CRICATS & CCDOpp: Software Manual

Sasha R. Brownsberger
Ian A. O. MacMillan

Department of Physics
Harvard University

July 2018

1 Introduction

This is the manual for the use and operation of the spectrometer built and developed by Professor Christopher Stubbs' Lab in summer 2018. The system is called the Changing Relative Intensity Continuous Auxilary Telescope Spectrometer or CRICATS for short.

2 Equipment

This is the briefest overview of the equipment used. Full documentation on the CRICATS device is not included in this manual.

2.1 Computer

For this software to operate properly, you will need a computer[1] with a Linux operating system, preferably Ubuntu. You will also need to install the INDI Library. Instructions on how to do that can be found at `http://indilib.org`.

To let the CCDOpp software work correctly, make sure you change the directories in the `FileMan.py` module to reflect the current directory of the "Data" folder that is inside the CCDOpp folder. Currently the directory is `/home/ian/Desktop/CCDOpp/Data/`. You will need to change lines 10, 21, and 31 in the `FileMan.py` module if you are moving the folder, renaming the folder, or using the software on a new computer.

2.2 Camera

This software is made for the SBIG STF-8300.[2] To use with other cameras, first, check the and install the camera's driver from the INDI Library. It may

[1] The password to the laptop is stubbslab
[2] http://diffractionlimited.com/product/stf-8300/

work for other cameras by, but none have been tested.

3 Operating

The operation of this camera is built on the INDI Library, which uses a virtual server that runs the driver of the camera and holds values for different properties. The code provided acts like the client, which sends new values and switches to the server. For example, to take a picture you send an exposure time to the server.

3.1 Connecting

To connect, first, connect the camera to power and plug the USB-A cord into the camera and then into the computer.[3] once you have done this open terminal and confirm that the computer sees the camera by entering the command:

```
$ lsusb
```

This lists all devices seen by the computer through the USB ports. You should see the name of the camera attached to one of the USB ports. If you do not see this, the camera is not physically connected properly.

3.2 Server Operation

First, we must start the server and connect the camera to the server which runs the driver. To do this use the command:

```
$ indiserver -v indi_sbig_ccd
```

Once you enter this, you will not be able to run any commands in this terminal window because it is running the server. Open a new terminal to continue.

The server is now running and waiting to interface with the client. To make the client aware of the camera and to let the client access the server you must run the CCD connection script by running the command

```
$ python CCDConnect.py
```

This should deal with all of the connections as well as creating a directory if one is not already created.

Almost all of the script codes included in CCDOpp start out with a connection script in case you forget to run `$ python CCDConnect.py`, but because the connection can take several seconds, the connection section will sometimes

[3] If using the laptop associated with this software, plug it into the USB port on the right side of the laptop. The other two ports on the left side had problems.

crash the other program. This could be solved by adding a four-second waiting period to ensure the camera is connected but this would tack on a waiting period to every picture taken and the camera only needs to be connected once.

3.3 Client Operation

To start taking pictures you must be in the `TakePic` directory.[4] Once you are in this directory, you will want to start the cooler program. To do this use the command

```
$ python coolon.py
```

This will start the cooler. Because the CCD's connection to the client can sometimes take up to a second, the program may fail the first time it is run. If the program fails at first, run it again. Once the CCD is connected to the client, it will not need to connect again so it all programs will run smoothly. Wait until the CCD's temp has gone down to about -13 degrees Celsius before proceeding.[5]

To take pics, open a new terminal window. This new window should still be in the same directory, so you can simply run one of the commands

```
$ python TakeLight.py
$ python TakeDark.py
$ python TakeBias.py
$ python TakeFlat.py
```

This will prompt you to enter an exposure time in seconds. Enter the time then hit enter to take a pic.

To take a number of pictures in succession, you will have to manually enter a python list of exposures in the code on line 107.

When running the `TakeBias.py` program, the computer will prompt you for how many bias images you would like to take instead of what exposure.

Once you have taken the pic it will appear in the file directory `CCDOpp\Data`. To shut everything off you can just close out of the terminal windows and unplug the CCD.

4 File management

4.1 Data Directory

The data directory is in the Data folder and is broken up by date in UT time as seen in figure 2. Each day will have all of the data taken from the CCD and

[4] For the laptop associated with this camera the directory is: `\home\ian\Desktop\CCDOpp`

[5] The camera manufacturer suggests cooling to 35 degrees below the ambient temperature but says that there is no risk of cooling the CCD below that point.

Command	Description
`$ cd`	Opens the directory named
`$ cd ..`	Leaves the directory you are in
`$ pwd`	Shows what directory you are in
`$ ls`	Shows what is in the directory is open
`$ lsusb`	Shows USB port connections
`$ indiserver -v indi_sbig_ccd`	Starts the server
`$ python coolon.py`	Runs the cooler program
`$ python TakeLight.py`	Runs the take picture program
`$ python CCDProperties.py`	Shows server communication

Table 1: A list of useful commands. For running the CCD

Prefix	Description
`IMG`	Image from CCD
`PLT`	Pdf plot of data; Normally a spectrum
`MSB`	Master bias image
`MSD`	Master dark image
`MSF`	Master flat image
`SCI`	Scientifically reduced image

Table 2: List of file prefixes.

calculations and plots from that data. Each file has a three letter prefix which tells you what is in the file. Table 2 gives the definitions of all of the prefixes.

The names for the files from the CCD always named start with their prefix then the date they were created in DD-MM-YY format then the image number for that folder. The image number is only calculated for images from the CCD so master bias, dark, and flat images do not contribute to the count.

The python scripts for taking the various types of images already puts the new image in the proper directory with the proper name.

4.2 File Manager Module

Included in CCDOpp is a file management module called FileMan.py. This module runs the file management of the whole system and is used in both the scripts for taking pictures and for the data analysis module. I will go through each function included in the file management system and explain its purpose.

Here it is important to note that the default file used is the last image taken and the default directory (dir) is today's directory.

4.2.1 `makeDir()`

When run, this function makes the days directory in the correct Data folder. The directory name is in the format of DD-MM-YY and is made through UT

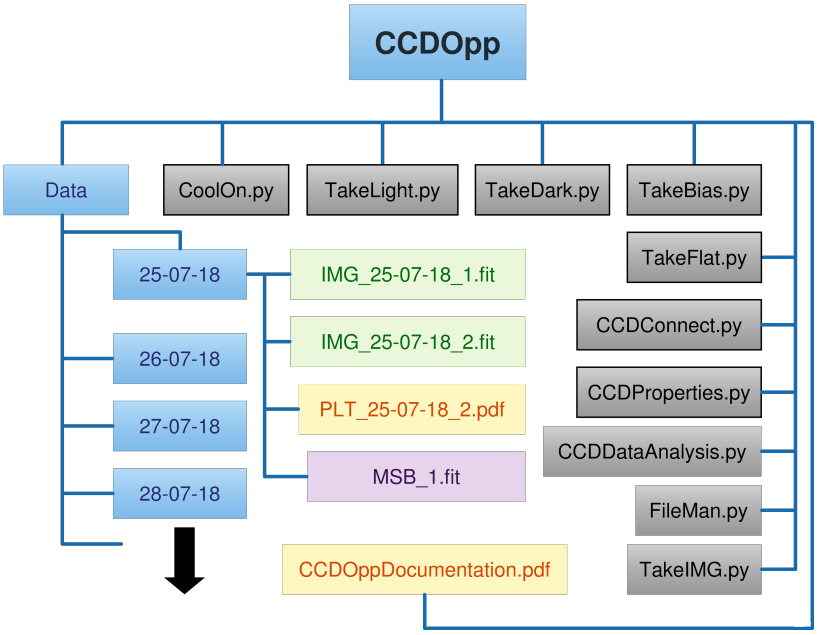

Figure 1: The directory map of CCDOpp. The data is in the Data folder and is orginized by day in UT time. The python files with black boarders are script and the files with no boarders are modules. Once the modules are compiled a `.pyc` file with the same name will appear. This is a compiled python file and is normal.

time. This means that data can be saved in two folders depending on what day it is in UT time.

The `makeDir()` function is made to run on its own when there is no directory for the day made. If the function is called and there is already a directory for the day the function will just display "Directory already exists" and return the path to the directory.

4.2.2 `saveFile(data, dir, fname)`

Takes in any text data, normally CCD data, and saves it as `fname` to directory `dir`.

4.2.3 `getDir()`

This function is the more commonly used version of `makeDir()`. It functionally does the same thing as `makeDir()` but prints out different information. `getDir()` is the function that should be used when you believe a directory already exists to avoid unneeded output from the client.

4.2.4 newFName()

This function returns the next image name that should be used for images. It counts the number of images in the current directory that have the filename format of IMG then returns the next available image name in the form of a string.

4.2.5 listFile(dir)

When given a directory in the form of a string, this function returns a list of all the files in that directory with their paths as strings.

For example, if our files looked like they do in the directory in 2 and we ran `listFile("/home/ian/Desktop/CCDOpp/Data/25-07-18")`, then we would get the result:
`['/home/ian/Desktop/CCDOpp/Data/25-07-18/IMG_25-07-18_1.fits', '/home/ian/Desktop/CCDOpp/Data/25-07-18/IMG_25-07-18_2.fits', '/home/ian/Desktop/CCDOpp/Data/25-07-18/PLT_25-07-18_2.pdf', '/home/ian/Desktop/CCDOpp/Data/25-07-18/MSB_1£its']`

4.2.6 listFileMD(dir)

This operates the same way that `listFile(dir)` does but instead of returning the full path to all files it simply returns their names the command `listFileMD("/home/ian/Desktop/CCDOpp/Data/25-07-18")` I would get the result:
`['IMG_25-07-18_1.fits','IMG_25-07-18_2.fits', 'PLT_25-07-18_2.pdf','MSB_1£its']`

4.2.7 listFNameData(list_fname)

given a list of file paths, like is the result from the function `listFile(dir)`, this function will return a list of the data matrices from the images in the list. Note that this function is not made to take files that are not fits files.

4.2.8 listDirData(dir)

This simply returns the data of an entered directory as a combination of the functions `listFile(dir)` and `listFNameData(list_fname)`.

4.2.9 listDarks(exp, dir)

This function is especially useful for making master dark images. When given an exposure time (`exp`) and a directory (`dir`), this function will return a list of the paths to darks in that directory with the entered exposure time. The returned list is similar in structure to the output of `listFile(dir)`.

4.2.10 `lastF(n=1)`

This is a very useful and widely used function. When run without a value of `n`, this function returns the file name of the last image taken that day. The input `n` represents the number of files back you would like. For example, if you wanted the image before the last image you took, you would enter a `n` of two.

4.2.11 `addKeyword(id,val,file=lastf(),dir=getDir())`

This function takes in a directory and file name of a .fits file and changes the keyword in the header with the name you entered (`id`) to the value `val`. If a keyword with the name `id` does not exist, this function adds it with the given value.

4.2.12 `getMJD()`

This function simply returns the Modified Julian Date for the current time. This function takes no arguments and when run returns a float in the form: `58325.75694`

5 Data Analysis

5.1 Data Analysis Module

We added a basic data analysis package that provides a few features that are useful in analyzing spectrometer data. Here I describe the functions and their uses.

5.1.1 `removeOverscan(image_array)`

This function takes in an array of data from a CCD and calculates where the overscan is then removes it from the array. The function then returns the modified array.

5.1.2 `readFitsFilesToData(files)`

Similar to the `listDirData(dir)` function, this function takes in a list of files with their directory in the string, and if there are multiple files given, it returns a list of lists where the sublists contain the headers in index 0 and the data array in index 1. If there is only one file, it returns that files sublist of header and data array.

5.1.3 `computeBiasStats(bias_files, dir, ...)`

This Function takes in a list of bias files and their directory and saves a file where the pixel value is the standard deviation value of that pixel in the bias frames given and one that is the median combination of all of the input files.

The function then returns a list where index 0 is the median value of all the pixels in the median master bias and index 1 is the median pixel value of the standard deviation master bias. This function takes many inputs which I have put in table 3.

This function also creates a header for the new data files. The keyword `'MADEFROM'` indicates the number of Bias images this master bias was made from

Input and Defalt Value	Description
`Bias_files`	A list of the file names for reduction
`dir =getDir()`	The directory in which the files are stored
`master_med_file ='MSB_med.fits'`	The desired name of the saved median file
`save_master_med =1`	If you want to save the master median file, you set this equal to 1. If you not, you set it as 0.
`master_std_file ='MSB_std.fits'`	The desired name of the saved standard deviation file
`save_master_std =1`	If you want to save the master standard deviation file, you set this equal to 1. If you not, you set it as 0.
`correct_overscan =1`	If you want to overscan, you set this equal to 1. If you not, you set it as 0.
`overwrite =1`	If there are already images in the directory with these names and this is set to 1 it will over write them when it saves. If you don't want it to over write them, you set it to 0

Table 3: List of the inputs to the `computeBiasStats()` function and their descriptions.

5.1.4 `computeDarkStats(dark_files, dir, ...)`

This function does the same as the `computeBiasStats()` function except with darks. Because we are creating a master dark we have the option to use the bias images to correct the read noise. By default, this function corrects for read noise. Other than correcting for read noise this function is almost identical to `computeBiasStats()` except it also returns the median pixel array and the standard deviation pixel array in the returned list after overall median and overall standard deviation. The input and default values of those inputs are seen in table 4

In this function, the keyword `'EXPTIME'` is the exposure time that the darks were taken.

Input and Defalt Value	Description
bias_files	A list of the file names for reduction
dir =getDir()	The directory in which the files are stored
master_bias_file ='MSB_med.fits'	The name of the master bias file to use for read noise reduction. It is assumed that the master bias image is in the directory given.
master_med_file ='MSD_med.fits'	The desired name of the saved median file
save_master_med =1	If you want to save the master median file, you set this equal to 1. If you not, you set it as 0.
master_std_file ='MSD_std.fits'	The desired name of the saved standard deviation file
save_master_std =1	If you want to save the master standard deviation file, you set this equal to 1. If you not, you set it as 0.
correct_overscan =1	If you want to overscan, you set this equal to 1. If you not, you set it as 0.
correct_bias =1	If you want to correct for bias, you set this equal to 1. If you not, you set it as 0.
overwrite =1	If there are already images in the directory with these names and this is set to 1 it will over write them when it saves. If you don't want it to over write them, you set it to 0

Table 4: List of the inputs to the computeDarkStats() function and their descriptions. Note some are different from the computeBiasStats() table (table 3

5.1.5 computeCameraFlatStats(flat_files, dir, ...)

This function combines all the camera flats given. Camera flats are the flats taken only through the camera and lens. computeCameraFlatStats() also corrects using a master dark or dark current scalar. This function is similar in function to the 'Stats' functions, and just like computeDarkStats(), This function returns the medians pixel array for standard deviation pixel array. Its inputs and default values are shown in table 5.

5.1.6 computeSystemFlatStats(image_files, dir, ...)

This function makes flats for the entire system in much the same way that computeCameraFlatStats does. It also assumes that all the flats were taken at the same exposure. And combines them. This function uses the flats of the whole system and returns the same four statistics returned by the

Input and Default Value	Description
image_files	A list of the file names for reduction.
dir =getDir()	The directory in which the files are stored
master_bias_file ='MSB_med.fits'	The name of the master bias file to use for read noise reduction. It is assumed that the master bias image is in the directory given.
master_dark_file ='MSD_med.fits'	The name of the master dark file to use for dark current reduction. It is assumed that the master dark image is in the directory given.
global_dark_cur =0.2	This is the global dark current in electrons per second.
master_med_file ='MSF_cs_med.fit'	The desired name of the saved median file. Note here the 'cs' stands for camera scaled.
save_master_med =1	If you want to save the master median file, you set this equal to 1. If you not, you set it as 0.
master_std_file ='MSF_cs_std.fit'	The desired name of the saved standard deviation file. Here the 'cs' also stands for camera and scaled
save_master_std =1	If you want to save the master standard deviation file, you set this equal to 1. If you not, you set it as 0.
correct_overscan =1	If you want to overscan, you set this equal to 1. If you not, you set it as 0.
correct_bias =1	If you want to correct for bias, you set this equal to 1. If you not, you set it as 0.
correct_dark_img =1	This corrects using the dark image. If you don't want it to correct using the dark them, you set it to 0.
correct_dark_scr =1	If you want to correct for the dark current using a scalar, you set this to 1 and texttttcorrect_dark_img to one. When both are set to 1, it will correct using the dark image.
overwrite =1	If there are already images in the directory with these names and this is set to 1 it will over write them when it saves. If you don't want it to over write them, you set it to 0.

Table 5: List of the inputs to the computeCameraFlatStats() function and their descriptions.

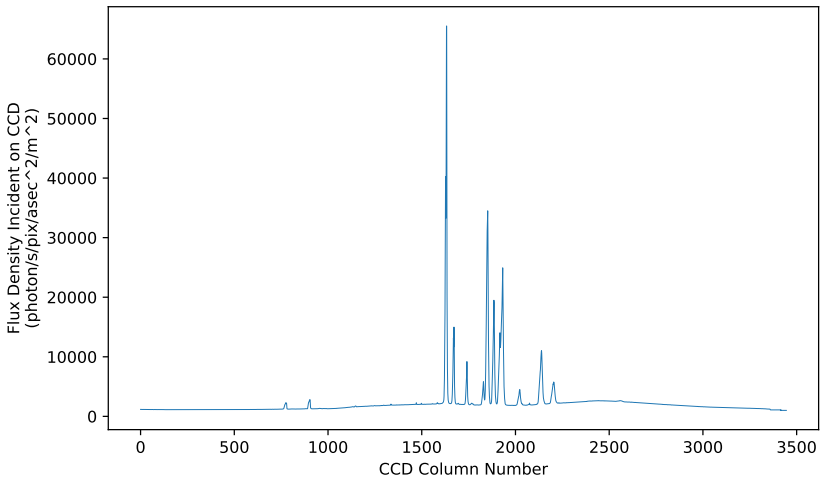

Figure 2: The sample graph of the spectrum produced from the function `plotSpectra()` from a similar spectra shown in figure 3.

`computeCameraFlatStats` function just for the entire system. It also subtracts the camera flats (if left to default settings) so you are left with the flat that is the result of only the system and not the camera and lens. The input values and their default values are shown in table 6.

5.1.7 `cleanScienceImage(image_files, dir, ...)`

This is the *pièce de résistance* of the CCDDataAnalysis module. It takes in all the files that we have made and reduces a straight up image to a scientifically reduced image. It compensates for all of the master images we have made and returns the data array of the reduced image. The input values and their default values are shown in table 7.

5.1.8 `plotSpectra(file,dir)`

This function plots the observed spectrum and also saves it to a pdf file with the prefix `PLT`. The function takes the median of a specified number of columns (`num_col` in the code) starting at a specific row (`row_to_plot`).

5.1.9 `numFulPx(file, dir, fulval)`

This simple function returns the number of pixels over a certain value. It is most commonly used by the `autoExp` function for calculating the number of pixels that are at full well capacity. This is why default value for `fulval`, which

Input and Default Value	Description
flat_files	A list of the file names for reduction.
dir =getDir()	The directory in which the files are stored
master_bias_file ='MSB_med.fits'	The name of the master bias file to use for read noise reduction. It is assumed that the master bias image is in the directory given.
master_dark_file ='MSD_med.fits'	The name of the master dark file to use for dark current reduction. It is assumed that the master dark image is in the directory given.
global_dark_cur =0.2	This is the global dark current in electrons per second.
master_med_file ='MSF_ss_med.fit'	The desired name of the saved median file. Note here the 'ss' stands for system scaled.
save_master_med =1	If you want to save the master median file, you set this equal to 1. If you not, you set it as 0.
master_std_file ='MSF_ss_std.fit'	The desired name of the saved standard deviation file. Here the 'ss' also stands for system and scaled
save_master_std =1	If you want to save the master standard deviation file, you set this equal to 1. If you not, you set it as 0.
correct_overscan =1	If you want to overscan, you set this equal to 1. If you not, you set it as 0.
correct_bias =1	If you want to correct for bias, you set this equal to 1. If you not, you set it as 0.
correct_dark_img =1	This corrects using the dark image. If you don't want it to correct using the dark them, you set it to 0.
correct_dark_scr =0	If you want to correct for the dark current using a scalar, you set this to 1 and texttttcorrect_dark_img to one. When both are set to 1, it will correct using the dark image.
correct_cam_flat =1	This is if you want to correct for the camera flat. if this is set to 1 then the function will eliminate the effects from the camera from the flat leaving you with just the system flat.
overwrite =1	If there are already images in the directory with these names and this is set to 1 it will over write them when it saves. If you don't want it to over write them, you set it to 0.

Table 6: List of the inputs to the computeSystemFlatStats function and their descriptions.

Input and Default Value	Description
image_files	A list of the file names for full scientific reduction.
dir =getDir()	The directory in which the files are stored
cleaned_filen	A list of new file names that the cleaned files can be saved to.
master_bias_file ='MSB_med.fits'	The name of the master bias file to use for read noise reduction. It is assumed that the master bias image is in the directory given.
master_dark_file ='MSD_med.fits'	The name of the master dark file to use for dark current reduction. It is assumed that the master dark image is in the directory given.
global_dark_cur =0.2	This is the global dark current in electrons per second.
master_flat_file ='MSF_ss_med.fit'	The name of the master flat file to use for flat reduction.
correct_overscan =1	If you want to overscan, you set this equal to 1. If you not, you set it as 0.
correct_bias =1	If you want to correct for bias, you set this equal to 1. If you not, you set it as 0.
correct_dark_img =1	This corrects using the dark image. If you don't want it to correct using the dark them, you set it to 0.
correct_dark_scr =0	If you want to correct for the dark current using a scalar, you set this to 1 and textttcorrect_dark_img to one. When both are set to 1, it will correct using the dark image.
correct_cam_flat =1	This will normalize over the camera flat. Set to 1 if you want to normalize or 0 if not.
overwrite =1	If there are already images in the directory with these names and this is set to 1 it will over write them when it saves. If you don't want it to over write them, you set it to 0.

Table 7: List of the inputs to the computeSystemFlatStats function and their descriptions.

Figure 3: A sample non-reduced image of a spectrum.

is the minimum value that the function will count, is set at 65520 near the full well capacity of 65535. However, this function can be used to count the number of pixels above any integer value.

5.1.10 `autoExp(file, dir)`

This is a useful function that when given the name of a .fits and its directory it will suggest an exposure time that would yield a cleaner image. The function first calculates the number of full pixels which gives the program a general idea of how blown-out the image is. If 7% of the pixels are at the max value then we consider the picture to be far too bright. This low 7% was chosen because, in our images of spectra, only the spectral lines would be blown out, which take up only a small portion of the CCD's surface. If the frame seems to be blown out then the program suggests an exposure time of half what the original image was taken at.

After determining if the spectrum is totally blown out the program looks at the value of the pixel at the 99.999th percentile pixel or the pixel whose value is only lower than about 80 other pixels. This compensates for defective pixels whose values are always high. Then the function divides the pixels value by the exposure time to get the value per second that that pixel acquired. Finally, if the function returns the suggested exposure that would saturate that pixel to 70% of full well capacity (`fillto` in the code).

The `autoExp` function works best when it is given images produced by the `cleanScienceImage` function because those images will be free from artificially high pixels and will have dark current removed. This function does not yet eliminate cosmic rays from the data. Therefore it is best to feed it images that are a median combination of three or more exposures.

5.2 Still To Do In Data Analysis

There are a few functions that I would like to add but have not been able. I have tried to list them here.

- `lineWidth()`: This function would take in a file and the limits edges of a line and would return the FWHM value, intensity, and position of that line after fitting a Gaussian to the spectral line.
- `lineFinder()`: This function would take in an image reduce it to a spectrum and then fit a polynomial to the spectrum. It would then try to identify where the spectral lines were and in conjunction with the `lineWidth()` function would return the location, intensity, and position of all of the lines along with an associated error.
- `waveMatch()`: This function would take in a spectrum represented as a list of values and return their associated wavelengths. It would do this by fitting the data to the expected lines by transforming and scaling it. return the wavelength associated with all columns in the CCD.

6 Take Image Module

While CCDOpp contains many scripts for operating the camera, I would like to develop a module to operate it. This way it would be simple to operate the CCD and run data analysis using the data analysis module together. I am immensely close to this and have made a module called `TakePic.py`, which is included in CCDOpp. Most of the functions included work as expected but there is an error in the actual `takePic` function where it fails to read out the Blob data from the CCD and thus fails.

With exception to this admittedly large hole in the `TakeIMG` module, the module works well and can be fully implemented once that one error is fixed. For this reason, I have included the module in CCDOpp and will go on to explain its functions in the same manner as I have in the previous sections.

6.1 Functions

All of these functions rely on the standard properties heald and controled by indiserver. Full documentation for these can be found at `http://indilib.org/develop/developer-manual/101-standard-properties.html`.

6.1.1 `connectServer()`

This function is pretty self explanitory. It connects the INDIServer to the module.

6.1.2 `isServerConnected()`

Simply returns a True/False value. If it comes up false, then running the function `connectServer()` will most likely fix the problem.

6.1.3 `connectCCD()`

This function provides the server with the name of the CCD and the ability to connect to it. the function ends by fully connecting to the CCD.

6.1.4 `isCCDConnected()`

This simply returns a true/false statement telling whether the CCD is connected.

6.1.5 `connect()`

This function runs the `connectServer()` and `connectCCD()` functions if needed to fully connect the CCD. this function is meant to replace the working script called `CCDConnect.py`.

Frame Type	Entered String
Light	'l','L','light', or 'Light'
Dark	'd','D','dark', or 'Dark'
Flat	'f','F','flat', or 'Flat'
Bias	'b','B','bias', or 'Bias'

Table 8: List of acceptable strings for the input of the `frameType(f)` function described in section 6.1.8.

6.1.6 `setOverscan(oscan)`

This function adds overscan to the image by telling the CCD that it is larger than it actually is. The default overscan is 30. If the overscan seems to be more or less than expected check to see if the parameters `default_height` and `default_width` are set to the un-overscanned areas in the image. I have found that the area reported for the SBIG CCD[6] are not what they are seen to be in images.

6.1.7 `takePic(exp)`

This function takes the picture with the settings that are already specified. To change parameters like overscan and frame type run those function before triggering this function. Here `exp` is the desired exposure time.

6.1.8 `frameType(f)`

This sets the frame type (light, dark, bias, flat) that you are taking. To change the type, run this function with a string from table 8 that corrisponds to the desired frame type.

6.1.9 `takeLight(exp)`

This function sets up and takes a light image with the exposure time `exp`. It does not set any overscan however so you will have to do that prior to triggering the function. This function also ensures the CCD is connected.

6.1.10 `takeDark(exp)`

This function sets up and takes a dark image with the exposure time `exp`. It does not set any overscan however so you will have to do that prior to triggering the function. This function also ensures the CCD is connected.

6.1.11 `takeBias()`

This function sets up and takes a bias image. Note bias images have an exposure time of zero so there is no entered `exp`. It does not set any overscan however

[6]The technical specs say that the frame size is 3326×2504 pixels

so you will have to do that prior to triggering the function. This function also ensures the CCD is connected.

6.1.12 `takeFlat(exp)`

This function sets up and takes a flat field image with the exposure time `exp`. It does not set any overscan however so you will have to do that prior to triggering the function. This function also ensures the CCD is connected.

6.1.13 `setTemp(temp=-15)`

This function takes in a temperature as an argument and uses it as a set point. The function then sets the CCD to cool to that temperature then waits for the temperature to reach the set point. While waiting for the function prints out the current temperature every three seconds. The CCD will continue to cool after the function has stopped running.

6.1.14 `getTemp()`

This function simply returns the current temperature of the CCD as a float.