A data reduction pipeline for SHOC: Manual

Marissa Kotze

June 26, 2013

Abstract

The Sutherland High-speed Optical Cameras (SHOC) have become the more frequently used high-speed photometry instruments on the 0.75 m, 1.0 m and 1.9 m. The one aspect that has not been addressed during their commissioning phase is the availability of on-the-fly data reduction software, similar to what is available for UCT CCD and the SAAO CCDs (STE3 and STE4). While this was not considered part of the requirements for commissioning, the users of the instruments do have the expectation for such functionality.

In the absence of more formal software, I am prepared to share the pipeline I developed for my own observations, which have been expanded to include functionality that other users may require. The pipeline does not only allow a quick-look of lightcurves which may be useful at the telescope, but also the extraction of suitable quality photometry when run in the more comprehensive mode.

The process is driven by PYTHON scripts that may be run on the servers (ASTRO/LTSP-SUTH). It facilitates the correction of the FITS headers (PyFITS), running IRAF photometry tasks (PyRAF), extraction of raw and differential lightcurves and plotting (GNUPLOT).

1 Introduction

The SHOC instruments create FITS data cubes during the observation of a kinetic series. Each extension contains only an image while the single FITS header is applicable to the entire cube itself. One may therefore either create appropriate FITS headers for each extension and reduce the cube as a whole or split the cube into individual FITS files and update their headers with the correct information for each FITS image. I have elected to do the latter.

As described in the SHOC User manual, data cubes may only be copied from the SHOC computer to the storage facility on the server once a cube has been completed. Thereafter the user may copy the data to their own /home/ directory on the server where they are free to run the data reduction pipeline provided in SHOCpipeline.tar[1] (which should be extracted there). The **README** contains instructions for running the scripts.

2 SHOCpipeline.py

This PYTHON script facilitates the following:

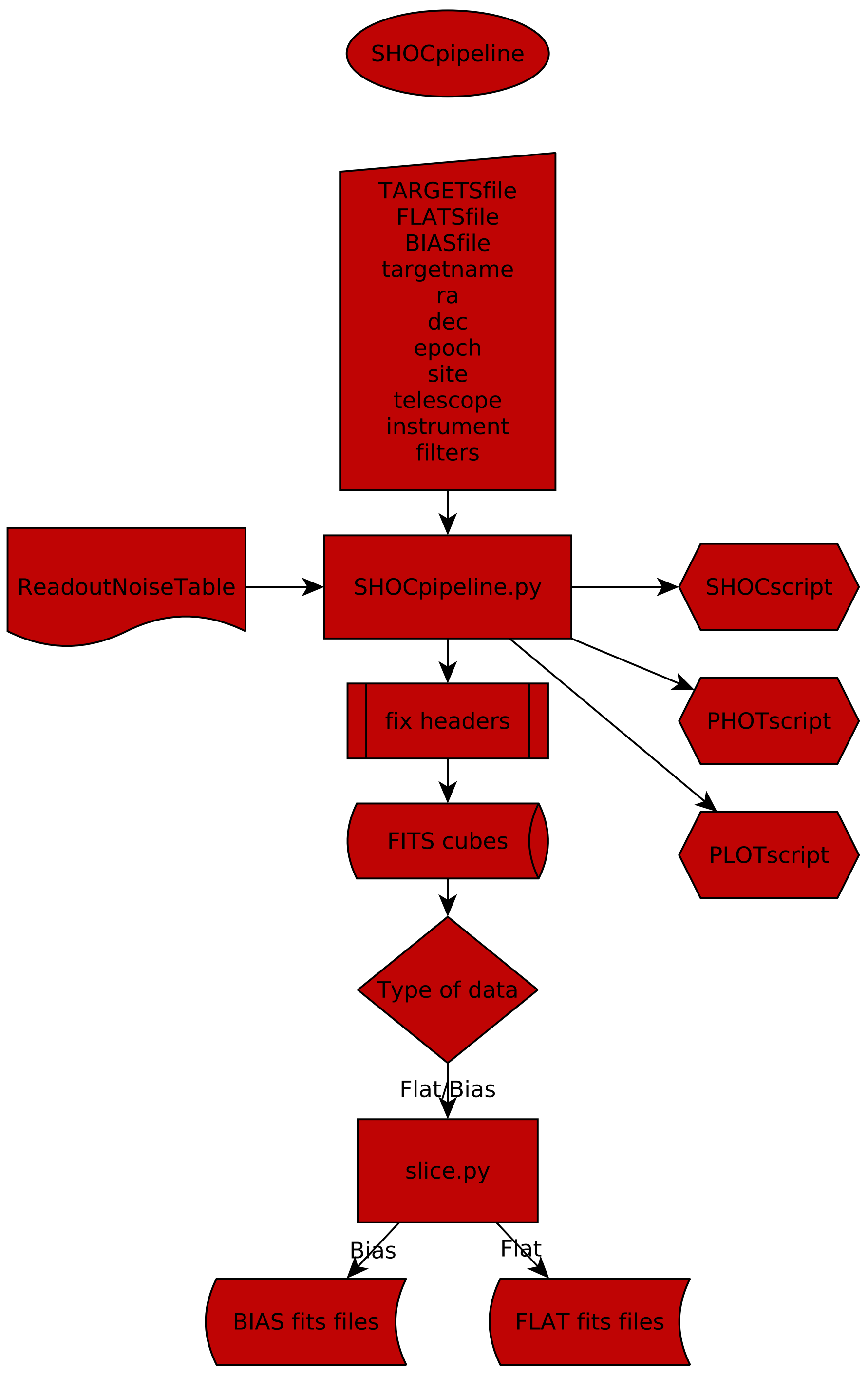
- corrects header info (target and filters) as input by the user
- populates timing headers (UT, JD and HJD)
- bias-corrections (including making master bias files)
- flat-fielding (including making master flat files)
- warns the user if any of the pre-reductions couldn't be done, but allows continuation of the process to produce a quick-look results
- finds all sources above a threshold OR uses an input file for positions
- does aperture-corrected photometry
- plots all raw extracted lightcurves

It sets up 3 LINUX scripts (SHOCscript, PHOTscript and PLOTscript) which may be rerun separately if needed. It also updates the headers FRAME, EXPOSURE, ACT, KCT and TRIGGER with the user input in the event of externally (GPS) triggered observations (TRIGGER = External). The RON (readout noise) and SENSITIVITY headers (if absent) are extracted from the instrument documentation [2]. The GAIN header for the Conventional mode is updated with the value of the SENSITIVITY header. For observations in the Electron Multiplying mode, the GAIN and RON headers are divided by the EM gain value (originally stored in GAIN). See Appendix A.

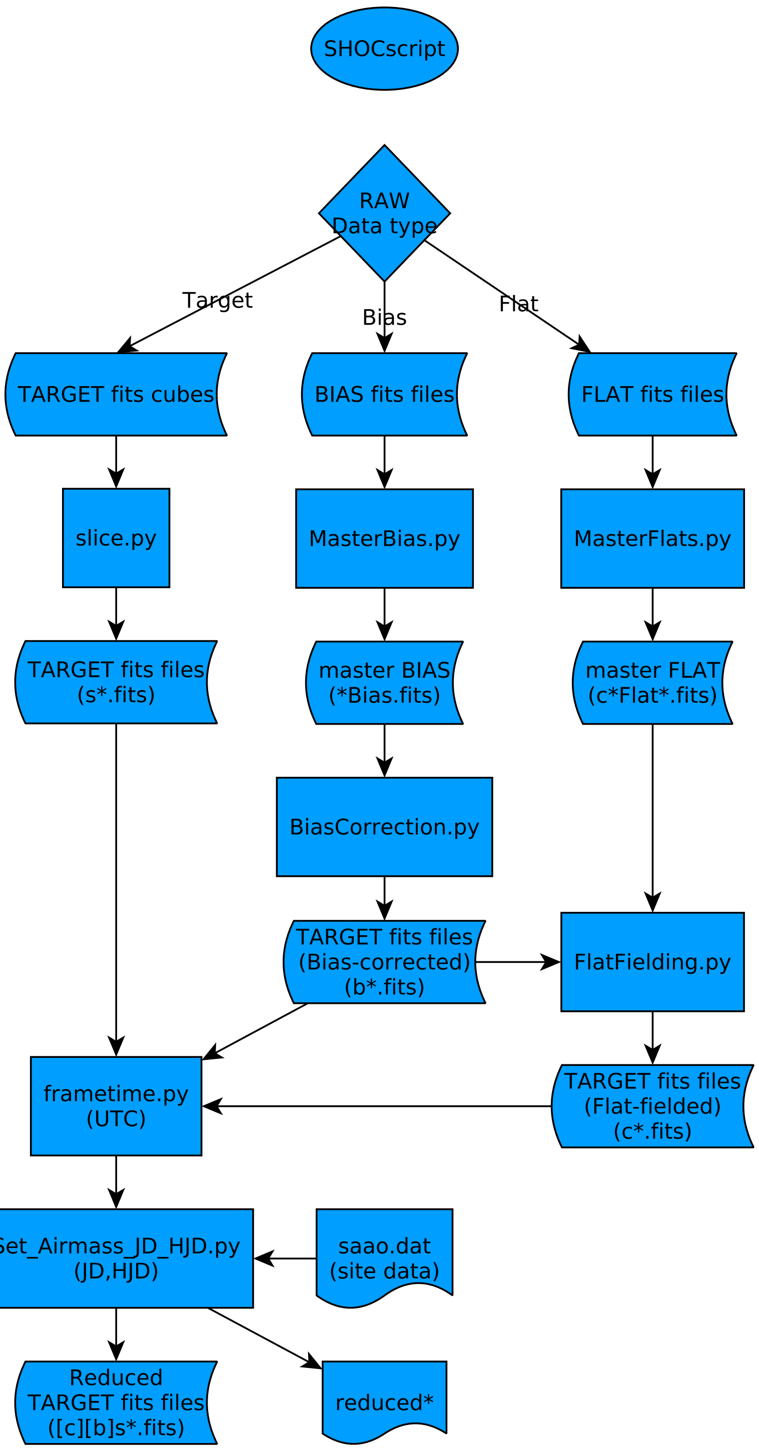
Multiple cubes can be reduced together if they were observations of the same target using the same filters. Raw bias and flat files (in the same filter) may be included in the reduction, or in their absence the master flat and bias files may be used if they are available. The user will be allowed to continue even if they are absent, but on-screen warnings will remind the user of it.

[1]http://www.saao.ac.za/∼marissa/SHOCpipeline/
[2]http://shoc.saao.ac.za/Documentation.html

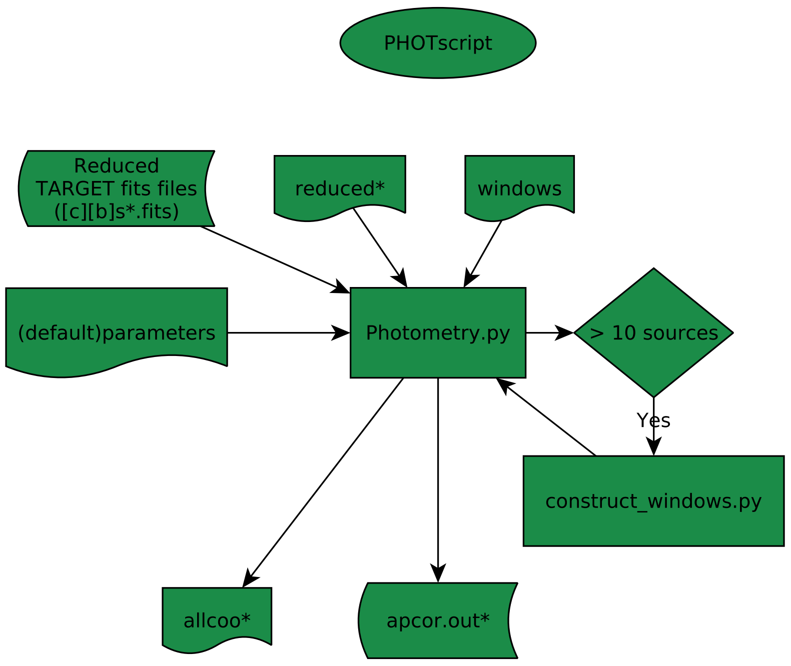
2.1 Flow Chart of SHOCpipeline.py

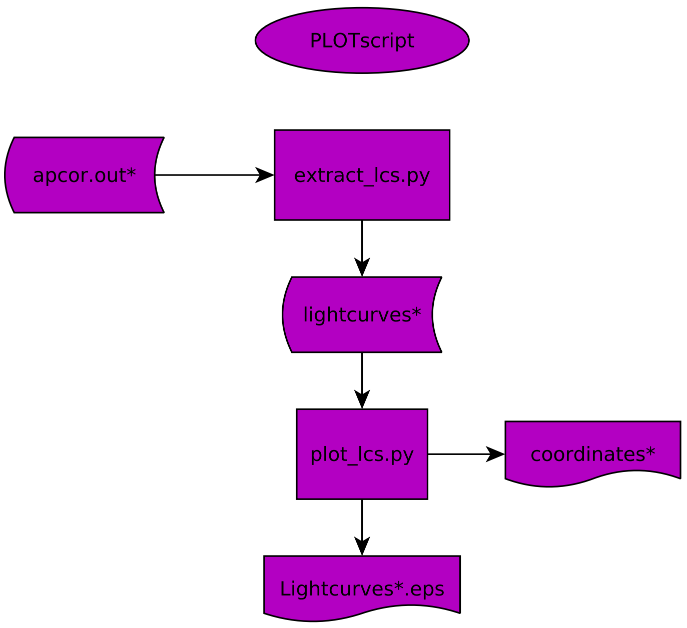


2.1.1 Flow Chart of SHOCscript



2.1.2 Flow Chart of PHOTscript and PLOTscript





2.2 SHOCscript

This LINUX script runs all the necessary pre-reduction PYTHON scripts. It splits the data cubes, populates all the timing related fits headers, makes master flats and master biases, and finally flat-fields and bias-corrects science frames. These reduced fits files are then stored in the **ReducedData** folder.

2.2.1 slice.py

This PYTHON script uses PyRAF to run the IRAF **images**, **imutil** task **imslice** to split the data cubes into the individual fits files.

2.2.2 frametime.py

This PYTHON script uses PyFITS to populate the start times of exposures (UTC) in the FITS files. For internally triggered (TRIGGER = Internal) observations the time stored in the FRAME header is the end of the first exposure in the cube. All subsequent frames follow at increments of ACT or KCT (Integration cycle time = exposure time + readout time) thereafter.

For externally triggered observations (TRIGGER = External), the first two frames in a cube are deleted. The user entered the start time of the cube (CS) and GPS triggering time (dT), so the FRAME and EXPOSURE (also ACT and KCT) fits headers could be corrected (setting TRIGGER = GPS). In general the start time for frame number n = CS + (n-2)dT.

2.2.3 Set_Airmass_JD_HJD.py

This PYTHON script uses PyRAF to run the IRAF **astutil** tasks **asthedit** and **setjd** to populate the AIRMASS, JD and HJD headers (to 8 decimals). It uses an input file **saao.dat** that contains all the site information.

2.2.4 MasterFlats.py

This PYTHON script uses PyRAF to run the IRAF **images**, **immatch**, **imutil** tasks **imcombine** to average all raw flat frames and **imarith** to normalize the master flat respectively.

2.2.5 MasterBias.py

There is no over-scan region on the SHOC CCDs, so users need to take raw bias frames if bias-corrections are required. This PYTHON script uses PyRAF to run the IRAF **images**, **immatch**, **imutil** task **imcombine** to average all raw bias frames.

2.2.6 BiasCorrection.py

This PYTHON script uses PyRAF to run the IRAF **images**, **immatch**, **imutil** task **imarith** to subtract the appropriate master bias file from all the science frames.

2.2.7 FlatFielding.py

This PYTHON script uses PyRAF to run the IRAF **images**, **immatch**, **imutil** task **imarith** to divide all the science frames by the appropriate master flat file.

2.3 PHOTscript

This LINUX script runs the PYTHON script for the photometry tasks.

2.3.1 Photometry.py

Unless a user provides a text file (to be named **windows**) which contains the initial X and Y coordinates of each source (one source per line) that should be extracted, this PYTHON script will use PyRAF to run IRAF **noao**, **digiphot**, **daophot** task **daofind** with parameters **datapars** and **findpars** to determine the positions of all sources detected above the threshold set by the user. The PYTHON script **construct_windows.py** will automatically be executed if >10 sources are detected, allowing elimination of sources close to the edge or each other.

Lightcurves may be extracted successfully from unguided and/or poorly tracked observations, where there is significant drift from initial positions. Since positions then need to be re-determined for each frame, the run-time is longer and supplying a **windows** file is irrelevant.

Instrumental magnitudes are extracted by the IRAF **noao**, **digiphot**, **daophot** task **phot** with parameters **datapars**, **centerpars**, **fitskypars** and **photpars**. Important parameters may be altered in the **parameters** input file, which initially contains default values, that has been copied to the working directory where the reductions were performed. It specifies a list of apertures to facilitate aperture-corrected photometry, but the user may specify any number of suitable apertures there, alter the inner and outer radii of the annular sky region surrounding the source, size and allowable shift of centroid box, as well as the maximum permitted magnitude error.

Aperture-corrected photometry is performed by the IRAF **noao**, **digiphot**, **photcal** task **mkapfile**, extracting the magnitudes for the smallest apertures that maximize the signal-to-noise (the turn-over in the curve of growth).

Should the curve of growth fail to converge, this PYTHON script may be rerun with the option of dumping the dataset on-screen so that the user may investigate which sources and/or apertures should be eliminated.

2.4 PLOTscript

This LINUX script runs the PYTHON scripts for extracting and plotting the raw lightcurves.

2.4.1 extract_lcs.py

This PYTHON script will extract the lightcurves for all sources that were detected on the frame that the user specifies (usually the first one). If a **windows** file exists, only lightcurves for the sources specified therein will be extracted. The maximum frame-to-frame drift is set in the **parameters** file.

2.4.2 plot_lcs.py

This PYTHON script generates GNUPLOT scripts to plot lightcurves on comparable scales. The user specifies the magnitude range for the plots and source numbers and initial coordinates are included on the plots to enable to user to identify the target and suitable comparisons immediately. The plots are saved in EPS format and the GNUPLOT scripts are also retained to enable the user to alter them manually if desired.

3 Differential Photometry

An additional PYTHON script is available to extract the differential lightcurve of the target by specifying any number of suitable comparisons.

3.1 plot_differential_lcs.py

This PYTHON script runs all the necessary scripts to produce differential lightcurves. The user specifies the number of the target and also the numbers of any number of suitable comparisons. The previous task would have displayed the extracted sources (numbers, coordinates, average magnitudes and optimal apertures), but those are also available on the plots of the raw lightcurves and in the **coordinates*** output file.

3.1.1 extractSTAR_SHOClc.py

This PYTHON script will be called for the target with all the specified comparisons. It will also be run for each comparison individually with itself and the rest of the comparisons as its comparison stars. Differential lightcurves for the target and the comparisons are then plotted by **plot_lcs.py**, so that the user may verify the suitability of the comparisons.

4 QuickLook.py

This PYTHON script allows the user to skip past the more time consuming pre-reductions such as bias-correction and flat-fielding and extract quick-look raw lightcurves to get an indication of source variability and data quality. Note that the HJD timing and airmass headers will not be populated for the QuickLook scenario, to expedite the process.

It also automatically executes SHOCscript, PHOTscript and PLOTscript. The plots of the raw lightcurves should normally suffice for a quick-look. Should the user require differential lightcurves to assess the variability of the target, the differential photometry PYTHON script (previous section) should be run after completion of this process.

Note: If the user wishes to rerun the photometry with different parameters for quick-look purposes, it is clearly not necessary to run SHOCscript again, but only PHOTsscript and PLOTscript.

5 Conclusion

The **README** included with the code explains how to execute the scripts and on-screen prompts lead the user through the process. It was therefore kept brief, while this document aims to elaborate on the functionality and working of the pipeline, without becoming overly bogged down in detail that may be better discerned from the code itself. The code has been extensively commented. IRAF's online help should be consulted regarding IRAF tasks.

On-screen comments have been added to keep the user abreast of the progress as tasks are executed. They are also useful if you want to rerun part of the process. So KEEP AN EYE ON THE SCREEN.

Sufficient quality lightcurves may be produced if parameters are chosen carefully and pre-reductions are done properly. It is assumed that the user understands the principles of photometry and that the user will redo the photometry properly (by using **SHOCpipeline.py** or other means) and not publish quick-look results generated by this pipeline.

6 Tips

Inexperienced users should run the **QuickLook.py** on single data cubes, rather than lists of cubes. Setting inappropriate parameters for the observed field will result in poor quality lightcurves and possibly a failure to perform aperture-corrected photometry. Users may specify a single aperture in the **parameters** file and may also provide a **windows** file with the X and Y coordinates of the relevant sources to speed up the process.

7 Feedback

Having used the UCT and SAAO CCD software extensively, I aimed for a similar type of solution for SHOC using PYTHON. Essentially the pipeline just serves as a way to get your IRAF tasks run automatically and visualise the results of those efforts effectively.

This is by no means the final solution, but it works well for me. I tried to also include additional functionality other users may need. But since its still under development, you should always download the latest code[3] at the start of your observing run. Let me [4] know if something is not catered for and I can see about including it. Emails reporting errors should include the error that appeared on screen and the name(s) of the relevant data cube(s).

8 Troubleshooting

- Follow the instructions in the README carefully.
- The code was developed for use on the LTSP-SUTH and ASTRO servers and are not supported on any other platform.
- PERMISSION errors: inform the IT helpdesk.
- If you have low signal-to-noise (S/N) observations, you may not be able to get a convergence of the curve-of-growth unless the data is first bias-corrected. Alternatively, you may supply a **windows** file with the coordinates of the relevant sources and specify an appropriate single aperture for them in the **parameters** file. The latter option results in poorer quality lightcurves, but may suffice as a quick-look.

[3]http://www.saao.ac.za/∼marissa/SHOCpipeline/SHOCpipeline.tar
[4]marissa@saao.ac.za

A Appendix - FITS headers

The FITS headers created by SHOC are extremely basic, lacking target info (OBJECT, RA, DEC, EPOCH), site info (OBSERVAT) and other important info (FILTER, TELESCOP, INSTRUME). Headers (or comments) are often not populated as expected and/or values may be incorrect or inappropriate. **Don't blindly trust the raw data headers**.

FITS headers of the data cubes are corrected by **SHOCpipeline.py** as far as possible. If data cubes are too large (> 9000 256x256 frames), their fits headers cannot be corrected. Reduced data headers are always corrected.

A.1 EM mode with Internal triggering (raw)

```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                   16 / number of bits per data pixel
NAXIS   =                    3 / number of data axes
NAXIS1  =                  256 / length of data axis 1
NAXIS2  =                  256 / length of data axis 2
NAXIS3  =                10000 / length of data axis 3
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A...376..359H
BZERO   =                32768 / offset data range to that of unsigned short
BSCALE  =                    1 / default scaling factor
HEAD    = 'DU8201_BV'          / Head model
ACQMODE = 'Frame Transfer'     / Acquisition mode
ACT     =              0.20676 / Integration cycle time
KCT     =              0.20676 / Kinetic cycle time
NUMACC  =                    1 / Number of integrations
NUMKIN  =                10000 / Series length
READMODE= 'Image   '           / Readout mode
IMGRECT = '1, 1024, 1024, 1'   / Image format
HBIN    =                    4 / Horizontal binning
VBIN    =                    4 / Vertical binning
SUBRECT = '1, 1024, 1024, 1'   / Subimage format
DATATYPE= 'Counts  '           / Data type
XTYPE   = 'Pixel number'       / Calibration type
XUNIT   =                    0 / Calibration units
TRIGGER = 'Internal'           / Trigger mode
CALIB   = '0,1,0,0 '           / Calibration
DLLVER  = '4.18.30004.0'       / Software Version
EXPOSURE=                  0.2 / Total Exposure Time
TEMP    =              -68.863 / Temperature
```

```
READTIME=         3.333333E-07 / Pixel readout time
OPERATN =                    4 / Type of system
GAIN    =                   70 / Gain
HIERARCH EMREALGAIN =        1 / EM Real Gain
VCLKAMP =                    0 / Vertical Clock Amplitude
VSHIFT  =              6.5E-06 / Vertical Shift Speed
OUTPTAMP= 'Electron Multiplying' / Output Amplifier
PREAMP  =                  4.9 / Pre Amplifier Gain
SERNO   =                 5982 / Serial Number
UNSTTEMP=                -999. / Unstabilized Temperature
BLCLAMP =                    F / Baseline Clamp
PRECAN  =                    0 / Prescans
FLIPX   =                    0 / Horizontally Flipped
FLIPY   =                    0 / Vertically Flipped
HIERARCH COUNTCONVERTMODE =  0 / Count Convert Mode
HIERARCH COUNTCONVERT =      0 / Count Convert
HIERARCH DETECTIONWAVELENGTH = 500. / Detection Wavelength
HIERARCH SENSITIVITY =      0. / Sensitivity
HIERARCH SPURIOUSNOISEFILTER = 0 / Spurious Noise Filter Mode
HIERARCH THRESHOLD =        0. / Threshold
HIERARCH PHOTONCOUNTINGENABLED = 0 / Photon Counting Enabled
HIERARCH NOTHRESHOLDS =      0 / Number of Photon Counting Thresholds
HIERARCH PHOTONCOUNTINGTHRESHOLD1 = 0. / Photon Counting Threshold 1
HIERARCH PHOTONCOUNTINGTHRESHOLD2 = 0. / Photon Counting Threshold 2
HIERARCH PHOTONCOUNTINGTHRESHOLD3 = 0. / Photon Counting Threshold 3
HIERARCH PHOTONCOUNTINGTHRESHOLD4 = 0. / Photon Counting Threshold 4
HIERARCH AVERAGINGFILTERMODE = 0 / Averaging Filter Mode
HIERARCH AVERAGINGFACTOR =   1 / Averaging factor
HIERARCH FRAMECOUNT =        1 / Frame Count
USERTXT1= '        '           / User text
USERTXT2= '        '           / User text
USERTXT3= '        '           / User text
USERTXT4= '        '           / User text
DATE    = '2012-12-17T19:27:50' / file creation date (YYYY-MM-DDThh:mm:ss)
FRAME   = '2012-12-17T19:27:50.000' / Start of Frame Exposure
END
```

A.1.1 The following corrections are required to it

The correct comment for FRAME is "End of 1st Exposure in Data Cube" and while it appears to be capable of displaying fractions of seconds, it doesn't. Subsequently the absolute timing is not sub-second accurate unless the observer used the Externally triggered (GPS) mode.

The header RON (Read-out Noise) is obtained from the specification sheet and so is the SENSITIVITY header (if it is 0). If GAIN is 0 it is replaced by 1, but if it is > 1 the RON and GAIN values are to be divided by it.

A.2 Conventional mode with GPS triggering (raw)

For the GPS triggered timing mode the FRAME and EXPOSURE headers are completely unreliable and should be replaced by the values the user is prompted to input. The user must make a note of these when beginning an observation triggered by the GPS. There is presently NO way to determine these values after the fact if the user did not keep record of it. The SHOC user manual clearly specifies that this responsibility rests solely with the user!

```
SIMPLE  =                    T / file does conform to FITS standard
BITPIX  =                   16 / number of bits per data pixel
NAXIS   =                    3 / number of data axes
NAXIS1  =                  128 / length of data axis 1
NAXIS2  =                  128 / length of data axis 2
NAXIS3  =                14000 / length of data axis 3
EXTEND  =                    T / FITS dataset may contain extensions
COMMENT   FITS (Flexible Image Transport System) format is defined in 'Astronomy
COMMENT   and Astrophysics', volume 376, page 359; bibcode: 2001A&A...376..359H
BZERO   =                32768 / offset data range to that of unsigned short
BSCALE  =                    1 / default scaling factor
HEAD    = 'DU8201_BV'          / Head model
ACQMODE = 'Frame Transfer'     / Acquisition mode
NUMACC  =                    1 / Number of integrations
NUMKIN  =                14000 / Series length
READMODE= 'Image   '           / Readout mode
IMGRECT = '1, 1024, 1024, 1'   / Image format
HBIN    =                    8 / Horizontal binning
VBIN    =                    8 / Vertical binning
SUBRECT = '1, 1024, 1024, 1'   / Subimage format
DATATYPE= 'Counts  '           / Data type
XTYPE   = 'Pixel number'       / Calibration type
XUNIT   =                    0 / Calibration units
TRIGGER = 'External'           / Trigger mode
CALIB   = '0,1,0,0 '           / Calibration
DLLVER  = '4.18.30004.0'       / Software Version
EXPOSURE=              1.0E-05 / Total Exposure Time
TEMP    =              -58.479 / Temperature
READTIME=              1.0E-06 / Pixel readout time
OPERATN =                    4 / Type of system
```

```
GAIN    =                    0 / Gain
HIERARCH EMREALGAIN =        1 / EM Real Gain
VCLKAMP =                    0 / Vertical Clock Amplitude
VSHIFT  =              6.5E-06 / Vertical Shift Speed
OUTPTAMP= 'Conventional'       / Output Amplifier
PREAMP  =                  2.4 / Pre Amplifier Gain
SERNO   =                 5982 / Serial Number
UNSTTEMP=                -999. / Unstabilized Temperature
BLCLAMP =                    F / Baseline Clamp
PRECAN  =                    0 / Prescans
FLIPX   =                    1 / Horizontally Flipped
FLIPY   =                    0 / Vertically Flipped
HIERARCH COUNTCONVERTMODE =  0 / Count Convert Mode
HIERARCH COUNTCONVERT =      0 / Count Convert
HIERARCH DETECTIONWAVELENGTH = 500. / Detection Wavelength
HIERARCH SENSITIVITY =      0. / Sensitivity
HIERARCH SPURIOUSNOISEFILTER = 0 / Spurious Noise Filter Mode
HIERARCH THRESHOLD =        0. / Threshold
HIERARCH PHOTONCOUNTINGENABLED = 0 / Photon Counting Enabled
HIERARCH NOTHRESHOLDS =      0 / Number of Photon Counting Thresholds
HIERARCH PHOTONCOUNTINGTHRESHOLD1 = 0. / Photon Counting Threshold 1
HIERARCH PHOTONCOUNTINGTHRESHOLD2 = 0. / Photon Counting Threshold 2
HIERARCH PHOTONCOUNTINGTHRESHOLD3 = 0. / Photon Counting Threshold 3
HIERARCH PHOTONCOUNTINGTHRESHOLD4 = 0. / Photon Counting Threshold 4
HIERARCH AVERAGINGFILTERMODE = 0 / Averaging Filter Mode
HIERARCH AVERAGINGFACTOR =   1 / Averaging factor
HIERARCH FRAMECOUNT =        1 / Frame Count
USERTXT1= '        '           / User text
USERTXT2= '        '           / User text
USERTXT3= '        '           / User text
USERTXT4= '        '           / User text
DATE    = '2012-06-12T22:32:01' / file creation date (YYYY-MM-DDThh:mm:ss)
FRAME   = '2012-06-12T22:32:01.000' / Start of Frame Exposure
END
```

A.2.1 The following corrections are required to it

FRAME and EXPOSURE headers are populated by the values the user has to supply when **SHOCpipeline.py** is run, whether it was run explicitly by the user or automatically when running **QuickLook.py**. The TRIGGER is updated to GPS once those header values have been corrected. The GAIN value is replaced by the value of the SENSITIVITY header.