

# ACS calibration pipeline testing: basic image reduction

---

Max Mutchler, Robert Jedrzejewski, Warren Hack, Colin Cox  
Space Telescope Science Institute  
28 May 1999

---

## ABSTRACT

*This report describes the basic testing that was done to ensure that CALACS, the calibration software “pipeline” for the Advanced Camera for Surveys (ACS), properly reduces raw ACS images and makes the corresponding modifications to their headers. The suite of test images and reference files described here, along with the diagnostic output they produce, can serve as a simple regression test as future versions of CALACS are created.*

---

## 1. Introduction

The ACS calibration software pipeline (CALACS) will be automatically applied to all raw ACS images, to remove various instrument artifacts and populate the image headers with relevant information. For a detailed description of CALACS, see ACS Instrument Science Report 99-03 by Hack. This report describes testing of the basic image reduction tasks contained in the ACSCCD and ACS2D packages of CALACS. Single images and simple association tables (for processing multiple images) were tested for all three detectors: the Wide Field Channel (WFC), High Resolution Channel (HRC), and Solar Blind Channel (SBC). The testing of complex image reductions involving multiple images and cosmic ray rejection will be documented in a future report.

Our primary testing goals were to verify that CALACS performs the intended operations on the input image, that the proper output files are generated, and that the appropriate header keywords are created and/or populated correctly. A secondary goal was to define and create a self-consistent set of calibration reference files which are compatible with CALACS and the STScI calibration database system (CDBS).

The ACSCCD package takes a raw FITS image (\*\_raw.fits) as input. It contains the DQICORR, ATODCORR, BLEVCORR, and BIASCORR tasks, and produces an intermediate output file (\*\_blv\_tmp.fits). The ACS2D package includes the DARKCORR, FLATCORR, SHADCORR, and PHOTCORR tasks, and produces the final output image (\*\_ft.fits).

The WFC and HRC are CCD detectors which undergo a similar reduction process. The SBC is a MAMA detector which undergoes a slightly different reduction process, including the GLINCORR and LFLGCORR tasks.

In Section 6 we describe the tasks mentioned above, and the tests we have run against them, in the order that they will typically be performed in the calibration pipeline.

CALACS also produces a trailer file (\*.trl) which logs some processing details. However, a trailer file is usually already present prior to CALACS processing, since Generic Conversion generates one to log its processing. So CALACS will typically be appending information to a pre-existing trailer file, rather than creating a new file. Subsequent CALACS runs will retain the Generic Conversion notes and only overwrite the CALACS notes, which we verified.

## 2. Calibration switches

The header of a raw ACS image contains the switches (\*CORR) which tell CALACS which steps to perform, and the names and locations of the reference files (\*TAB and \*FILE) to be used. The following is excerpted from the header of our raw HRC test image, which we will use as an example throughout this report:

```
/ CALIBRATION SWITCHES:

STATFLAG =   T           / calculate statistics
DQICORR =   'PERFORM '   / data quality initialization
ATODCORR =   'PERFORM '   / correct for A to D conversion errors
BLEVCORR =   'PERFORM '   / subtract bias level computed from overscan
BIASCORR =   'PERFORM '   / subtract bias image
DARKCORR =   'PERFORM '   / subtract dark image
FLATCORR =   'PERFORM '   / flat field data
SHADCORR =   'PERFORM '   / apply shutter shading correction
PHOTCORR =   'OMIT'      / populate photometric header keywords
RPTCORR =   'OMIT'      / add individual repeat observations
CRCORR =   'OMIT'      / combine observations to reject cosmic rays
EXPSCORR =   'OMIT'      / process individual images after CR rejection
DITHCORR =   'OMIT'      / dither associated images

/ CALIBRATION REFERENCE FILES:

BPIXTAB =   'jtab$hr_bpx.fits' / bad pixel table
CCDTAB =   'jtab$hr_ccd.fits' / CCD calibration parameters
ATODTAB =   'jtab$hr_a2d.fits' / analog-to-digital correction table
OSCNTAB =   'jtab$hr_osc.fits' / CCD overscan table
BIASFILE =   'jref$hr_bia.fits' / bias image
DARKFILE =   'jref$hr_drk.fits' / dark image
PFLTFILE =   'jref$hr_pfl.fits' / pixel-to-pixel flat field image
DFLTFILE =   'jref$hr_dfl.fits' / delta flat field image
LFLTFILE =   'N/A '        / low order flat field image
SHADFILE =   'jref$hr_shd.fits' / shutter shading image
PHOTTAB =   'jtab$hr_pht.fits' / photometric throughput table
CRREJTAB =   'jtab$hr_crr.fits' / cosmic ray rejection table
DITHTAB =   'N/A '        / dither processing reference table
IDCTAB =   'N/A '        / image distortion correction table
```

## 3. Image formats

Every ACS image is a FITS file with three extensions in each image set (imset): the science data (SCI) array, the error (ERR) array, and the data quality (DQ) array. The HRC and SBC have only one imset, while the WFC has two imsets: one for each CCD chip.

For example, the following `catfits` output for our raw HRC test image (`hr_raw.fits`) shows that the dimensions of the SCI extension are 1062x1044 pixels, which is made up of the 1024x1024 science data plus 19 columns of physical overscan on both sides in the x dimension ( $19+1024+19 = 1062$ ), and 20 rows of virtual overscan on top in the y dimension ( $1024+20 = 1044$ ). No dimensions are given for the ERR and DQ extensions since they have not been initialized yet. Every extension has BITPIX set to 16, reflecting the fact that only integer pixel values (a.k.a. data numbers or DN) are possible prior to processing.

EXT#	FITSNAME	FILENAME	EXTVE	DIMENSIONS	BITPIX
0	<code>hr_raw.fits</code>	<code>hr_raw.fits</code>	1		16
1	IMAGE	SCI	1	1062x1044	16
2	IMAGE	ERR	1		16
3	IMAGE	DQ	1		16

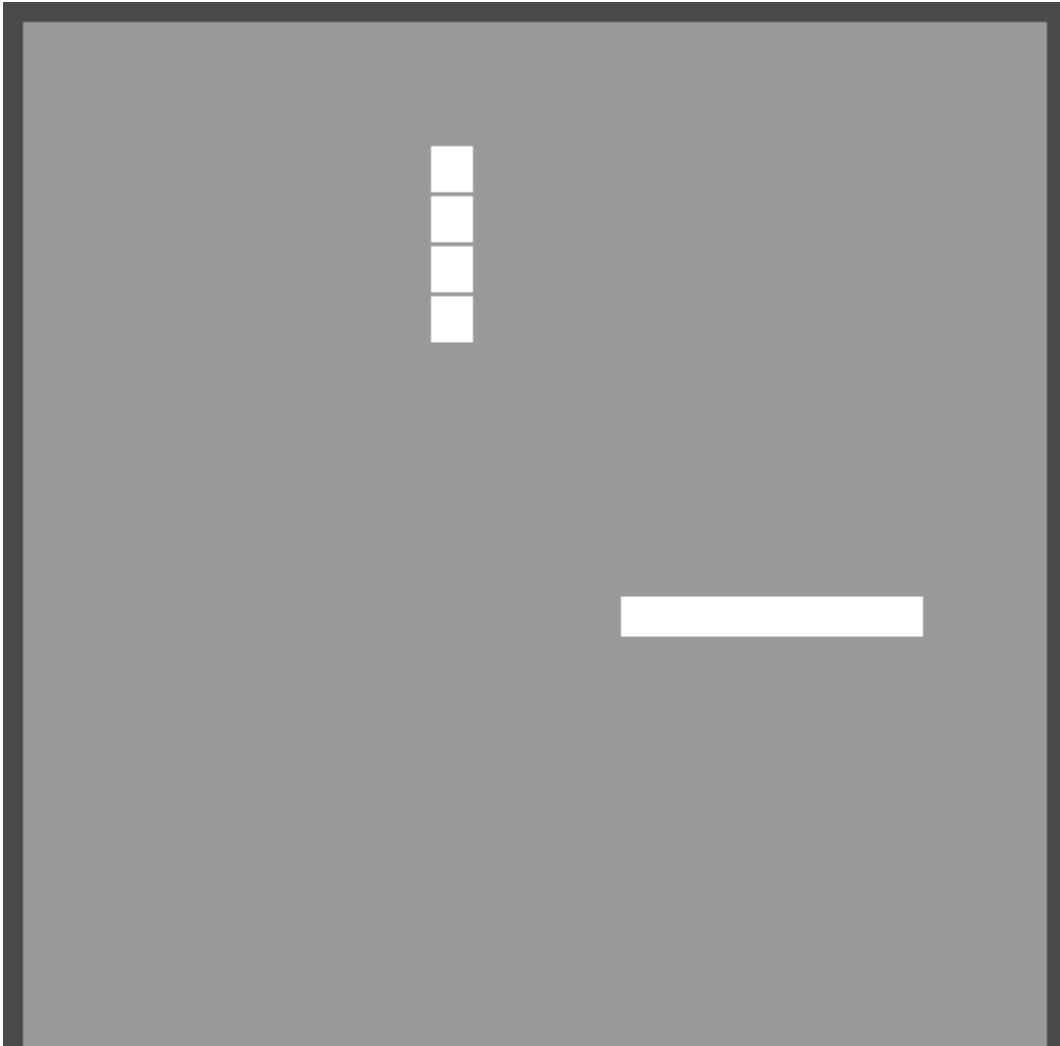
The following `catfits` output for the pipeline processed HRC test image (`hrflt.fits`) shows that the dimensions of the SCI extension are now 1024x1024, because the overscan regions have been trimmed off. The ERR and DQ arrays now have full dimensions as well, because they have been initialized and populated. The BITPIX has changed to -32 (floating point values) for the SCI and ERR extensions, because processing inevitably leads to non-integer pixel values. However, DQ flags will always remain as integer values.

EXT#	FITSNAME	FILENAME	EXTVE	DIMENSIONS	BITPIX
0	<code>hrflt.fits</code>	<code>hrflt.fits</code>	1		16
1	IMAGE	SCI	1	1024x1024	-32
2	IMAGE	ERR	1	1024x1024	-32
3	IMAGE	DQ	1	1024x1024	16

The dimensions of a raw WFC image (one imset) are 4144x2068 pixels. This includes 24 columns of physical overscan ( $24+4096+24 = 4144$ ), and 20 rows of virtual overscan ( $2048+20 = 2068$ ). Therefore, the dimensions of a pipeline processed WFC image are 4096x2048 pixels for each imset. The dimensions of both a raw and processed SBC image are 1024x1024 pixels, since there are no overscans for MAMA imaging.

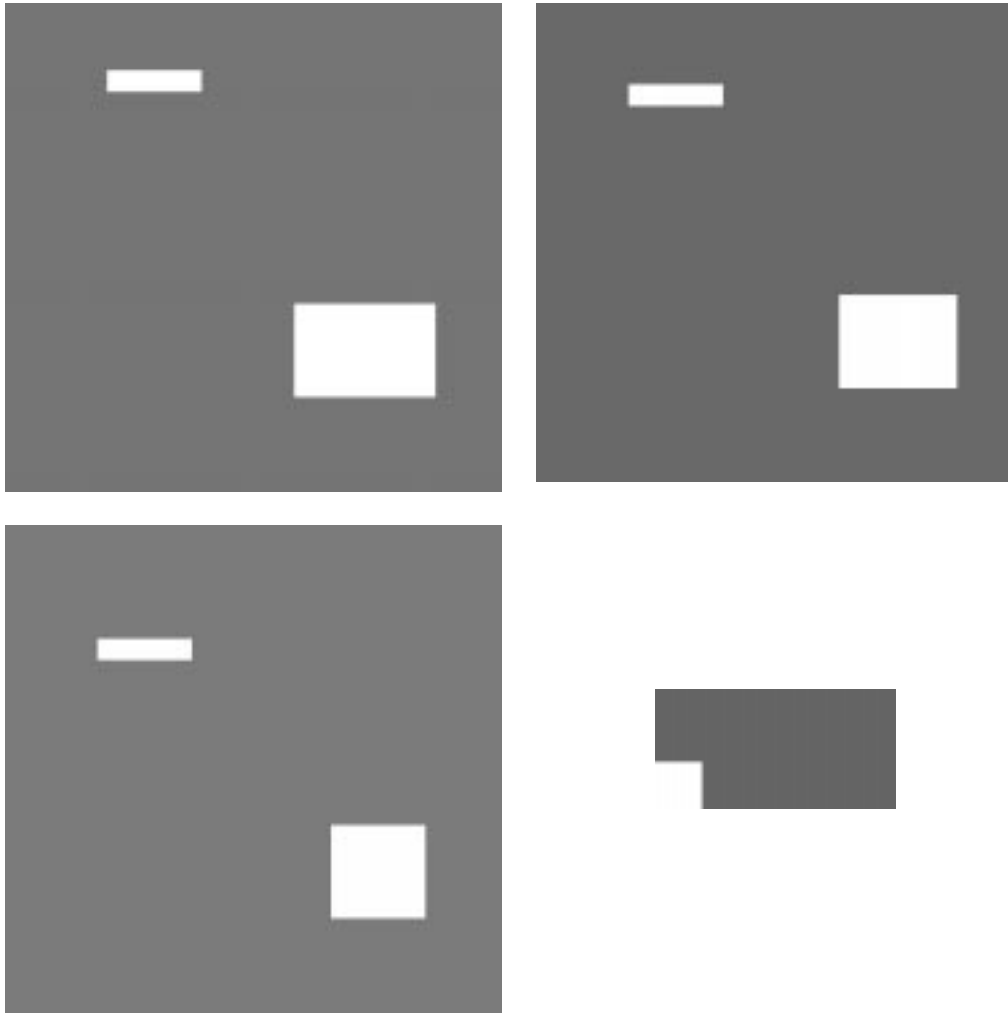
#### 4. Test images and calibration reference files

To test each of the CALACS tasks in isolation and in combination, an ideal set of raw images and calibration reference files were created by modifying the headers and pixel values of ACS ground test data. When these test files are used as input for CALACS, the resulting output image serves as a comprehensive diagnostic of the overall process. We illustrate this with our HRC test files in this section. Some “crash test” files were also created with deliberately problematic features to see how CALACS responds to erroneous input images and/or reference files.



**Figure 1.** Our raw HRC test image (hr\_raw.fits).

As we described in the previous section, the dimensions of a raw HRC image are 1062x1044 pixels. In our raw HRC test image (see Figure 1), the physical overscan regions (the first and last 19 pixels in each row) and the virtual overscan region (the last 20 rows on top) are set to 5,000 DN. The general background level is set to 15,000 DN, and some bright “objects” are set to 65,000 DN. These objects are placed off-center and adjacent to the test regions defined in the reference images in Figure 2, so that any misalignment or rotation between the raw image and the reference images would be readily apparent in the output image. In the primary image header (hr\_raw.fits[0]), we set the exposure time (EXPTIME) to 100 seconds, and the gain (CCDGAIN) to 1.



**Figure 2.** The HRC test reference images. The BIASFILE is at upper left, the DARKFILE is at upper right, the PFLTFILE is at lower left, and the SHADFILE is at lower right.

Each of the test reference images (see Figure 2) has a background which should have no effect on the output, a test region which will reveal its effect in isolation, and another test region which will reveal its effect in combination with the others.

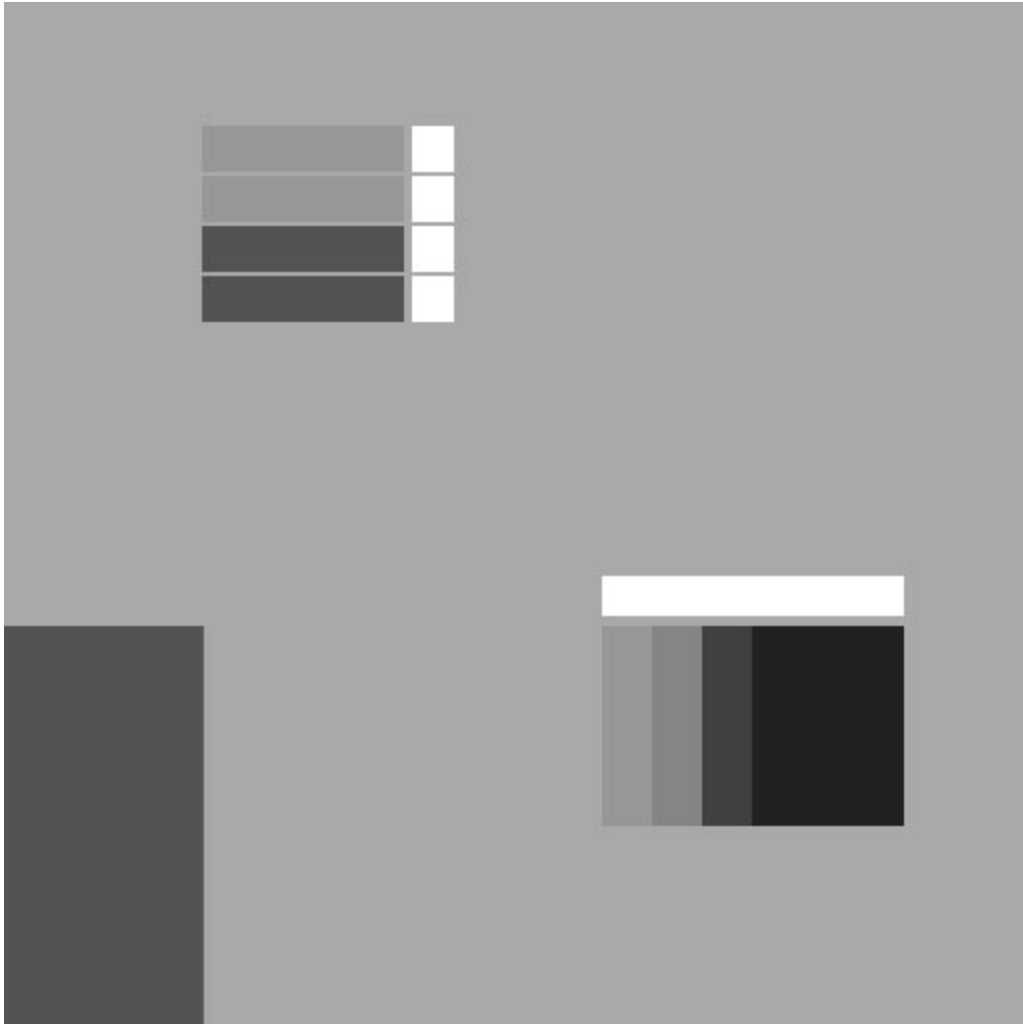
Our HRC test reference images are displayed in Figure 2. Our BIASFILE (hr\_bias.fits) has a background set to 0 DN, and test regions set to 1,000 DN. Our DARKFILE (hr\_drk.fits) has a background of 0 DN, and test regions set to 10 DN. Our PFLTFILE (hr\_pfl.fits) has a background set to 1 DN, and test regions set to 2 DN, as does the DFLTFILE (hr\_dfl.fits, not displayed in Figure 2). Our SHADFILE (hr\_shd.fits) has a background set to 0 DN, and a 100x100 pixel test region in the lower left corner which is set to 100 DN.

The BIASFILE has dimensions of 1062x1044 pixels since it is applied before the overscan regions are trimmed off. The DARKFILE and PFLTFILE have dimensions of 1024x1024 pixels since they are applied after the overscans have been trimmed. The SHADFILE is a binned image (512x256 pixels for the HRC example in Figure 2) which is scaled up by CALACS before being applied. Selected columns of the test reference tables are included in Section 8.

## **5. The test output image**

A simple visual inspection of our CALACS output image provides a quick and comprehensive diagnostic of overall pipeline processing. The output image in Figure 3 is produced when our test input image and reference files (described in the previous sections) are used to run CALACS. Any deviation from this output should be easily apparent, and indicate which aspect of CALACS is not functioning properly.

In the output image, the overscan regions are trimmed, so the image dimensions are 1024x1024 pixels. The bias level has been measured from the overscan regions (which were set to 5,000 DN) and subtracted, so the background level is now at 10,000 DN, and the bright objects are at 60,000 DN.



**Figure 3.** The CALACS output image (hr\_fit.fits) resulting from the HRC raw image in Figure 1, the reference files in Figure 2, and the reference tables in Section 8.

At upper left in our output image, the effect of BIASCORR, DARKCORR, and FLATCORR on the 10,000 DN background can be seen in isolation. At lower right, the effect of these tasks can be seen in combination from left to right, with the rightmost part of the test region being the only area that has been processed by every task. Also, the test regions are properly aligned with each other, and with the bright objects from the raw image, indicating that there are no misalignments or rotations among the various images. The SHADCORR test region, which was 100x100 pixels in the SHADFILE, creates a 200x400 pixel region in the output image, showing that CALACS has properly expanded and applied the SHADFILE.

To further verify that CALACS has performed properly, statistics were generated for various regions of the raw and processed images. For example, the pixel values in Table 1

were generated by `imstatistics` with a standard deviation of zero, to verify that each region in our raw and processed HRC images has the expected pixel values.

**Table 1.** Image statistics for various regions of the raw and processed HRC test image.

HRC image region	Image region	Raw image hr_raw.fits	CALACS output image hr_fit.fits
Overscan	[1:19,1:1044]	5,000 DN	trimmed off
Background	[600:900,600:900]	15,000 DN	10,000 DN
Objects	[650:850,415:445]	65,000 DN	60,000 DN
SHAD isolation	[1:199,1:399]	10,000 DN	5,000 DN
BIAS isolation	[210:390,860:890]	10,000 DN	9,000 DN
DARK isolation	[210:390,810:840]	10,000 DN	9,000 DN
PFLT isolation	[210:390,760:790]	10,000 DN	5,000 DN
DFLT isolation	[210:390,710:740]	10,000 DN	5,000 DN
BIAS combination	[610:640,210:390]	10,000 DN	9,000 DN
BIAS+DARK combination	[660:690,210:390]	10,000 DN	8,000 DN
BIAS+DARK+PFLT combination	[710:740,210:390]	10,000 DN	4,000 DN
BIAS+DARK+PFLT+DFLT combination	[760:790,210:390]	10,000 DN	2,000 DN

## 6. Calibration steps

The following subsections briefly describe the function of each calibration step, and the corresponding tests run against them. To analyze each step individually, we performed incremental CALACS runs, where we turned on (i.e. set to `PERFORM`) one switch at a time in the usual processing order. We also performed isolation CALACS runs where only one switch is set to `PERFORM` and all the others are set to `OMIT` (or their “identity” reference file is used, as defined below, if the step is required by subsequent steps). We verified that whenever a calibration switch was set to `PERFORM`, the corresponding tasks were performed, and the switch was changed to `COMPLETE` in the output image. When the switches were set to `OMIT`, the corresponding steps were not performed.

### *doNoise*

This first CALACS step is always done automatically, so it is not controlled via a header keyword (switch). The `ERR` array in a raw ACS image is a dimensionless null placeholder, which is expanded by `doNoise` to a two-dimensional array according to the `NPIX1` and `NPIX2` keywords. These keywords should be set such that the `ERR` array matches the dimensions of the `SCI` array.

For example, our raw HRC test image has NPIX1=1062 and NPIX2=1044. The `catfits` output in Section 3 reflects that `doNoise` properly expanded the error array, which was then trimmed of its overscan regions (see `BIASCORR`) down to 1024x1024 pixels. If the NPIX values are not set properly, `CALACS` cannot function properly.

`doNoise` gives each pixel in the newly expanded error array the value specified by `PIXVALUE` -- usually zero so that every pixel is assumed to be good by default, until flagged otherwise by subsequent `CALACS` processing.

At present, `doNoise` populates the `ERR` array with a simple noise model. For the MAMA detector (SBC), the `ERR` array is simply the square root of the `SCI` array. For the CCDs (WFC and HRC), the `ERR` array is determined by the following:

$$ERR = \sqrt{\left(\frac{SCI - bias}{gain}\right)^2 + \left(\frac{readnoise}{gain}\right)^2}$$

A more thorough test of error propagation throughout the calibration pipeline will be covered in a future report.

### ***DQICORR***

`DQICORR` populates the data quality (DQ) array with information from the bad pixel table (`BPIXTAB`). Typically, the DQ array will have already been initialized and populated with some flags from Generic Conversion. However, if the DQ array has not been initialized yet (as is the case with out test images), it will appear as a dimensionless null placeholder, and `DQICORR` will initialize it, which we verified (see Section 3).

The DQ array is a mask which identifies or “flags” pixels in the science data (`SCI`) array which are bad and why. Good pixels have a value of zero in the DQ array. Each type of bad pixel (e.g. hot, saturated, blemished, etc.) has a different flag value (see ACS Instrument Science Report 99-03 by Hack).

We created an “identity” `BPIXTAB` which contains one row with a flag value of zero. We verified that when the identity `BPIXTAB` is used, no changes are made to the DQ array. In fact, if all the other steps are set to `OMIT`, the DQ array will not be expanded at all. It will remain dimensionless in the output image unless one of the steps populates it with a bad pixel flag.

We created a test `BPIXTAB` (see Section 7) which includes all the legal bad pixel flag values and creates two orthogonal test patterns in the DQ array (see Figure 4). This allows for quick visual verification of `DQICORR` function.



**Figure 4.** A bad pixel test pattern in the DQ array of the output HRC test image (`hr_fit.fits[dq,1]`). Each row in this test pattern corresponds to bad pixel flag values (0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, and 8192).

### ***ATODCORR***

ATODCORR applies only to CCD data (WFC and HRC). Raw pixel values in the science data (SCI) array are converted from electron counts to data numbers (DN) by Generic Conversion. Small corrections to this analog-to-digital conversion are made by ATODCORR by replacing each pixel value with the corresponding value from the ATOD column of the ATODTAB, which is a simple look-up table.

We created an identity ATODTAB with an integer list from 1 to 65536 (the maximum pixel value possible), which should replace each pixel value with the exact same value. We verified that the input and output image statistics were the same when the identity ATODTAB was used.

We created a test ATODTAB that made a very simple and uniform correction to all pixel values: it increased their value by 0.5 DN. To isolate this step, we ran CALACS with either all the other switches set to OMIT, or by using identity reference files for them. Basic image statistics were then used to verify that the proper correction was performed.

### ***BLEVCORR***

BLEVCORR applies only to CCD data (WFC and HRC). Pixel values in the data array are intentionally biased towards greater pixel values, usually by several thousand DN, to minimize the possibility of negative pixel values. This bias is subtracted out by CALACS, but since the bias can vary slightly with time, it must be calculated for each image. BLEVCORR calculates the bias level by measuring the counts in the overscan region of the raw image (see Figure 1). If an input image has no overscan region, the CCDBIAS value from CCDTAB should be subtracted.

Overscans are extra readouts that occur after the science data is read out. Ideally, overscan pixel values should closely reflect the bias level. But in reality, some residual charge

from the science pixels can appear in them as a result of imperfect charge transfer efficiency (CTE). The OSCNTAB file defines the science data region, and the region that BLEVCORR uses to estimate the bias level, which is usually an area slightly smaller than the full overscan region. This is because most deferred charge (CTE residuals, etc) appears in the first pixel or two of overscan. BLEVCORR fits a straight line to the mean of the pixel intensities in the overscan region as a function of row number, and this straight line model is subtracted from the data. So excluding the first few “tainted” pixels from the BLEVCORR calculation usually improves the accuracy.

The MEANBLEV keyword value was properly changed from zero in the raw image, to the overscan level set in our raw test images (5,000 DN, see Figure 1), so CALACS properly calculated the bias level from the overscan region. Image statistics taken from the raw and output image verified that this bias value was properly subtracted from the output SCI array (see Table 1).

We also verified that when the BLEVCORR switch was set to OMIT, that this step was not performed. Specifically, the overscan regions were not trimmed, the MEANBLEV keyword value was not modified, and subsequent ACS2D processing was not allowed.

The CRPIX1 and CRPIX2 keywords reflect the offset created by trimming the overscans. The LTV1 and LTV2 keywords indicate the coordinates of the first science pixel in the SCI array, which should be 0,0 after the overscan has been trimmed. For our raw HRC test image, both CRPIX1 and CRPIX2 are zero, and LTV1=19 and LTV2=0. In our output HRC test image, CRPIX1=-19 and CRPIX2=0, and LTV1 and LTV2 are both zero.

## ***BIASCORR***

BIASCORR applies only to CCD data (WFC and HRC). The bias reference image (BIASFILE) contains any pixel-to-pixel variations in the bias level. These artifacts are created by variations in the amplifiers which occur during a readout. BIASCORR removes these artifacts by subtracting the BIASFILE from the raw image.

We created an identity BIASFILE which contained all zero pixel values, and verified with image statistics that its use has no effect on the output image. We also created a test BIASFILE which contained mostly zeros, but also contained two regions with non-zero pixel values (see Figure 2) for testing this step in isolation and in combination with the other steps. Image statistics verified that the test regions contained the correct pixel values in the output image (see Table 1). CALACS halted and generated a helpful error message when a BIASFILE of the wrong dimensions was used.

BIASCORR is the last step in the ACSCCD package. Following this step the intermediate output file `*_blv_tmp.fits` is created with the bias level subtracted, and the overscan regions trimmed off. We verified that the dimensions of the science data array (SCI) were reduced. For the WFC the dimensions change from 4144x2062 to 4096x2048

pixels, and for the HRC the dimensions change from 1062x1044 to 1024x1024 pixels (see the `catfits` output in Section 3). The remaining tasks are part of the ACS2D package, and their effect will be seen in the final output image `*_flt.fits`.

### ***GLINCORR and LFLGCORR***

We discuss these tasks together because they both involve linearity corrections which apply only to MAMA data from the SBC.

GLINCORR corrects for global non-linearities. If the GLOBRATE value in the science header is less than the GLOBAL\_LIMIT value in the MLINTAB reference file, then a linearity correction is performed. The correction is defined by the value of the exponential factor TAU in the MLINTAB.

We created an identity MLINTAB which has no effect on the output image. There are two ways to accomplish this: set TAU=0 or set the GLOBAL\_LIMIT lower than the GLOBRATE. We also created a test MLINTAB which makes a simple correction (it divides the pixel values by two) and verified the resulting output. We set GLOBRATE=100, and the following is our test MLINTAB:

```
# Table sb_lin_test.fits[1] Thu 14:49:21 27-May-99
# DETECTOR GLOBAL_LIMIT LOCAL_LIMIT TAU EXPAND
SBC 3.E5 200. -0.014 1.
```

LFLGCORR flags local and global non-linearities in the data quality (DQ) array. It will flag (with a value of 256) any pixels which exceed the LOCAL\_LIMIT count rate. Since we set the exposure time to be 100 seconds, the background has a count rate of 100 counts per second, and the bright objects have a count rate of 600 counts per second. We set the LOCAL\_LIMIT to 200 counts per second so that only the bright object pixels, and any adjacent pixels (since we set EXPAND=1), would be flagged. We verified that the DQ array was populated as expected.

### ***DARKCORR***

DARKCORR applies only to CCD data (WFC and HRC). The dark reference image (DARKFILE) is multiplied by the exposure time and divided by the gain, and then subtracted from the trimmed image produced by ACSCCD.

We created an identity DARKFILE which contained all zero pixel values, and verified with image statistics that its use has no effect on the output image. We also created a test DARKFILE which contained mostly zeros, but also contained two regions with non-zero pixel values (see Figure 2) for testing this step in isolation and in combination with the other steps.

Image statistics verified that the test regions contained the expected pixel values in the output image (see Table 1). For our HRC test image, DARKCORR subtracted 1,000 DN ( $10 * 100 / 1$ ) in the test regions. The MEANDARK keyword value was changed from zero to 56.93.

### ***FLATCORR***

The PFLTFILE contains the pixel-to-pixel flat field artifacts. FLATCORR divides the input image by the PFLTFILE image.

We created an identity PFLTFILE which contained all pixel values of one, and verified with image statistics that its use has no effect on the output image. We also created a test PFLTFILE which contained mostly ones, but also contained test regions with pixel values of two (see Figure 2). This was also done for a delta flat (DFLTFILE), which can be used to add new flat field features which may appear over time. Image statistics verified that the output test regions contained pixel values which were half the input value (see Table 1), as expected. We did not test low order flat (LFLTFILE) usage at this time.

### ***SHADCORR***

SHADCORR corrects for the difference in illumination across the CCD as the shutter opens and closes. This is generally a slight gradient across the CCD, which would only be significant for very short exposures. The shutter shading correction for a given pixel is based on the corresponding SHADFILE pixel value as follows:

$$\text{corrected} = \text{raw} * \text{EXPTIME} / (\text{EXPTIME} + \text{SHADFILE})$$

The SHADFILE is a binned image which gets expanded by CALACS to the science image dimensions before being applied. For example, our SHADFILE in Figure 2 gets expanded by a factor of two in the x dimension and a factor of 4 in the y dimension (to 1024x1024 pixels) for an HRC image.

We created an identity SHADFILE which contained all zero pixel values, and verified with image statistics that its use has no effect on the output image. We also created a test SHADFILE which contained mostly zeros, but with a test region in the lower left corner set to non-zero values (see Figure 2). Our HRC test image has an exposure time of 100 seconds, and our corresponding SHADFILE (see Figure 2) has a 100x100 pixel test region set to 100 DN . The output image in Figure 3 shows that this region was properly expanded to 200x400 pixels. Also, Table 1 shows that the pixel values in the test region are 5,000 DN or half the background level, which verifies that the correction equation above has been properly applied:  $5,000 \text{ DN} = 10,000 \text{ DN} * 100\text{s} / (100\text{s} + 100 \text{ DN})$ .

### ***PHOTCORR***

We did not test this task at this time.

## ***STATFLAG***

The statistics flag (STATFLAG) is a boolean switch which controls whether CALACS calculates the minimum, maximum, and mean good pixel values in the image, and populates the GOODMIN, GOODMAX, and GOODMEAN keywords.

We set these keyword values to zero in our raw test images. When STATFLAG was set to F (false), these values remained unchanged. When STATFLAG was set to T (true), these keywords were populated with the proper values. For our HRC output image GOODMIN=2000, GOODMAX=60000, and GOODMEAN=10041.112

We set STATFLAG to illegal values of PERFORM, OMIT, and COMPLETE and these keywords were left unmodified. CALACS properly exited with an error message indicating that STATFLAG had an invalid value.

## **7. Summary**

The basic testing described in this report demonstrates that CALACS is functioning properly, and it can serve as a regression test for future versions of CALACS. The following tests were beyond the scope of this project, but will be covered in future reports: error propagation, cosmic ray rejection, header keyword population (especially photometric keywords), complex association tables, further “crash tests”, and processing of grism, ramp filter, and subarray data.

## **8. Appendix: Test reference tables**

Selected columns of the test HRC reference tables are printed below:

```
# Table hr_osc_id.fits[1] Mon 14:00:23 01-Mar-99
# CCDAMP CCDCHIP NX NY TRIMX1 TRIMX2 TRIMY1 TRIMY2
D 1 1062 1044 19 19 0 20
# BIASSECTA1 BIASSECTA2 BIASSECTB1 BIASSECTB2
2 18 1046 1062
# VX1 VX2 VY1 VY2
21 1044 1025 1044
```

```

# Table hr_bpx_test.fits[1] Wed 13:46:42 24-Mar-99
# CCDAMP PIX1 PIX2 LENGTH AXIS VALUE CCDCHIP CCDGAIN

D      200   600   200    2    0    1    1
D      202   600   200    2    1    1    1
D      204   600   200    2    2    1    1
D      206   600   200    2    4    1    1
D      208   600   200    2    8    1    1
D      210   600   200    2   16    1    1
D      212   600   200    2   32    1    1
D      214   600   200    2   64    1    1
D      216   600   200    2  128    1    1
D      218   600   200    2  256    1    1
D      220   600   200    2  512    1    1
D      222   600   200    2 1024    1    1
D      224   600   200    2 8192    1    1

```

```

# Table hr_a2d_test.fits[1] Tue 14:54:00 23-Mar-99
# CCDAMP CCDCHIP CCDGAIN NELEM ATOD

D      1      1 65536    0.5
D      1      2 65536    0.5
D      1      4 65536    0.5
D      1      8 65536    0.5

```

```

# Table hr_ccd_id.fits[1] Thu 11:48:03 04-Mar-99
# CCDAMP CCDCHIP CCDGAIN CCDBIAS SATURATE

D      1      1      4000. 1.0E5
D      1      1      4000. 1.0E5
D      1      1      4000. 1.0E5

```

```

# CCDOFSTA CCDOFSTB CCDOFSTC CCDOFSTD

      1      1      1      1
      2      2      2      2
      3      3      3      3

```

```

# ATODGNA ATODGNB ATODGNC ATODGND

      1.      1.      1.      1.
      1.      1.      1.      1.
      1.      1.      1.      1.

```

```

# READNSEA READNSEB READNSEC READNSED

      3.1      3.2      3.3      3.4
      3.1      3.2      3.3      3.4
      3.1      3.2      3.3      3.4

```

## 9. Appendix: IRAF scripts for testing CALACS

The IRAF scripts used to prepare the HRC test files, run CALACS, and check the results are provided below in the order of their use. Similar scripts were used for WFC and SBC testing. The `setraw.cl` script sets the pixel values in the science data array (SCI) in the raw input image (`hr_raw.fits`). It also sets the exposure time (EXPTIME) and gain (CCDGAIN) keywords in the header.

The `makeref.cl` script creates the identity and test reference images. It also makes the test reference tables which can be edited to make the identity and test versions. However, the ATODFILE cannot be easily edited, because the ATOD array contained in it makes it a 3-dimensional table. The `a2d_update.cl` script creates the test ATOD array (which adds 0.5 to each pixel value) and inserts it into the test ATODFILE.

The `runcal.cl` script sets the calibration switches and identifies the reference files in the primary header of the raw input image. Then it runs CALACS. The `check_*.cl` scripts help the user quickly verify that the relevant keywords and pixel values in the raw, intermediate, and final output images have been properly modified by CALACS. The `display_SCI.cl` script quickly displays the science data array (SCI) for the raw, intermediate, and final output images, so visual verification can be done. Similar scripts were created to display the ERR and DQ arrays, and the test reference images.

### *setraw.cl*

```
print ("Setting exposure time and gain in raw image...")
hedit hr_raw.fits[0] EXPTIME 100
hedit hr_raw.fits[0] CCDGAIN 1
print ("Setting pixel values in raw image...")
imrep hr_raw.fits[sci,1] value=5000
imrep hr_raw.fits[sci,1][20:1043,1:1024] value=15000
imrep hr_raw.fits[sci,1][429:469,855:900] value=65000
imrep hr_raw.fits[sci,1][429:469,805:850] value=65000
imrep hr_raw.fits[sci,1][429:469,755:800] value=65000
imrep hr_raw.fits[sci,1][429:469,705:750] value=65000
imrep hr_raw.fits[sci,1][619:919,410:450] value=65000
print ("Pixel values in the raw input image...")
imstatistics.fields = "min,max,npix,image"
imstat hr_raw.fits[sci,1]
hedit hr_raw.fits[0] EXPTIME .
hedit hr_raw.fits[0] CCDGAIN .
```

*makeref.cl*

```
print ("Removing old reference images...")
!rm hr_bia*.fits
!rm hr_drk*.fits
!rm hr_pfl*.fits
!rm hr_dfl*.fits
!rm hr_lfl*.fits
!rm hr_shd*.fits
print ("Copy new versions of reference tables?")
!cp -i /milan/data6/calacs/ref/hr_bpx.fits hr_bpx_id.fits
!cp -i /milan/data6/calacs/ref/hr_ccd.fits hr_ccd_id.fits
!cp -i /milan/data6/calacs/ref/hr_osc.fits hr_osc_id.fits
!cp -i /milan/data6/calacs/ref/hr_a2d.fits hr_a2d_id.fits
!cp -i /milan/data6/calacs/ref/hr_crr.fits hr_crr_id.fits
!cp -i /milan/data6/calacs/ref/hr_pht.fits hr_pht_id.fits
print ("You must tedit the ID tables to make TEST versions.")
print ("Making IDENTITY reference images...")
imreplace.imaginary = 0.
imreplace.radius = 0.
!cp /milan/data6/calacs/ref/hr_bia.fits hr_bia_id.fits
imrep hr_bia_id.fits[sci,1] value=0
!cp /milan/data6/calacs/ref/hr_drk.fits hr_drk_id.fits
imrep hr_drk_id.fits[sci,1] value=0
!cp /milan/data6/calacs/ref/hr_pfl.fits hr_pfl_id.fits
imrep hr_pfl_id.fits[sci,1] value=1
!cp /milan/data6/calacs/ref/hr_dfl.fits hr_dfl_id.fits
imrep hr_dfl_id.fits[sci,1] value=1
!cp /milan/data6/calacs/ref/hr_lfl.fits hr_lfl_id.fits
imrep hr_lfl_id.fits[sci,1] value=1
!cp /milan/data6/calacs/ref/hr_shd.fits hr_shd_id.fits
imrep hr_shd_id.fits[sci,1] value=0
print ("Making TEST reference images...")
!cp /milan/data6/calacs/ref/hr_bia.fits hr_bia_test.fits
imrep hr_bia_test.fits[sci,1] value=0
imrep hr_bia_test.fits[sci,1][219:419,855:900] value=1000
imrep hr_bia_test.fits[sci,1][619:919,200:400] value=1000
```

```

lcp /milan/data6/calacs/ref/hr_drk.fits hr_drk_test.fits
imrep hr_drk_test.fits[sci,1] value=0
imrep hr_drk_test.fits[sci,1][200:400,805:850] value=10
imrep hr_drk_test.fits[sci,1][650:900,200:400] value=10
lcp /milan/data6/calacs/ref/hr_pfl.fits hr_pfl_test.fits
imrep hr_pfl_test.fits[sci,1] value=1
imrep hr_pfl_test.fits[sci,1][200:400,755:800] value=2
imrep hr_pfl_test.fits[sci,1][700:900,200:400] value=2
lcp /milan/data6/calacs/ref/hr_dfl.fits hr_dfl_test.fits
imrep hr_dfl_test.fits[sci,1] value=1
imrep hr_dfl_test.fits[sci,1][200:400,705:750] value=2
imrep hr_dfl_test.fits[sci,1][750:900,200:400] value=2
lcp /milan/data6/calacs/ref/hr_lfl.fits hr_lfl_test.fits
imrep hr_lfl_test.fits[sci,1] value=1
imrep hr_lfl_test.fits[sci,1][200:400,655:700] value=2
imrep hr_lfl_test.fits[sci,1][800:900,200:400] value=2
lcp /milan/data6/calacs/ref/hr_shd.fits hr_shd_test.fits
imrep hr_shd_test.fits[sci,1] value=0
imrep hr_shd_test.fits[sci,1][1:100,1:100] value=100
print ("Test reference image statistics...")
imstatistics.fields = "min,max,mean,stddev,npix,image"
imstatistics.lower = INDEF
imstatistics.upper = INDEF
imstatistics.binwidth = 0.1
imstatistics.format = yes
imstat hr_bia*.fits[sci,1]
imstat hr_drk*.fits[sci,1]
imstat hr_pfl*.fits[sci,1]
imstat hr_dfl*.fits[sci,1]
imstat hr_shd*.fits[sci,1]

```

### *a2d\_update.cl*

```
print ("Removing old files and creating new ones...")
!rm atodwf*.tab
!rm atodhr*.tab
!rm hr_a2d_test.fits
!rm wf_a2d_test.fits
cp hr_a2d_id.fits hr_a2d_test.fits
print ("Creating the ATOD tables for each gain setting...")
!cp 2_a2d_id.tab atodhr1.tab
!cp 2_a2d_id.tab atodhr2.tab
!cp 2_a2d_id.tab atodhr4.tab
!cp 2_a2d_id.tab atodhr8.tab
print ("Modifying the ATOD tables...")
tcalc atodhr1.tab c1 "c1+0.5"
tcalc atodhr2.tab c1 "c1+0.5"
tcalc atodhr4.tab c1 "c1+0.5"
tcalc atodhr8.tab c1 "c1+0.5"
print ("Updating ATOD array in reference table...")
set xstis = /data/eh3/xstis/
task xstis.pkg = xstis$xstis.cl
reset helpdb = (envget ("helpdb") // ",xstis$helpdb.mip")
xstis
colupdate atodhr1.tab hr_a2d_test.fits "ccdamp=D,ccdgain=1" ATOD
colupdate atodhr2.tab hr_a2d_test.fits "ccdamp=D,ccdgain=2" ATOD
colupdate atodhr4.tab hr_a2d_test.fits "ccdamp=D,ccdgain=4" ATOD
colupdate atodhr8.tab hr_a2d_test.fits "ccdamp=D,ccdgain=8" ATOD
tprint *a2d_id.fits columns="ccdamp,chip,ccdgain,nelem,atod"
tprint *a2d_test.fits columns="ccdamp,ccdgain,nelem,atod"
!rm atodwf*.tab
!rm atodhr*.tab
```

*runcal.cl*

```
print ("Removing old output...")
!rm hr_blv_tmp.fits
!rm hr_ft.fits
!rm hr.trl
print ("Setting header keywords...")
hedit hr_raw.fits[0] DQICORR PERFORM
hedit hr_raw.fits[0] ATODCORR PERFORM
hedit hr_raw.fits[0] BLEVCORR PERFORM
hedit hr_raw.fits[0] BIASCORR PERFORM
hedit hr_raw.fits[0] DARKCORR PERFORM
hedit hr_raw.fits[0] FLATCORR PERFORM
hedit hr_raw.fits[0] SHADCORR PERFORM
hedit hr_raw.fits[0] PHOTCORR OMIT
hedit hr_raw.fits[0] STATFLAG T
hedit hr_raw.fits[0] RPTCORR OMIT
hedit hr_raw.fits[0] CRCORR OMIT
hedit hr_raw.fits[0] EXPSCORR OMIT
hedit hr_raw.fits[0] BPIXTAB /milan/data6/calacs/reftest/hr_bpx_test.fits
hedit hr_raw.fits[0] CCDTAB /milan/data6/calacs/reftest/hr_ccd_id.fits
hedit hr_raw.fits[0] OSCNTAB /milan/data6/calacs/reftest/hr_osc_id.fits
hedit hr_raw.fits[0] ATODTAB /milan/data6/calacs/reftest/hr_a2d_test.fits
hedit hr_raw.fits[0] BIASFILE /milan/data6/calacs/reftest/hr_bia_test.fits
hedit hr_raw.fits[0] DARKFILE /milan/data6/calacs/reftest/hr_drk_test.fits
hedit hr_raw.fits[0] PFLTFILE /milan/data6/calacs/reftest/hr_pfl_test.fits
hedit hr_raw.fits[0] DFLTFILE /milan/data6/calacs/reftest/hr_dfl_test.fits
hedit hr_raw.fits[0] SHADFILE /milan/data6/calacs/reftest/hr_shd_test.fits
hedit hr_raw.fits[0] LFLTFILE N/A
hedit hr_raw.fits[0] IDCTAB N/A
hedit hr_raw.fits[0] PHOTTAB /milan/data6/calacs/reftest/hr_pht_id.fits
hedit hr_raw.fits[0] CRREJTAB /milan/data6/calacs/reftest/hr_crr_id.fits
print ("Running CALACS...")
calacs.verbose = yes
calacs.savetmp = yes
calacs.quiet = no
calacs hr_raw.fits
```

### *check\_keywords.cl*

```
print ("Keyword values from the raw image... ")
hedit hr_raw.fits[0] *CORR .
hedit hr_raw.fits[0] *TAB .
hedit hr_raw.fits[0] *FILE .
hedit hr_raw.fits[0] EXPTIME .
hedit hr_raw.fits[sci,1] CRPIX*,LTV*,GOOD*,MEANBLEV,MEANDARK .
print ("Keyword values from the intermediate image... ")
hedit hr_blv_tmp.fits[0] *CORR .
hedit hr_blv_tmp.fits[sci,1] CRPIX*,LTV*,GOOD*,MEANBLEV,MEANDARK .
print ("Keyword values from the final output image... ")
hedit hr_ft.fits[0] *CORR .
hedit hr_ft.fits[sci,1] CRPIX*,LTV*,GOOD*,MEANBLEV,MEANDARK .
```

### *check\_stats.cl*

```
imstatistics.fields = "mean,stddev,npix,image"
imstatistics.format = yes
print ("Raw image statistics...")
print ("Overscan:")
imstat hr_raw.fits[sci,1][1:19,*]
print ("Feature:")
imstat hr_raw.fits[sci,1][650:850,415:445]
print ("Background:")
imstat hr_raw.fits[sci,1][600:900,600:900]
print ("Error and data quality arrays:")
imstat hr_raw.fits[err,1]
imstat hr_raw.fits[dq,1]
print ("Processed image statistics...")
print ("Feature:")
imstat hr_ft.fits[sci,1][601:899,415:445]
print ("Background:")
imstat hr_ft.fits[sci,1][600:900,600:900]
print ("Error and data quality arrays:")
imstat hr_ft.fits[err,1]
imstat hr_ft.fits[dq,1]
print ("Isolation regions...")
print ("SHAD subtraction:")
```

```
imstat hr_ft.fits[sci,1][1:199,1:399]
print ("BIAS subtraction:")
imstat hr_ft.fits[sci,1][210:390,860:890]
print ("DARK subtraction:")
imstat hr_ft.fits[sci,1][210:390,810:840]
print ("FLAT field division:")
imstat hr_ft.fits[sci,1][210:390,760:790]
print ("DELTA FLAT field division:")
imstat hr_ft.fits[sci,1][210:390,710:740]
print ("Combination region...")
print ("BIAS subtraction:")
imstat hr_ft.fits[sci,1][610:640,210:390]
print ("BIAS+DARK subtraction:")
imstat hr_ft.fits[sci,1][660:690,210:390]
print ("BIAS+DARK+FLAT:")
imstat hr_ft.fits[sci,1][710:740,210:390]
print ("BIAS+DARK+FLAT+DELTA:")
imstat hr_ft.fits[sci,1][760:790,210:390]
```

*display\_SCI.cl*

```
display.zscale = no
display.zrange = no
display.ztrans = "linear"
display.z1=-10
display.z2=65001
display hr_raw.fits[1] 1
display hr_blv_tmp.fits[1] 2
display hr_ft.fits[1] 3
```