the cooler can make it. I then set it to regulate the temperature a few degrees warmer than the coldest it can reach.

Constant Power Mode

To set the cooler at full power, use "sbigSetCooler 2,255". The '2' specifies constant power mode, and the '255' is the maximum level. To set the cooler to half power, use "sbigSetCooler 2,128". To turn the cooler off use "sbigSetCooler 0". A variable in the camera's data (`root:sbig0:Tccd`) is updated every few seconds with the current temperature of the CCD. This update happens during `Igor Pro`'s idle loop, so if `Igor Pro` is not idling, the temperature (and other so-called 'idler variables') will not be updated.

In addition to `root:sbig0:Tccd`, several other idler variables are also updated with the temperature to indicate the status of the cooler. These variables are `root:sbig0:coolerAutoFreeze`, `root:sbig0:coolerFrozen`, `root:sbig0:coolerRegulating`, and `root:sbig0:coolerSetPoint`. The meaning of these variables are described below. Also, some older cameras also returned data about the ambient temperature in the variable `root:sbig0:Tamb`, however more recent SBIG cameras have eliminated this sensor and always return a default fixed temperature (25°C).

Temperature Regulation

To regulate the temperature at, say, −30°C use "`sbigSetCooler 1,-30`". The '1' sets the temperature regulation mode, and the second parameter specifies the desired temperature in Celsius. The driver software will periodically adjust the cooler power to try to maintain the temperature you set. If you set the temperature goal too low, however, the cooler will pin at 100% power (power level 255), and will not be able to regulate the temperature very well. If this happens, you should set the cooler to regulate at a higher temperature. While the cooler is in temperature regulation mode, the variable `root:sbig0:coolerRegulating` will be set to 1 and the variable `root:sbig0:coolerSetPoint` will be the desired temperature.

Freezing the Cooler

If temperature regulation changes the power level of the cooler *while* you are reading imaging data from the camera, the resulting power change might cause a glitch in the readout (which could appear as a bad pixel or pixels in your image). If this is a problem, it can be avoided by "freezing" the cooler. While frozen, the cooler is temporarily in the constant power mode at the power level it

had when the freeze command was issued. The cooler remains in constant power mode until it is unfrozen. (Note that this blocks temperature regulation. So the cooler shouldn't be left frozen for long.)

You can freeze the cooler manually (using "`sbigSetCooler 3`" to freeze it just before the readout, and "`sbigSetCooler 4`" to unfreeze it right after the readout) or you can have the driver do it for you by enabling the autofreeze mode with "`sbigSetCooler 5`". In autofreeze mode, the driver will automatically freeze the cooler before every readout, and unfreeze it when the readout ends. (You can disable the autofreeze with the "`sbigSetCooler 6`" command.)

5 Controlling the Filter Wheel

If the SBIG driver detects a filter wheel on your camera, two variables in the camera's data directory will tell you what it found. `root:sbig0:cfwModel` contains an integer that specifies the filter wheel model, and `root:sbig0:cfwPositions` tells you how many positions the filter wheel supports. The table to the right shows the code stored for each type of filter wheel. Note that code 9 ("CFW-10 serial") is not supported by this XOP.

Code	Filter Wheel
0	Unknown
1	CFW-2
2	CFW-5
3	CFW-8
4	CFW-L
5	CFW-402
6	(unused)
7	CFW-6A
8	CFW-10
9	CFW-10 serial
10	CFW-9
11	CFW-L8
12	CFW-L8G

To move the filter wheel to a specific filter position, use "sbigSetFilter *n*", where *n* runs from 1 to `root:sbig:cfwPositions`. You can also tell the filter wheel to recalibrate itself by issuing "sbigSetFilter 0". (This calibration is normally done by the camera on power up. You shouldn't have to repeat it.) Calibration normally leaves the filter wheel in position 1.

You can monitor the progress of the wheel's motion with the "`sbigFWStatus`" function. This function returns a 1 if the wheel is stopped, and a 2 if the wheel is moving. If there is an error, the function will return 0. The following `Igor Pro` function will tell the filter wheel to move to a specified position, wait until the wheel stops, and returns a zero if everything went well, or a negative number if an error occurred.

```
function SetFilter(pos)
    variable pos
    variable moving
    NVAR numFilters = root:sbig0:cfwPositions
    if ( (pos<0) || (pos>numFilters) )
        return -1
    endif
    sbigSetFilter pos
    do
        moving = sbigFWStatus(0)
    while (moving==2)
    if (moving!=1)
        return -2
    endif
    return 0
end
```

6 Taking Exposures

SBIG cameras are capable of making a variety of different kinds and sizes of exposures, as well as varying the length of the exposure. I will discuss several of the options below. But first, let's look at the `Igor Pro` code to take a simple 0.25 second exposure and read it out to an `Igor Pro` array:

```
make/n=(1,1) myImage    // image data will go here
variable status
sbigStartExposure 0,0.25,1  // begin exposure
do
    status = sbigExposureStatus(0) & 3 // wait until done
while (status != 3)
sbigEndExposure 0
sbigReadout 0,0,myImage // read out the image
```

This code fragment assumes that you have only a single camera, connected as `sbig0`. It will take an 0.25 sec exposure on the imaging CCD with an open shutter, wait for the exposure to finish, and read it out at full resolution to the `Igor Pro` wave `myImage`. The wave `myImage` will be coerced to single precision, and will be redimensioned to fit the size of the image that was read out.

6.1 Parameters for sbigStartExposure:

CCD selection

Most SBIG camera have two CCDs: a large main CCD (the "imager"), and a smaller "tracking" CCD used to help guide telescopes during long exposures. In addition, many SBIG cameras can also connect to an external "remote tracking head" which contains an imaging chip identical to the camera's internal tracker CCD. It is possible to have simultaneous exposures running on both the main CCD and either the internal or remote tracker. (You do need to be careful with the shutter modes, however. See below.) The first parameter to the `sbigStartExposure` command specifies which of these CCD's is to begin taking an exposure.

Exposure Time

The maximum exposure time for the imaging CCD is 167,777.16 ($= 2^{24}/100$) seconds. For the tracker, the maximum is 655.35 ($= [2^{16}-1]/100$) seconds. (Note that the exposure times are rounded to the nearest 0.01 sec by the SBIG driver.) The minimum exposure time, however, depends on whether the camera has a mechanical or an electronic shutter. Due to physical constraints, cameras with mechanical shutters have a minimum exposure of 0.12 seconds if the shutter is to be opened or closed for the exposure. Cameras with electronic shutters, however, have a special "millisecond resolution" mode which can take very short exposures from 1-255 ms. You can tell if your camera is capable of millisecond exposures by checking if the variable `imagerCapabilities` has bit 1 set for your camera. (See the description of the `sbigStartExposure` command below.) To take a millisecond resolution exposure, use the `sbigStartExposure` command with the `/M` flag, and make the exposure time an integer from 1-255. Warning: if the specified exposure time is greater than 255 milliseconds, SBIG's driver will round it up to the nearest hundredth of a second and attempt a normal, non-millisecond exposure. Also, the model ST-402 camera, although it has an electronic shutter, has a minimum exposure time of 40 milliseconds. If you attempt a shorter exposure with this camera, it will take a 40 ms exposure.

Shutter Modes

The last required parameter to `sbigStartExposure` specifies one of three shutter modes available. The modes are:

0 : Do not move shutter for exposure,

1 : Open shutter for exposure, close it for readout, and

2 : Leave shutter closed for exposure and readout.

Shutter mode 1 is used for normal ("light") exposures. Mode 2 is used for "dark" exposures. Mode 0 is used for simultaneous exposures on both the imager and tracker CCD's. The light and dark exposures are typical exposures for all CCD camera operations. The simultaneous mode, however, is useful, for example, to take a short exposure on the tracking CCD without disturbing the shutter position in use during a longer exposure on the imaging CCD. The following code fragment shows how to take a long (2 minute) exposure on the imaging CCD and several overlapped 1-second tracking exposures:

```
variable imgStatus, trkStatus
make/o/n=(2,2) trkImage, myImage
sbigStartExposure 0,120,1  // open shutter and begin 2 minute exposure on imager
do
    sbigStartExposure 1,1,0 // tracker exposure, leave shutter alone!
    do
        trkStatus=sbigExposureStatus(0) & 12
    while (trkStatus!=12)
    sbigReadout 1,1, trkImage
    \\ insert code here to examine tracker image and guide the telescope
    imgStatus = sbigExposureStatus(0) & 3 // is imager done?
while (imgStatus != 3)
sbigReadout 0,1,myImage // read out the image
```

Note that in the code above, the imaging CCD continues its long 2-minute exposure while the several tracker exposures are made and read out. Also note that the tracker exposures leave the shutter undisturbed to avoid interfering with the 2 minute exposure on the imaging CCD.

If you have an optional remote tracking head installed on your camera, its internal shutter opens and closes along with the shutter of the main camera body: *i.e.* both shutters open and close at the same time. This follows the behavior of the internal tracker CCD. If you want to do guiding exposures with a remote head, you can follow the sample code above, with modifications to `sbigStartExposure` and `sbigReadout` to select the remote guide head instead of the internal tracker. Also note that the remote tracking head uses the internal tracker's timing electronics, so that it is not possible to make simultaneous exposures with both the internal and remote trackers. (You could, however, take sequential ones.)

Optional Parameters:

Anti-blooming Gate: If your camera has an anti-blooming gate, you can activate it for a given exposure by setting the optional fourth parameter of `sbigStartExposure` to a non-zero number.

Faster Image Repetition Rate: Some SBIG cameras have the option to reduce the image collection time by bypassing the time consuming reduction of the readout amplifier's power at the start of an exposure. This will speed up the image repetition rate, but the heat of the readout

amplifier will cause a glow in the upper-left corner of the imaging CCD. The time gained by suppressing the read amplifier's power reduction could be useful, for example, when you want a fast partial-image readout to facilitate rapid focusing of the telescope. To disable powering-down of the readout amplifier, use the `/F` flag with the `sbigStartExposure` command.

Triggered Exposures: You can tell the camera to wait for an external electronic signal before starting an exposure with the `/T` flag. When this flag is used with `sbigStartExposure`, the beginning of the exposure is suspended until receipt of the correct signal on the external trigger input to the camera. See your camera's manual for details on how to connect this external signal properly. After the `sbigStartExposure/T` command has been given, while the camera is waiting for receipt of the external trigger, a special exposure status is returned. See the next section.

6.2 Exposure Status

The `sbigExposureStatus()` function returns a single number that describes the current exposure status of both the imager and tracker CCD's, each in two bits of the returned value. The low two bits contain the status of the imager, and the next two bits the status of the tracker. You can extract the status of the individual CCD's using `Igor Pro`'s bitwise AND operator '&'. To check the status of the imaging CCD, use "`status=sbigExposureStatus(0) & 3`". Status values of 0, 1, and 3 indicate the idle, exposing, and done states. Similarly for the tracker CCD, use "`status=sbigExposureStatus(0) & 12`". Values of 0, 4, and 12 indicate the idle, exposing and done states respectively.

Note that, if you have a remote tracker head, SBIG's driver uses the same exposure timing electronics for both the internal and remote tracking CCD's. Both also report their exposure status on the *same two bits*. Hence, you will also use "`status=sbigExposureStatus(0) & 12`" for the remote tracker head. This effectively precludes simultaneous exposures on both the internal and remote trackers. You should only begin an exposure on either tracker when *both* are idle.

For a triggered exposure, the exposure status reports a special value to indicate that the camera is still waiting for the external trigger in order to begin the exposure. To test this value, use "`sbigExposureStatus(0) & 32768`". If the value is not zero, the camera is still waiting for the trigger signal.

6.3 Image Readout

After an exposure is finished, the resulting image data can be read out to an `Igor Pro` array with the `sbigReadout` command. Parameters specify which CCD to read, the readout mode, and the `Igor Pro` wave to receive the data. An optional set of four parameters also allow you to specify a rectangle smaller than the full CCD to be read out. The `Igor Pro` array that receives the image will be automatically converted to single precision (or unsigned word if the `/I` flag is used) and resized to the dimensions of the image being read out.

Readout Modes and Binning

Binning refers to combining individual pixels electronically to make larger effective pixels. From a data standpoint, this is equivalent to having a CCD with fewer, but larger pixels. This makes the readout faster (fewer pixels need to be converted), at the expense of resolution (the effective pixel size is the sum of the sizes of the individual pixels). Readout binning is specified by the *mode*

parameter of `sbigReadout`. The standard square binning modes, in pixels, are 1×1 (mode=0, full resolution), 2×2 (mode=1, medium resolution), or 3×3 (mode=3, low resolution). It is also possible to do rectangular binning on some cameras (see the next section). Some cameras also support a large 9×9 binning, specified by mode 9.

Binning in the horizontal direction is always performed directly on the CCD chip. However you can choose to have binning in the vertical direction done off-chip. (CCD's without anti-blooming gates can bloom in both the horizontal and vertical direction. Binning off-chip can remove the horizontal blooming.) You specify this off-chip mode by adding 6 to the binning mode. Hence mode 6 is still 1×1 binning, but modes 7 and 8 specify 2×2 and 3×3 binning with the vertical binning performed off-chip by external electronics.

Extra Vertical Binning

For some applications, *e.g.* spectroscopy, it is possible to have the camera perform a larger binning in the vertical direction. Vertical binning is specified by adding 3 to the mode command, and also adding 256 $\times$ the desired additional vertical binning. For example, a mode of "$mode = 3 + 256 * 10$" would result in 1×10 binning. Similarly, "$mode = 4 + 256 * 10$" would result in 2×20 binning, and "$mode = 5 + 256 * 10$" would give 3×30 binning. Note that horizontal binning is specified as 1, 2, or 3 pixels with modes 3, 4, and 5, respectively. Vertical binning is specified in units of the horizontal bin size. So the last example ("$mode = 5 + 256 * 10$") could also be viewed as first binning the array in 3×3 mode, then performing an additional binning of 10 of the larger pixels in the vertical direction. Note that the vertical bin size must be *at least* one binned pixel or else an error will be returned.

Partial Frames

The `sbigReadout` command has the capability of reading out only a rectangular subsection of the full image, called a partial frame. Reading out a partial frame can be much faster than reading out the entire array. This could be useful if, for example, you wanted to concentrate on a single star in the image for purposes of focusing or tracking, and wanted to maximize the frame rate. A partial frame readout is indicated by adding arguments to the `sbigReadout` command that specify the top row and left column of the first pixel to be read out, followed by the total width and height of the readout box, in pixels. Note that if you are in a binned readout mode, the four numbers are all in terms of the binned pixels. For example, suppose we are using a camera whose imaging CCD has an unbinned size of 1024×1024 pixels. If we read the array with 2×2 binning, the full image would have a size of 512×512 pixels. If instead we issue the command "`sbigReadout 0,1,MyWave,100,200,75,150`", we would get an image of 75×150 pixels extracted from the binned frame starting with row (X) 100 and column (Y) 200.

7 Miscellaneous Commands

7.1 sbigSetMisc

The `sbigSetMisc` command can be used to open, close, and initialize the shutter; to turn the camera's fan on or off; or to change the behavior of the camera's LED. It is rarely used, except to re-initialize the shutter. See the command description below.

7.2 Using the External Relays

SBIG cameras contain four relays that can be connected, for example, to the North, South, East, and West controls of a telescope for guiding purposes. A single command `sbigActivateRelay` activates the relays labeled "+X", "–X", "+Y" and "–Y". Four parameters control the lengths of time, in hundredths of a second, for which each of the relays will be active. (From the telescope's point of view, these times will be seen as how long the directional buttons were pressed.) Note that each relay can have a different activation time. Also note that, although it is possible to activate all four relays with this command, for controlling a telescope you would never activate +X and –X (or +Y and –Y) at the same time. A time of zero means that the corresponding relay will not be activated. Because the times for all four relays are specified together, you cannot have overlapping start times: after giving the activate command you should wait until all the relays have timed out before issuing another one, otherwise, you will abort any activations in progress. In fact, the command "`sbigActivateRelay 0,0,0,0`" can be used to cancel any relay activations in progress.

The current status of the relays can be examined with the `sbigRelayStatus` function. The status of each relay is encoded as one bit of the returned number. The value returned will be the *sum* of 8(+X active) + 4(–X active) + 2(+Y active) + 1(–Y active). A returned value of 0 means all the relays are inactive.

7.3 Using an Adaptive Optics Unit

Two commands support an attached SBIG Adaptive Optics unit. `sbigAOCenter` moves the AO mirror to its nominal center position. `sbigAOTipTilt` moves the mirror to some other position. See the command descriptions below.

8 Running Multiple Cameras

The SBIG driver is capable of running up to four cameras at once. Using more than one camera is, in principle, not much more difficult that using only one, but in practice it calls for a higher degree of detailed programming to keep the individual cameras straight.

Camera Data Directories

When you connect to a camera, a number of variables, strings and waves are created which identify the camera and specify its capabilities. (See Table 1 below.) These data are stored in a subdirectory unique to each camera. These directories, and the data in them, are created if necessary by the `sbigConnect` operation. Directories will be created in the order in which cameras are connected, and named `root:sbig0:`, `root:sbig1:`, `root:sbig2:` and `root:sbig3:`. So, for example, the serial number of the third camera connected can be found in the string `root:sbig2:serialNumber`. Similarly, the model number of the filter wheel attached to the second camera connected will be in the variable `root:sbig1:cfwModel`. In general, you should think of the cameras as being numbered 0, 1, 2, and 3 in the order in which `sbigConnect` was called on them.

Controlling Individual Cameras

Most of the commands in this XOP can take an argument of the form `/C=n`, where `n` is an integer from 0–3 identifying which camera should receive the command. The four functions that return a

camera status also take a single integer argument to specify the target camera. The integers 0–3 refer to the cameras whose data is stored in `root:sbig0:` through `root:sbig3:` respectively. Hence, sending a command to a specific camera is relatively straightforward. The difficulty is in making sure that the camera you send it to is actually the one you want.

Indentifying Cameras and Connecting in the Right Order

When you have more than one camera plugged in to the USB bus, the driver assigns them to ports USB0 through USB3 in the order in which the driver recognizes them. Unfortunately, this order depends on such things as the order in which the cameras were plugged in, or the order in which they were powered up. This could change from one run to the next. For example, suppose you had two cameras plugged in (call them A and B) and your software always connected in the sequence `sbigConnect/USB=0; sbigConnect/USB=1`. Then one day you may find camera A connected to the data in directory `root:sbig0:`, and B in `root:sbig1:`, and the next day they may be the other way around! What's worse, the first day you would select camera A with the `/C=0` flag, and the second day you would need to select camera A with the `C=1` flag!

The secret to avoiding confusion between the cameras is to use the camera's serial numbers, as reported by the `sbigScanUSB` operation, to always connect camera A first and camera B second. The function below shows how to do this. (Note that SBIG's serial 'numbers' are actually 9-digit strings.)

```
function ConnectAB()
    SVAR SerialNumbers = root:sbig_SerialNumbers
    string SNA = "...(9-digit serial number for A)..."
    string SNB = "...(9-digit serial number for B)..."
    sbigDisconnect/A/Q
    sbigScanUSB/Q
    variable uA = WhichListItem(SNA,SerialNumbers)
    sbigConnect/USB=(uA)
    variable uB = WhichListItem(SNB,SerialNumbers)
    sbigConnect/USB=(uB)
    return 0
end
```

Note that this function always leaves camera A selected with `/C=0` (and its data in "`root:sbig0:`"), and camera B selected with `/C=1` (and its data in "`root:sbig1:`"), regardless of the order in which A and B appear in the USB chain. This can simplify the programming for the two cameras enormously. (The easiest way to find out your cameras' serial numbers is to plug them in one at time and execute an `sbigScanUSB` command.)

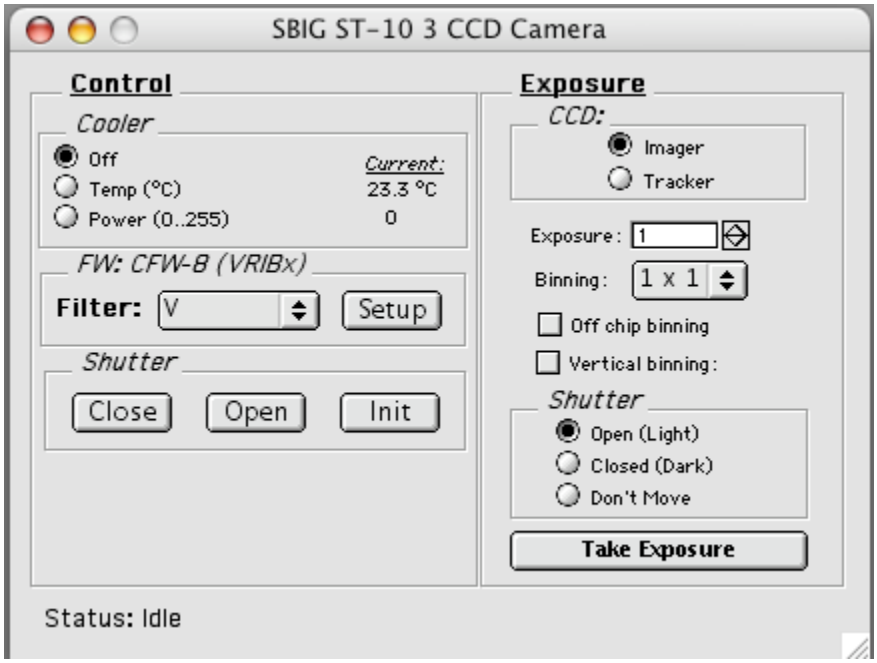
9 Sample Code

In this section we present several examples of using this package for various tasks. The goal was not to develop full-blown applications, but to give examples of programming with this package, and to hint at the range of possibilities available.

Point and Click Control Panel

For people who'd like a simple point-and-click interface to control the camera, the included file "SbigControlPanal.pxt" provides one. Simply plug in a camera, open the file, and select "Connect to Camera" from the Macro menu. A panel will pop up that allows you to control most functions of the

camera in a relatively simple way. (See the figure below.) The left side of the control panel allows you to control the cooler, filter wheel, and shutter. The right side is for taking a picture with the camera. Along the bottom is a status line that tells you what the panel is doing. You can examine the code in this file to see the camera commands in action.



Taking a Standard Light-Dark Image

The following function will take a 1.5-second light (i.e. open shutter) exposure followed by a 1.5-second dark (closed shutter) exposure, read the both out in 2x2 binned mode, and place the difference in the array `img`. The dark exposure will be left in the wave `dark`.

```
make/o/n=(2,2) img,dark
variable exposureStatus=0

sbigStartExposure 0,1.5,1    // begin light exposure
do
    exposureStatus = sbigExposureStatus(0) & 3
while (exposureStatus != 3)
sbigEndExposure 0
sbigReadout 0,1,img

sbigStartExposure 0,1.5,2    // begin dark exposure
do
    exposureStatus = sbigExposureStatus(0) & 3
while (exposureStatus != 3)
sbigEndExposure 0
sbigReadout 0,1,dark

img -= dark
```

Quick Readout of Sub-Array

Suppose you are at a telescope and have taken an image of a star field which is definitely out of focus. One technique used to get a rough focus is to take and display (as quickly as possible) many images of a small subset of the CCD's full frame, centered around the location of a bright star. Then you can change the focus of the telescope and observe (relatively quickly) the effect on the focus. The routine below will do that. Its arguments are the location of the star you want to monitor (xStar, yStar in pixels on the image), the size of the box (also in pixels), the exposure time for each image (in seconds), and the viewing magnification (i.e. screen pixels per image pixel). Once started, it will take exposures as fast as possible and read them out to a display window. The program will repeat until you abort out of it by pressing command-period, at which point it will print out some speed statistics so that you can see how bad the overhead is.

```
Function Focus(xStar,yStar,size,exposure,magnify)
    variable xStar,yStar,size,exposure.magnify
    variable nImages=0
    variable startTime=0
    variable endTime=0

    variable xStart = xStar-trunc(size/2)
    variable yStart = yStar-trunc(size/2)

    DoWindow/k Focus0
    Display/n=Focus0 as "Focus"
    Make/o/w/u/n=(size,size) focusImage
    focusImage=x+y
    AppendImage/L/T focusImage
    SetAxis/A/R left
    ModifyGraph width={perUnit,magnify,top},height={perUnit,magnify,left}
    ModifyGraph margin=-1
    DoUpdate
    try
        sbigSetMisc/S=1
        sbigStartExposure/F 0,exposure,0
        startTime=ticks
        do
            if ((sbigExposureStatus(0)&3)==3)
                sbigEndExposure/F 0
                sbigReadout/I 0,0,focusImage,xStart,yStart,size,size
                sbigStartExposure/F 0,exposure,0
                ModifyImage focusImage ctab= {*,*,Grays,0}
                DoUpdate
                nImages+=1
            endif
        while (1)
    catch
        endTime=(ticks-starttime)/60.15
        sbigEndExposure/F 0
        sbigSetMisc/S=2
        print nImages,"images in ",endTime,"seconds."
        print "   ",nImages/endTime,"images/sec"
        print "   ",endTime/nImages,"sec/image"
        return (V_AbortCode==-1) ? 0 : V_AbortCode
    endtry
end
```

There are several interesting techniques used for this function. First is the use of the try-catch-endtry structure to keep a loop going until the user breaks out of it with command-period. We also avoid the shutter motion timing by opening the shutter at the beginning of the display (`sbigSetMisc/S=1`), leaving it open for all the exposures, and only closing it when we break out of the loop (`sbigSetMisc/S=2`). Also, we read out the image in raw (`sbigReadout/I` - i.e. unsigned

word) format to avoid the conversion to single precision, and don't subtract any dark image. Finally, we use the `/F` flag on both `sbigStartExposure` and `sbigEndExposure` to skip the timing delays associated with powering down the readout amplifier and waiting for the shutter motor to turn off. In spite of all this, the image update rate is still pretty slow due to readout time and software overhead. (On my machine, an ancient 450MHz G4, the overhead typically adds more than a second to the time per image!) Nonetheless, this is still much easier than running back an forth between the telescope and computer to retake an image every time you adjust the focus.

Testing the Camera's Cooler

The following code will track the cooling behavior of your camera. It measures and graphs the temperature of the CCD at one-second intervals for 10 minutes. For a proper test, the camera should be on but the cooler off long enough for the CCD to come to room temperature. Then start the program. For the first 30 seconds, the cooler remains off. Then it is turned on, at full power, for eight minutes. Finally, it is turned off and the temperature is tracked for another minute and a half. The code assumes an active connection to a camera on `sbig0`.

First, enter the following function into the Procedure window. This is the function that will actually gather the temperature data.

```
function GetTemp()
    NVAR NextTemp = NextTemp
    WAVE Temps = Temps

    if (NextTemp<dimSize(Temps,0))
        Temps[NextTemp] = sbigTccd(0)
        NextTemp +=1
        if (NextTemp==30)
            print "Turning cooler on at n=",NextTemp
            sbigSetCooler 2,255
        endif
        if (NextTemp==510)
            print "Turning cooler off at n=",NextTemp
            sbigSetCooler 0
        endif
        return 0
    else
        Print "Done."
        return 1
    endif
end
```

Now we define the data and set the `GetTemp()` function as a background task, executed every 60 ticks. From the command line, type:

```
Make/n=600 Temps=0
Display Temps
variable NextTemp=0
SetBackground GetTemp()
CtrlBackground period=60,start
```

Now you can sit back and watch the temperature drop and rise again. After `GetTemp()` says it's done, to be safe, you should type one more command:

```
KillBackground
```

Security Camera

This example mimics a security camera system. It assumes that four SBIG ST-10 cameras have been hooked up. The ST-10's will be read out in 3x3 binning mode, and displayed at half-scale to show all four images in a single window. (The display window has been sized to fit the ST-10's 3x3 binned image size.) The camera control is actually quite easy: start an exposure on all four cameras and whenever a camera finishes, read it out and start another exposure. The most difficult part is getting the display window set up properly. Here is the window recreation macro. It looks long, but it is really just setting up four sub-windows, one for each image, along with some labels for the cameras.

```
Window SecurityWindow() : Graph
    PauseUpdate; Silent 1 // building window...
    Display /W=(6,50,734,540) as "Cameras"
    TextBox/C/N=text0/F=0/A=LT/X=1.00/Y=1.00 "Camera 0"
    TextBox/C/N=text0_1/F=0/A=LT/X=51.00/Y=1.00 "Camera 1"
    TextBox/C/N=text0_2/F=0/A=LT/X=1.00/Y=51.00 "Camera 2"
    TextBox/C/N=text0_3/F=0/A=LT/X=51.00/Y=51.00 "Camera 3"
// set up sub-window for first image:
    Display/W=(0,0,365,245)/HOST=#
    AppendImage/T img0
    ModifyImage img0 ctab= {*,*,Grays,0}
    ModifyGraph margin(left)=-1,margin(bottom)=-1,margin(top)=-1,margin(right)=-1
    ModifyGraph mirror=0, nticks=0, noLabel=2, standoff=0, axThick=0
    SetAxis/A/R left
    RenameWindow #,C0
    SetActiveSubwindow ##
// set up sub-window for second image:
    Display/W=(365,0,728,245)/HOST=#
    AppendImage/T img1
    ModifyImage img1 ctab= {*,*,Grays,0}
    ModifyGraph margin(left)=-1,margin(bottom)=-1,margin(top)=-1,margin(right)=-1
    ModifyGraph mirror=0, nticks=0, noLabel=2, standoff=0, axThick=0
    SetAxis/A/R left
    RenameWindow #,C1
    SetActiveSubwindow ##
// set up sub-window for third image:
    Display/W=(0,245,365,490)/HOST=#
    AppendImage/T img2
    ModifyImage img2 ctab= {*,*,Grays,0}
    ModifyGraph margin(left)=-1,margin(bottom)=-1,margin(top)=-1,margin(right)=-1
    ModifyGraph mirror=0, nticks=0, noLabel=2, standoff=0, axThick=0
    SetAxis/A/R left
    RenameWindow #,C2
    SetActiveSubwindow ##
// set up sub-window for fourth image:
    Display/W=(365,245,728,490)/HOST=#
    AppendImage/T img3
    ModifyImage img3 ctab= {*,*,Grays,0}
    ModifyGraph margin(left)=-1,margin(bottom)=-1,margin(top)=-1,margin(right)=-1
    ModifyGraph mirror=0, nticks=0, noLabel=2, standoff=0, axThick=0
    SetAxis/A/R left
    RenameWindow #,C3
    SetActiveSubwindow ##
EndMacro
```

Finally, here is the code to control the cameras. It is written as a macro since the update speed is primarily driven by the image downloads (i.e., compiling it as a function won't really speed it up significantly.) It starts by defining all the image arrays. Note that each image takes *two* waves: one for the final image and one to store the dark frame. It then puts up the display window and takes a dark frame for each of the four cameras. Finally, it starts light exposures on each camera and enters an infinite loop. In the loop, it checks each camera in turn and, if it's exposure has finished,

downloads the new one to the proper image, subtracts the dark frame, and begins a new exposure. To save space, the image is read out to the display wave, so PauseUpdate and ResumeUpdate are used to prevent flashing of the raw readout before the dark frame is subtracted.

```
macro SecurityCamera(etime)
    variable etime
    variable done
    variable status
    silent 1
    doWindow/k SecurityWindow
    make/o/n=(root:sbig0:imagerNX[2],root:sbig0:imagerNY[2]) img0
    make/o/n=(root:sbig1:imagerNX[2],root:sbig1:imagerNY[2]) img1
    make/o/n=(root:sbig2:imagerNX[2],root:sbig2:imagerNY[2]) img2
    make/o/n=(root:sbig3:imagerNX[2],root:sbig3:imagerNY[2]) img3
    make/o/n=(root:sbig0:imagerNX[2],root:sbig0:imagerNY[2]) dark0
    make/o/n=(root:sbig1:imagerNX[2],root:sbig1:imagerNY[2]) dark1
    make/o/n=(root:sbig2:imagerNX[2],root:sbig2:imagerNY[2]) dark2
    make/o/n=(root:sbig3:imagerNX[2],root:sbig3:imagerNY[2]) dark3
    SecurityWindow()

    print "Taking dark images"
    sbigStartExposure/C=0 0,etime,2
    sbigStartExposure/C=1 0,etime,2
    sbigStartExposure/C=2 0,etime,2
    sbigStartExposure/C=3 0,etime,2
    done=0
    do
        if ((sbigExposureStatus(0)&3) == 3)
            sbigEndExposure/C=0 0
            sbigReadout 0,2,dark0
            done+=1
        endif
        if ((sbigExposureStatus(1)&3) == 3)
            ccdEndExposure/C=1 0
            ccdReadout 0,2,dark1
            done+=1
        endif
        if ((sbigExposureStatus(2)&3) == 3)
            ccdEndExposure/C=2 0
            ccdReadout 0,2,dark2
            done+=1
        endif
        if ((sbigExposureStatus(3)&3) == 3)
            ccdEndExposure/C=3 0
            ccdReadout 0,2,dark3
            done+=1
        endif
    while (done<4)

    print "Beginning Scan"
    sbigStartExposure/C=0 0,etime,1
    sbigStartExposure/C=1 0,etime,1
    sbigStartExposure/C=2 0,etime,1
    sbigStartExposure/C=3 0,etime,1
    do
        if ((sbigExposureStatus(0) & 3)==3)
            sbigEndExposure/C=0 0
            pauseUpdate
            sbigReadout 0,2,img0
            img0 -= dark0
            ModifyImage/w=Security#C0 img0 ctab={*,*,Grays,0}
            ResumeUpdate
            sbigStartExposure 0,etime,1
        endif
        if ((sbigExposureStatus(1) & 3)==3)
            sbigEndExposure/C=1 0
            pauseUpdate
            sbigReadout 0,2,img1
```

```
            img1 -= dark1
            ModifyImage/w=Security#C1 img1 ctab={*,*,Grays,0}
            ResumeUpdate
            sbigStartExposure 0,etime,1
        endif
        if ((sbigExposureStatus(2) & 3)==3)
            sbigEndExposure/C=2 0
            pauseUpdate
            sbigReadout 0,2,img2
            img2 -= dark2
            ModifyImage/w=Security#C2 img2 ctab={*,*,Grays,0}
            ResumeUpdate
            sbigStartExposure 0,etime,1
        endif
        if ((sbigExposureStatus(3) & 3)==3)
            sbigEndExposure/C=3 0
            pauseUpdate
            sbigReadout 0,2,img3
            img3-=dark3
            ModifyImage/w=Security#C3 img3 ctab={*,*,Grays,0}
            ResumeUpdate
            sbigStartExposure 0,etime,1
        endif
    while (1)
end
```

10 Details of Individual Camera Operations and Functions

sbigScanUSB

Flags:

/Q Don't print camera information to the history area.

This function scans the USB bus for SBIG cameras and reports its results via two string variables `root:sbig_Cameras` and `root:sbig_SerialNumbers`, which it creates if necessary. The string `sbig_Cameras` contains a semi-colon separated list of the model names returned by any cameras found. The string `sbig_SerialNumbers` contains a semi-colon separated list of the corresponding serial numbers. You can find out the USB port to which a specific camera is connected by using Igor Pro's `WhichListItem` function to find that camera's serial number, e.g.

```
n = WhichListItem("...serial number...",sbig_SerialNumbers)
```

The index returned is the USB port number you need for the sbigConnect operation to link to that specific camera. The simplest way to find out your camera's serial number is to run sbigScanUSB without the /Q flag. It will print all the found serial numbers to the history.

NOTE: Due to a limitation of the SBIG camera driver, this operation will only return correct results if no cameras have been linked to by `sbigConnect`. So this routine should be called *after* the cameras have been plugged in to the USB bus, but *before* any cameras are actually connected via `sbigConnect`. Otherwise, the scan may return inaccurate results.

sbigConnect [/Q] [/USB=n] [/E={a,b,c,d}]

Connect to a specified camera.

Flags:

/Q	Don't print camera information to the history ares
/USB=n	Connect to camera on USB slot n. (See sbigScanUSB below.)
/E={a,b,c,d}	Connect to camera on SBIG Ethernet-to-Parallel box at IP a.b.c.d

If neither the `/USB` or `/E` options are specified, `sbigConnect` will attempt to connect to the first camera it finds on the USB bus. (Using this command with no flags is, in fact, the easiest way to connect if you only have one USB camera.)

After making the connection, the camera data are read out and transfered to `Igor Pro` in a data subdirectory, creating the subdirectory if necessary. For the first camera connected, data will be stored in directory "`root:sbig0:`". Subsequent connections will put each new camera's data sequentially in directories "`root:sbig1:`" up to "`root:sbig3:`". SBIG's driver can support simultaneous connections to at most four cameras. See Table 1 for a list of the data stored there.

sbigDisconnect [/Q] [/C=n] [/A]

Call this when you are completely done with a camera. You can disconnect from specific cameras with the `/C=n` flag, or from all of them with the `/A` flag. The `/A` flag is optional and is assumed if the `/C=n` flag is not used.

Flags:

/Q	Don't print results to the history area.
/C=n	Disconnect from camera n
/A	Disconnect from all cameras

NOTE: If you disconnect one camera, and others are still connected, the XOP will automatically make the lowest number camera still connected become the default for subsequent operations.

sbigSelectCamera/C=n

Specify the default camera to be used if the `/C=n` flag is omitted in other operations. The parameter n must be in the range 0...3, corresponding to data folders 'root:sbig0:' through 'root:sbig3:', and must correspond to a camera to which a successful connection has been made using the sbigConnect operation. Subsequent use of any `/C=n` flag in any future operation will replace this default setting with the new value. Note that when more that one camera is being used, it is probably a good idea to use the `/C=n` flag for *every* operation.

sbigSetFilter [/C=n] *filterposition*

Move the filter wheel [of camera 'n'] to position *filterposition*. Normally, the position should be between 1 and the maximum position given in the camera variable `cfwPositions`. However, if $filterposition = 0$, the filterwheel will calibrate itself and return to a default position (position 1). The motion of the filter wheel can be monitored with the function `sbigFWStatus()` (see below).

sbigSetCooler [/C=n] *mode* [,*value*]

Set the mode of the thermoelectric cooler [on camera 'n']. The mode (and corresponding values) are:

0: Off (no cooling)
1: Regulate the temperature at *value* in °C
2: Constant power level. *Value* = 0...255 for off to full-power.
3: Freeze cooler
4: Unfreeze cooler
5: Enable autofreeze
6: Disable autofreeze

"Freezing" the cooler temporarily disables temperature regulation and maintains a steady power level. This may lessen the readout noise in the image. You should freeze the cooler right before calling sbigReadout, and unfreeze it immediately afterwards. The autofreeze mode will cause sbigReadout to do this for you.

sbigStartExposure [/C=n] [/M] [/F] [/T] *ccd, time, shutter* [,*abg*]

Begin exposure [on camera 'n']. Shutter mode 0 (don't move) is mostly useful for performing an exposure on the tracking CCD while the main imaging CCD is performing a long exposure. Shutter modes 1 and 2 are for taking light and dark frames respectively.

/C=n	Select camera 'n' (0...3) and start exposure.
/M	Millisecond mode - only works if camera has an electronic shutter.
/F	Fast: don't power down the readout amplifier (see below)
/T	External triggered mode. See below.
ccd	0: imager, 1: tracker, 2: remote guider.
time	exposure time in seconds. (Resolution: 0.01 sec.) with /M flag: exposure time from 1–255 ms.
shutter	0: don't move, 1: open for exposure, close for readout (light frame), 2: closed for both exposure and readout (dark frame).
abg	0: turn off anti-blooming gate. 1: Clock ABG at low rate 2: Clock ABG at medium rate 3: Clock ABG at high rate.

The /M flag specifies millisecond time resolution. When /M is used, the time parameter should only range from 1–255 milliseconds. This option is only available if the CCD has an electronic shutter. To test this, type '`print root:sbig0:imagerCapabilities & 2`' after connecting to the camera. If the result is 2, your camera has an electronic shutter.

The /F flag tells the driver to skip the power-down of the readout amplifier. Normally, this amplifier is powered down during an exposure to prevent its heat from causing a "glow" in the upper left-hand corner of the imager. Skipping this slow power-down will result in starting the exposure sooner, but at the cost of the aforementioned glow. It is mostly useful when very fast frame rates are needed, for example when trying to obtain a good focus.

The /T flag tells the driver to wait for an external electronic trigger before starting the specified exposure. See your camera manual for directions on connecting to an external trigger.

sbigEndExposure [/C=n] [/F] [*ccd*]

End exposure [on camera 'n'] on specified CCD, and prepare it for readout. If *ccd* is omitted, the imager CCD is assumed. This command is also useful to abort a long exposure in progress without waiting for it to complete.

/C=n	Select camera 'n' (0...3) first.
/F	Fast mode: don't wait for the shutter motor to turn off before beginning the readout.
ccd	0: imager, 1: tracker, 2: remote guider.

The /F flag will skip the delay associated with turning the shutter motor off. This will decrease the readout time, but should only be done if the shutter was not moved for the exposure. For example, if you issued a `sbigStartExposure` command with a shutter mode of 0 (leave shutter alone) for a light frame, or with a shutter mode of 2 (closed for exposure and readout) for a dark frame. This scenario is most useful with you are using the tracking CCD while the imaging CCD is taking a long exposure.

sbigReadout [/C=n] [/I] *ccd, mode, wave*[*, top, left, height, width*]

Read out completed exposure data [from camera 'n'] to a wave. The wave is normally coerced to single precision and redimensioned to fit the size of the image being read. If the `/I` flag is used, the wave will be a 16-bit unsigned integer instead.

Flags:	
/C=n	Read image from camera n
/I	Leave data in raw (16-bit unsigned integer) format
Parameters:	
ccd	0: imager, 1: tracker, 2: remote guider.
mode	readout mode (pixel binning). Note: for vertical binning (modes 3-5), add 256×(# pixels to bin) to the mode number.
wave	`Igor Pro` wave in which to store the data. This wave will be set to 16-bit, unsigned, and redimensioned to fit the output image size.
top	first row to read for partial-frame readout
left	first column to read for partial-frame
height	number of rows in partial frame
width	number of columns in partial frame

sbigSetMisc [/C=n] [/F=n] [/L=n] [/S=n]

Sets the state of miscellaneous devices in the camera. If any flags are omitted when executing this command, the corresponding device's state will be left unchanged.

If the /C=n flag is set with n=0..3, that camera will be selected first. Otherwise, the command will go to the currently selected camera.

The /F flag controls the camera's fan. Use /F=0 to turn the fan off, and /F=1 to turn it on. Note that is is inadvisable to turn the fan off for very long.

The /L command controls the LED on the back of the camera. It accepts the following values:

0: Turn LED off
1: Turn LED on
2: Slow blink
3: Fast blink

The /S flag commands the mechanical shutter. The shutter codes are:

0: leave alone (don't move)
1: open shutter
2: close shutter
3: re-initialize shutter
4: open Remote Tracking Head's shutter
5: close Remote Tracking Head's shutter

Normally, a remote head's shutter only opens and closes together with the camera's internal shutter during a regular exposure. For this command, however, codes 1 and & 2 open and close *only* the internal shutter, and codes 4 & 5 open and close only the external head's shutter. This command is the only way to open or close only one of the shutters.

sbigAOCenter [/C=n]

Move the mirror of the adaptive optics (AO) unit [on camera 'n'] to the center position.

sbigAOTipTilt [/C=n] *xposition, yposition*

Move the mirror of the adaptive optics (AO) unit [on camera 'n'] to the specified position. The nominal range for both positions is from 0...4095. The center position is at (2048, 2048).

sbigActivateRelay [/C=n] *txPlus, txMinus, tyPlus, tyMinus*

Activate the four relays (X+, X-, Y+, Y-) [of camera 'n'] for the time specified. All times are in hundredths of a second. Calling with all four parameters zero will cancel any relay activation in progress.

sbigFWStatus(n)

Usage: (variable) status = sbigFWStatus(n)

Return the current status of the filter wheel. Set variable n=0..3 to specify a specfic camera, or set n=-1 to specify the last selected camera. Regardless of the value of n, this function does not change the currently selected camera. The values returned are:

Returns:

0: unknown. Generally a filter wheel error.
1: wheel is stationary at filter selected.
2: wheel is moving

sbigExposureStatus(n)

Usage: (variable) status = sbigExposureStatus(n)

Return the current status of exposures for the imager and tracker CCD's. Set parameter n=0..3 to specify a specfic camera, or set n=-1 to specify the last selected camera. Regardless of the value of n, this function does not change the currently selected camera. The returned result is the SUM of values for the imager and tracker:

0:	imager and tracker (or remote head) are idle
2:	imager is in a timed exposure
3:	imager exposure is complete and awaiting readout
8:	tracker (or remote head) is exposing
12:	tracker (or remote head) exposure is complete.
32768:	Camera is waiting for an external trigger to begin an exposure.

Unfortunately, in an attempt at backward compatibility, SBIG's driver puts the status of any remote guide head's CCD (*e.g.* attached to an ST-L camera) on the same bits as the tracker CCD. This will lead to status confusion if you attempt to overlap exposures on both the tracker and remote guider CCD's. You should only take exposures on one of them at a time.

sbigRelayStatus(n)

Usage: (variable) status = sbigRelayStatus(n)

Return the current status of a camera's relays. Set variable n=0..3 to specify a specfic camera, or set n=-1 to specify the last selected camera.Regardless of the value of n, this function does not change the currently selected camera. The result returned is the SUM of the following values:

8:	$+X$ relay active
4:	$-X$ relay active
2:	$+Y$ relay active
1:	$-Y$ relay active

sbigTccd(n)

Usage: (variable) temperature = sbigTccd(n)

Return the current temperature of a camera's CCD, in Celsius degrees. Set variable n=0..3 to specify a specfic camera, or set n=-1 to specify the last selected camera. Regardless of the value of n, this function does not change the currently selected camera.

This function is provided because `Igor Pro` does not update the idler variables unless it isn't doing anything else. If you're running a long program and need the current temperature during that program, this routine is for you. Note, however, that this function does NOT cause any idler variables, including the specified camera's "Tccd" to be updated. If you want the idler variables to be updated, use Igor's DoXOPIdle command instead.

Table 1: *This table lists the data (waves, variables, and strings) which are created after a camera has been successfully attached with the sbigConnect command. The data for the first camera connected will be stored in the directory "root:sbig0:". Connections for up to three additional cameras will have their data stored in directories "root:sbig1:" to "root:sbig3:", as they are attached with sbigConnect. Waves marked with an asterisk (*) are idler variables. They are updated approximately every two seconds while* `Igor Pro` *is idle.*

Waves for the Imaging CCD:	
badColumn	list of bad columns in imager
imagerDX	pixel X-sizes in μm for each readout mode
imagerDY	pixel Y-sizes in μm for each readout mode
imagerGain	gain (in electrons/ADU) for each readout mode
imagerMode	mode number for each readout mode
imagerNX	number of x-pixels for each readout mode
imagerNY	number of y-pixels for each readout mode
Waves for the Tracking/Remote CCD:	
trackerDX	pixel X-sizes in μm for each readout mode
trackerDY	pixel Y-sizes in μm for each readout mode
trackerGain	gain (in electrons/ADU) for each readout mode
trackerMode	mode number for each readout mode
trackerNX	number of x-pixels for each readout mode
trackerNY	number of y-pixels for each readout mode
Variables:	
badColumns	number of bad columns in imager
cameraFirmwareVersion	camera ROM version number
cameraHandle	used to select camera
cameraType	number specifying camera model
cfwModel	number specifying filter wheel model
cfwPositions	number of filter positions in filter wheel
coolerAutoFreeze*	=1 if autofreeze enabled
coolerFrozen*	=1 if cooler frozen at last update
coolerPower*	current power level (0–255)
coolerRegulating*	=1 if cooler regulating temperature at last update
coolerSetPoint*	cooler set point ($^\circ C$)
driverVersion	version number of the camera driver software
hasABG	=1 if imager has an anti-blooming gate
imagerCapabilities†	bits describe capabilities of imager CCD
maxRequests	maximum number of requests driver can support
readoutModes	number of readout modes for imaging CCD
Tamb*	ambient temperature monitor (only on parallel-port cameras)
Tccd*	temperature (in $^\circ C$) of imager CCD
trackerCapabilities†	bits describe capabilities of tracker CCD
trackerReadoutModes	number of readout modes for tracker CCD
usbDriverVersion	version number of USB driver
usbLoaderVersion	version number of USB loader
Strings:	
cameraName	camera model
driverName	camera driver name
serialNumber	serial number of camera
usbDriverName	USB driver name
usbLoaderName	USB loader name

† The values of 'imagerCapabilities' and 'trackerCapabilites' are the sum of:
+ 1 if CCD is a frame transfer device
+ 2 if CCD has an electronic shutter
+ 4 if CCD has electronics to support a remote guide head
+ 8 if CCD has BTDL support.

Table 2: *Quick Guide to the Operations and Functions*

Operations:

sbigScanUSB [/Q] Scan USB bus and print camera names and serial numbers to history area.

sbigConnect [/Q] [/USB=*n*] [/E=*{a,b,c,d}*] Connect to camera

sbigDisconnect [/Q] [/A] [/C=*n*] Disconnect from camera

sbigSelectCamera [/C=*n*] Select camera *n* for future commands.

sbigSetFilter [/C=*n*] *filternum* Move filterwheel [on camera *n*] to position *filternum*

sbigSetCooler [/C=*n*] *mode* **[,** *value***]** Set mode of thermoelectric cooler [on camera *n*]

sbigStartExposure [/C=*n*] [/M] [/F] [/T] *ccd, time, shutter* **[,***abg***]** Start exposure

sbigEndExposure [/C=*n*] [/F] [*ccd***]** Stop exposure

sbigReadout [/C=*n*] [/I] *ccd, mode, wave* **[,***top, left, height, width***]** Read out image to wave

sbigSetMisc [/C=*n*] [/F=*n*] [/L=*n*] [/S=*n*] Set the Fan, LED, and shutter [on camera *n*].

sbigAOCenter[/C=*n*] Center the adaptive optics mirror [on camera *n*]

sbigAOTipTilt [/C=*n*] *xposition, yposition* Set AO mirror position

sbigActivateRelay[/C=*n*] *txPlus, txMinus, tyPlus, tyMinus* Activate relays

Functions:

sbigFWStatus(*n***)** Returns status of filterwheel on camera *n*

sbigExposureStatus(*n***)** Returns status of exposures on imager and tracker CCDs

sbigRelayStatus(*n***)** Returns status of the four relays