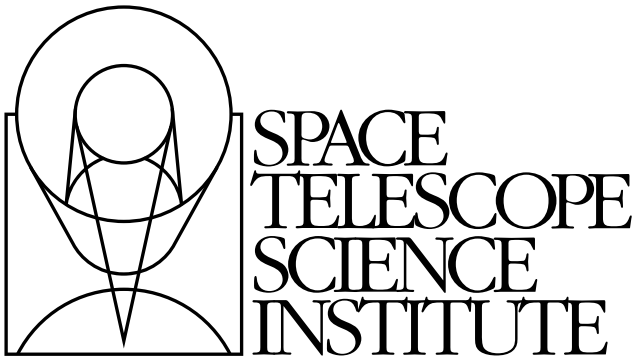

acstools Documentation

Release 1.7.1

Warren Hack, Matt Davis, Pey Lian Lim, Jay Anderson, Norman Gr

June 09, 2014

CONTENTS

Modules for Advanced Camera for Surveys (ACS).

http://www.stsci.edu/hst/acs/

Contents:

CHAPTER

ONE

CALACS

The calacs module contains a function `calacs` that calls the CALACS executable. Use this function to facilitate batch runs of CALACS, or for the TEAL interface.

1.1 Examples

In Python without TEAL:

```
>>> from acstools import calacs
>>> calacs.calacs(filename)
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from acstools import calacs
>>> teal.teal('calacs')
```

In Pyraf:

```
--> import acstools
--> epar calacs
```

acstools.calacs.**calacs**(*input_file*, *exec_path=None*, *time_stamps=False*, *temp_files=False*, *verbose=False*, *debug=False*, *quiet=False*, *single_core=False*)

Run the calacs.e executable as from the shell.

By default this will run the calacs given by 'calacs.e'.

Parameters

input_file : str

Name of input file.

exec_path : str, optional

The complete path to a calacs executable.

time_stamps : bool, optional

Set to True to turn on the printing of time stamps.

temp_files : bool, optional

Set to True to have CALACS save temporary files.

verbose : bool, optional

Set to True for verbose output.

debug : bool, optional

Set to True to turn on debugging output.

quiet : bool, optional

Set to True for quiet output.

single_core : bool, optional

CTE correction in CALACS will by default try to use all available CPUs on your computer. Set this to True to force the use of just one CPU.

CHAPTER

TWO

CALACS.E (HSTCAL)

A detailed description of this new and improved CALACS is available in ACS Data Handbook v7.0 or later. If you have questions not answered in the documentation, please contact STScI Help Desk (`help[at]stsci.edu`).

2.1 Running CALACS

2.1.1 Where to Find CALACS

CALACS is now part of HSTCAL package, which can be downloaded from STSDAS download page.

2.1.2 Usage

From the command line:

```
calacs.e jb1f89eaq_raw.fits [command line options]
```

2.1.3 Command Line Options

CALACS supports several command line options:

- -t
  - Print verbose time stamps.
- -s
  - Save temporary files.
- -v
  - Turn on verbose output.
- -d
  - Turn on debug output.
- -q
  - Turn on quiet output.
- -1
  - Turn off parallel processing for PCTECORR. Try this option if you encounter problems running CALACS with PCTECORR=PERFORM.

2.1.4 Parallel Processing with OpenMP

By default, CALACS will attempt to perform PCTECORR using all available CPUs on your machine. You can set the maximum number of CPUs available for CALACS by setting the `OMP_NUM_THREADS` environmental variable.

In tcsh:

```
setenv OMP_NUM_THREADS 2
```

In bash:

```
export OMP_NUM_THREADS=2
```

2.1.5 Batch CALACS

The recommended method for running CALACS in batch mode is to use Python and the `acstools` package in STSDAS distribution.

For example:

```
from acstools import calacs
import glob

for fits in glob.iglob('j*_raw.fits'):
    calacs.calacs(fits)
```

2.2 BIASCORR

BIASCORR is now performed before BLEVCORR. This should not significantly affect science results. This change was necessary to accomodate BIASFILE subtraction in DN with the rest of the calculations done in ELECTRONS.

2.3 Unit Conversion to Electrons

The image is multiplied by gain right after BIASCORR, converting it to ELECTRONS. This step is no longer embedded within FLATCORR.

2.4 BLEVCORR

BLEVCORR is now performed after BIASCORR. Calculations are done in ELECTRONS.

For post-SM4 full-frame WFC exposures, it also includes:

- de-striping to remove stripes introduced by new hardware installed during SM-4 (J. Anderson; ACS ISR 2011-05); and
- if JWROTYPE=DS_int and CCDGAIN=2, also correct for bias shift (ACS ISR 2012-02) and cross-talk (N. Grogin; ACS ISR 2010-02).

2.5 Pixel-Based CTE Correction (PCTECORR)

For all full-frame WFC exposures, pixel-based CTE correction (ACS ISR 2010-03 and 2012-03) is applied in ACSCTE that occurs after the ACSCCD series; i.e., after BLEVCORR.

Because the CTE correction is applied before DARKCORR and FLSHCORR, it is necessary to use a CTE-corrected dark (DRKCFILE) if the PCTECORR step is enabled.

Parameters characterizing the CTE correction are stored in a reference table, PCTETAB.

Note: CALACS 8.2 and later uses a slightly different PCTETAB format, where the COL_SCALE extension does not include overscan columns.

2.5.1 Required Keywords

Running CALACS with pixel-based CTE correction requires the following header keywords:

- PCTECORR
  - By default, set to PERFORM for all full-frame WFC exposures, except BIAS.
- PCTETAB
  - Reference table containing CTE correction parameters. By default, it should be in the `jref` directory and have the suffix `_cte.fits`.
- DRKCFILE
  - Similar to DARKFILE but with CTE correction performed. By default, it should be in the `jref` directory and have the suffix `_dkc.fits`. This is necessary because PCTECORR is done before DARKCORR.

2.5.2 Optional Keywords

You may adjust some CTE correction algorithm parameters by changing the following keywords in RAW image header. The default values are picked for optimum results in a typical WFC full-frame exposure. Changing these values is not recommended unless you know what you are doing.

- PCTENSMD
  - Read noise mitigation mode:
    - 0 - No mitigation
    - 1 - Perform noise smoothing
    - 2 - No noise smoothing
  - Overwrites NSEMODEL in PCTETAB.
- PCTERNCL
  - Read noise level of image in ELECTRONS. This is not used if you specified no mitigation in read noise mitigation mode.
  - Overwrites RN_CLIP in PCTETAB.
- PCTETRSH
  - Over-subtraction correction threshold. Pixel below this value in ELECTRONS after CTE correction is considered over-corrected and will re-corrected with smaller correction.

  - Overwrites SUBTHRSH in PCTETAB.
- PCTESMIT
  - Number of iterations of readout simulation per column.
  - Overwrites SIM_NIT in PCTETAB.
- PCTESHFT
  - Number of shifts each readout simulation is broken up into.
  - Overwrites SHFT_NIT in PCTETAB.

2.6 Dark Current Subtraction (DARKCORR)

It uses DARKFILE if PCTECORR=OMIT, otherwise it uses DRKCFILE (CTE-corrected dark reference file).

Dark image is now scaled by EXPTIME and FLASHDUR. For post-SM4 non-BIAS WFC images, extra 3 seconds are also added to account for idle time before readout. Any image with non-zero EXPTIME is considered not a BIAS.

2.7 Post-Flash Correction (FLSHCORR)

Post-flash correction is now performed after DARKCORR in the ACS2D step. When FLSHCORR=PERFORM, it uses FLSHFILE.

2.8 FLATCORR

Conversion from DN to ELECTRONS no longer depends on FLATCORR=PERFORM. Unit conversion is done for all exposures after BIASCORR.

2.9 Photometry Keywords (PHOTCORR)

The PHOTCORR step is now performed using tables of precomputed values instead of calls to SYNPHOT. The correct table for a given image must be specified in the IMPHTTAB header keyword in order for CALACS to perform the PHOTCORR step. By default, it should be in the `jref` directory and have the suffix `_imp.fits`. Each DETECTOR uses a different table.

If you do not wish to use this feature, set PHOTCORR to OMIT.

2.10 CALACS Output

Using RAW as input:

- flt.fits: Same as existing FLT.
- flc.fits: Similar to FLT, except with pixel-based CTE correction applied.

Using ASN as input with ACSREJ:

- crj.fits: Same as existing CRJ.

- crc.fits: Similar to CRJ, except with pixel-based CTE correction applied.

CALACS uses HSTIO that utilizes `PIXVALUE` keyword to represent a data extension with constant value. However, this is not a standard FITS behavior and is not recognized by PyFITS. Therefore, one should use `stsci.tools.stpyfits`, which is distributed as part of `stsci_python`, instead of `pyfits` or `astropy.io.fits` when working with CALACS products. To use `stpyfits` in Python:

```
from stsci.tools import stpyfits as pyfits
```

CHAPTER

THREE

ACSCCD

The acsccd module contains a function `acsccd` that calls the ACSCCD executable. Use this function to facilitate batch runs of ACSCCD, or for the TEAL interface.

Warning: Do not use with SBC MAMA images.

3.1 Examples

In Python without TEAL:

```
>>> from acstools import acsccd
>>> acsccd.acsccd('*raw.fits')
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from acstools import acsccd
>>> teal.teal('acsccd')
```

In Pyraf:

```
--> import acstools
--> epar acsccd
```

acstools.acsccd.**acsccd**(*input*, *exec_path=''*, *time_stamps=False*, *verbose=False*, *quiet=False*)

Run the acsccd.e executable as from the shell.

Expect input to be `*_raw.fits`. Output is automatically named `*_blv_tmp.fits`.

Note: Calibration flags are controlled by primary header.

Warning: Do not use with SBC MAMA images.

Parameters

input : str or list of str

Input filenames in one of these formats:

- a single filename ('j1234567q_raw.fits')
- a Python list of filenames

- a partial filename with wildcards ('*raw.fits')
- filename of an ASN table ('j12345670_asn.fits')
- an at-file (`@input`)

exec_path : str, optional

The complete path to ACSCCD executable. If not given, run ACSCCD given by 'acsccd.e'.

time_stamps : bool, optional

Set to True to turn on the printing of time stamps.

verbose : bool, optional

Set to True for verbose output.

quiet : bool, optional

Set to True for quiet output.

CHAPTER

FOUR

ACSCTE

The acscte module contains a function `acscte` that calls the ACSCTE executable. Use this function to facilitate batch runs of ACSCTE, or for the TEAL interface.

Warning: Only use with WFC full-frame images.

4.1 Examples

In Python without TEAL:

```
>>> from acstools import acscte
>>> acscte.acscte('*blv_tmp.fits')
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from acstools import acscte
>>> teal.teal('acscte')
```

In Pyraf:

```
--> import acstools
--> epar acscte
```

acstools.acscte.**acscte**(*input*, *exec_path=''*, *time_stamps=False*, *verbose=False*, *quiet=False*, *single_core=False*)

Run the acscte.e executable as from the shell.

Expect input to be `*_blv_tmp.fits`. Output is automatically named `*_blc_tmp.fits`.

Note: Calibration flags are controlled by primary header.

Warning: Only use with WFC full-frame images.

Parameters

input : str or list of str

Input filenames in one of these formats:

- a single filename ('j1234567q_blv_tmp.fits')
- a Python list of filenames

- a partial filename with wildcards ('*blv_tmp.fits')
- filename of an ASN table ('j12345670_asn.fits')
- an at-file (`@input`)

exec_path : str, optional

The complete path to ACSCTE executable. If not given, run ACSCTE given by 'acscte.e'.

time_stamps : bool, optional

Set to True to turn on the printing of time stamps.

verbose : bool, optional

Set to True for verbose output.

quiet : bool, optional

Set to True for quiet output.

single_core : bool, optional

CTE correction in ACSCTE will by default try to use all available CPUs on your computer. Set this to True to force the use of just one CPU.

CHAPTER

FIVE

ACSREJ

The acsrej module contains a function `acsrej` that calls the ACSREJ executable. Use this function to facilitate batch runs of ACSREJ, or for the TEAL interface.

5.1 Examples

In Python without TEAL:

```
>>> from acstools import acsrej
>>> acsrej.acsrej('*flt.fits', 'combined_image.fits')
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from acstools import acsrej
>>> teal.teal('acsrej')
```

In Pyraf:

```
--> import acstools
--> epar acsrej
```

acstools.acsrej.**acsrej**(*input*, *output*, *exec_path=''*, *time_stamps=False*, *verbose=False*, *shadcorr=False*, *crrejtab=''*, *crmask=False*, *scalense=None*, *initgues=''*, *skysub=''*, *crsigmas=''*, *crradius=None*, *crthresh=None*, *badinpdq=None*, *newbias=False*)

Run the acsrej.e executable as from the shell.

Parameters

input : str or list of str

Input filenames in one of these formats:

- a Python list of filenames
- a partial filename with wildcards ('*flt.fits')
- filename of an ASN table ('j12345670_asn.fits')
- an at-file (`@input`)

output : str

Output filename.

exec_path : str, optional

The complete path to ACSREJ executable. If not given, run ACSREJ given by 'acsrej.e'.

time_stamps : bool, optional

Set to True to turn on the printing of time stamps.

verbose : bool, optional

Set to True for verbose output.

shadcorr : bool, optional

Perform shutter shading correction. If this is False but SHADCORR is set to PERFORM in the header of the first image, the correction will be applied anyway. Only use this with CCD image, not SBC MAMA.

crrejtab : str, optional

CRREJTAB to use. If not given, will use CRREJTAB given in the primary header of the first input image.

crmask : bool, optional

Flag CR-rejected pixels in input files. If False, will use CRMASK value in CRREJTAB.

scalense : float, optional

Multiplicative scale factor (in percents) applied to noise. Acceptable values are 0 to 100, inclusive. If None, will use SCALENSE from CRREJTAB.

initgues : {'med', 'min'}, optional

Scheme for computing initial-guess image. If not given, will use INITGUES from CRREJTAB.

skysub : {'none', 'mode'}, optional

Scheme for computing sky levels to be subtracted. If not given, will use SKYSUB from CRREJTAB.

crsigmas : str, optional

Cosmic ray rejection thresholds given in the format of 'sig1,sig2,...'. Number of sigmas given will be the number of rejection iterations done. At least 1 and at most 20 sigmas accepted. If not given, will use CRSIGMAS from CRREJTAB.

crradius : float, optional

Radius (in pixels) to propagate the cosmic ray. If None, will use CRRADIUS from CRREJTAB.

crthresh : float, optional

Cosmic ray rejection propagation threshold. If None, will use CRTHRESH from CRREJTAB.

badinpdq : int, optional

Data quality flag used for cosmic ray rejection. If None, will use BADINPDQ from CRREJTAB.

newbias : bool, optional

ERR is just read noise, not Poisson noise. This is used for BIAS images.

CHAPTER

SIX

ACS2D

The acs2d module contains a function `acs2d` that calls the ACS2D executable. Use this function to facilitate batch runs of ACS2D, or for the TEAL interface.

6.1 Examples

In Python without TEAL:

```
>>> from acstools import acs2d
>>> acs2d.acs2d('*blv_tmp.fits')
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from acstools import acs2d
>>> teal.teal('acs2d')
```

In Pyraf:

```
--> import acstools
--> epar acs2d
```

acstools.acs2d.**acs2d**(*input*, *exec_path=''*, *time_stamps=False*, *verbose=False*, *quiet=False*, *dqicorr=False*, *glincorr=False*, *lflgcorr=False*, *darkcorr=False*, *flshcorr=False*, *flatcorr=False*, *shadcorr=False*, *photcorr=False*)

Run the acs2d.e executable as from the shell.

Output is automatically named based on input suffix:

INPUT	OUTPUT	EXPECTED DATA
*_raw.fits	*_flt.fits	SBC image.
*_blv_tmp.fits	*_flt.fits	ACSCCD output.
*_blc_tmp.fits	*_flc.fits	ACSCCD output with PCTECORR.
*_crj_tmp.fits	*_crj.fits	ACSREJ output.
*_crc_tmp.fits	*_crc.fits	ACSREJ output with PCTECORR.

Parameters

input : str or list of str

Input filenames in one of these formats:

- a single filename ('j1234567q_blv_tmp.fits')
- a Python list of filenames

- a partial filename with wildcards ('*blv_tmp.fits')
- filename of an ASN table ('j12345670_asn.fits')
- an at-file (`@input`)

exec_path : str, optional

The complete path to ACS2D executable. If not given, run ACS2D given by 'acs2d.e'.

time_stamps : bool, optional

Set to True to turn on the printing of time stamps.

verbose : bool, optional

Set to True for verbose output.

quiet : bool, optional

Set to True for quiet output.

dqicorr, glincorr, lflgcorr, darkcorr, flshcorr, flatcorr, shadcorr, photcorr : bool, optional

Enable XXXXCORR. If all False, will set all but FLSHCORR and SHADCORR to PERFORM. If any is True, will set that to PERFORM and the rest to OMIT. GLINCORR and LFLGCORR are used for SBC MAMA only. FLSHCORR and SHADCORR are used for CCD only.

CHAPTER

SEVEN

ACSSUM

The acssum module contains a function `acssum` that calls the ACSSUM executable. Use this function to facilitate batch runs of ACSSUM, or for the TEAL interface.

7.1 Examples

In Python without TEAL:

```
>>> from acstools import acssum
>>> acssum.acssum('*flt.fits', 'combined_image.fits')
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from acstools import acssum
>>> teal.teal('acssum')
```

In Pyraf:

```
--> import acstools
--> epar acssum
```

`acstools.acssum.`**`acssum`**(*input*, *output*, *exec_path=''*, *time_stamps=False*, *verbose=False*, *quiet=False*)

Run the acssum.e executable as from the shell.

Parameters

input : str or list of str

Input filenames in one of these formats:

- a Python list of filenames
- a partial filename with wildcards ('*flt.fits')
- filename of an ASN table ('j12345670_asn.fits')
- an at-file (`@input`)

output : str

Output filename. If *output* is '' and *input* is '*_asn.fits', *output* will be automatically set to '*_sfl.fits'. Otherwise, it is an error not to provide a specific *output*.

exec_path : str, optional

The complete path to ACSSUM executable. If not given, run ACSSUM given by 'acssum.e'.

time_stamps : bool, optional

Set to True to turn on the printing of time stamps.

verbose : bool, optional

Set to True for verbose output.

quiet : bool, optional

Set to True for quiet output.

CHAPTER

EIGHT

ACS PIXEL CTE CORRECTION

This task implements a pixel-based CTE correction for calibrated ACS/WFC images. Functions to apply pixel-based CTE correction to ACS images.

The algorithm implemented in this code was described in detail by [Anderson] as available online at:

http://adsabs.harvard.edu/abs/2010PASP..122.1035A

Note:

- This code only works for ACS/WFC but can be modified to work on other detectors.
- It was developed for use with full-frame GAIN=2 FLT images as input.
- It has not been fully tested with any other formats.
- Noise is slightly enhanced in the output (see [Anderson]).
- This code assumes a linear time dependence for a given set of coefficients.
- This algorithm does not account for traps with very long release timescale but it is not an issue for ACS/WFC.
- This code also does not account for second-exposure effect.
- Multi-threading support was not implemented in this version.

8.1 Optional preprocessing for nonstandard FLT input

If you are not using a fully calibrated FLT image as input, you might also need to do one or more of the following before running the task:

- Convert image to unit of electrons.
- For combined image (e.g., superdark), set *noise_model* to 0.
- **Primary FITS header (EXT 0) must have these keywords populated:**
  - ROOTNAME
  - INSTRUME (must be ACS)
  - DETECTOR (must be WFC)
  - CCDAMP (ABCD, AD, BC, A, B, C, or D)
  - EXPSTART

  - ATODGNA, ATODGNB, ATODGNC, ATODGND

- **SCI FITS header (EXT 1 or 4) must have these keywords populated:**
  - EXTNAME (must be SCI)
  - EXTVER (1 or 2)
- **ERR FITS header (EXT 2 or 5) must have these keywords populated:**
  - EXTNAME (must be ERR)
  - EXTVER (1 or 2)
- **DQ FITS header (EXT 3 or 6) must have these keywords populated:**
  - EXTNAME (must be DQ)
  - EXTVER (1 or 2)

8.1.1 Examples

To correct a set of ACS FLT images, with one new CTE-corrected image for each input.

```
>>> from acstools import PixCteCorr
>>> PixCteCorr.CteCorr('j*q_flt.fits')
```

Using the TEAL GUI.

```
>>> from acstools import PixCteCorr
>>> from stsci.tools import teal
>>> teal.teal('PixCteCorr')
```

From within PyRAF:

```
--> from acstools import PixCteCorr
--> epar PixCteCorr
```

8.1.2 References

acstools.PixCteCorr.**CteCorr**(*input*, *outFits=''*, *read_noise=None*, *noise_model=None*, *oversub_thresh=None*, *sim_nit=None*, *shift_nit=None*)

Run all the CTE corrections on all the input files.

This function simply calls `YCte` on each input image parsed from the *input* parameter, and passes all remaining parameter values through unchanged.

Parameters

input : str or list of str

name of FLT image(s) to be corrected. The name(s) can be specified either as:

•a single filename ('j1234567q_flt.fits')

•a Python list of filenames

•a partial filename with wildcards ('*flt.fits')

•filename of an ASN table ('j12345670_asn.fits')

- an at-file (`@input`)

outFits : str

USE DEFAULT IF 'input' HAS MULTIPLE FILES. CTE corrected image in the same directory as input. If not given, will use ROOTNAME_cte.fits instead. Existing file will be overwritten.

read_noise : float, optional

Read noise level. If None, takes value from PCTETAB header RN_CLIP keyword.

noise_model : {0, 1, 2, None}, optional

Noise mitigation algorithm. If None, takes value from PCTETAB header NSEMODEL keyword.

0: No smoothing 1: Normal smoothing 2: Strong smoothing

oversub_thresh : float, optional

Pixels corrected below this value will be re-corrected. If None, takes value from PCTETAB header SUBTHRSH keyword.

sim_nit : int, optional

Number of times readout simulation is performed per column. If None, takes value from PCTETAB header SIM_NIT keyword.

shift_nit : int, optional

Number of times column is shifted during simulated readout. If None, takes value from PCTETAB header SHFT_NIT keyword.

`acstools.PixCteCorr.`**`YCte`**(*inFits*, *outFits=''*, *read_noise=None*, *noise_model=None*, *oversub_thresh=None*, *sim_nit=None*, *shift_nit=None*)

Apply correction for parallel CTE loss.

Input image that is already de-striped is desired but not compulsory. Using image with striping will enhance the stripes in output. Calibrations that have been applied to FLT should not significantly affect the result.

Parameters

inFits : str

FLT image to be corrected.

outFits : str

CTE corrected image in the same directory as input. If not given, will use ROOTNAME_cte.fits instead. Existing file will be overwritten.

read_noise, noise_model, oversub_thresh, sim_nit, shift_nit : see `CteCorr`

Notes

- EXT 0 header will be updated. ERR arrays will be added in quadrature with 10% of the correction. DQ not changed.
- Does not work on RAW but can be modified to do so.

Examples

Correct a single FLT image and write output to 'j12345678_cte.fits':

```
>>> import PixCteCorr
>>> PixCteCorr.YCte('j12345678_flt.fits')
```

acstools.PixCteCorr.**AddYCte**(*infile*, *outfile*, *shift_nit=None*, *units=None*)
Add CTE blurring to input image using an inversion of the CTE correction code.

Note: No changes are made to the error or data quality arrays.

Data should not have bias or prescan regions.

Image must have PCTETAB, DETECTOR, and EXPSTART header keywords, as well as gain information if the image is in counts.

Parameters

infile : str

Filename of image to be blurred. Should have the PCTETAB header keyword pointing to the PCTETAB reference file.

outfile : str

Filename of blurred output image.

shift_nit : int, optional

Number of times column is shifted during simulated readout. If None, takes value from PCTETAB header SHFT_NIT keyword.

units : {None,'electrons','counts'}, optional

If 'electrons', the input image is assumed to have units of electrons and no gain operations are performed. If 'counts', the data are assumed to be in DN and they are converted to electrons before CTE blurring is performed. The ATODGN* keywords from the primary header are used for the conversions. If None, the BUNIT keyword from the science extension headers is used to set the unit behavior. Defaults to None.

Raises

ValueError :

If the units keyword is not a valid value.

acstools.PixCteCorr.PixCteError :

If the input image comes from an imcompatible detector.

8.2 Version

acstools.PixCteCorr.**__version__ = '1.2.3'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

acstools.PixCteCorr.**__vdate__ = '13-Aug-2013'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

acstools.PixCteCorr.**ACS_CTE_NAME = 'PixelCTE 2012'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

acstools.PixCteCorr.**ACS_CTE_VER = '3.3'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

8.3 Authors

Original Fortran program by Jay Anderson.

Conversion to Python done by (ordered by last name): Matt Davis, Warren Hack, and Pey Lian Lim

CHAPTER

NINE

ACS DESTRIPE

Remove horizontal stripes from ACS WFC post-SM4 data.

9.1 Examples

In Python without TEAL:

```
>>> from acstools import acs_destripe
>>> acs_destripe.clean('uncorrected_flt.fits', 'csck', clobber=False,
...                    maxiter=15, sigrej=2.0)
```

In Python with TEAL:

```
>>> from acstools import acs_destripe
>>> from stsci.tools import teal
>>> teal.teal('acs_destripe')
```

In Pyraf:

```
--> import acstools
--> teal acs_destripe
```

From command line:

```
% ./acs_destripe [-h][-c] input output [maxiter # [sigrej #]]
```

acstools.acs_destripe.**clean**(*input*, *suffix*, *maxiter=15*, *sigrej=2.0*, *clobber=False*)

Remove horizontal stripes from ACS WFC post-SM4 data.

Note: Input data must be an ACS/WFC FLT image, with 2 SCI extensions. Does not work on RAW image.

Uses the flatfield specified by the image header keyword PFLTFILE. If keyword value is 'N/A', as is the case with biases and darks, then unity flatfield is used.

Uses the dark image specified by the image header keyword DARKFILE. If keyword value is 'N/A', then dummy dark with zeroes is used.

Parameters

input : str or list of str

Input filenames in one of these formats:

- a Python list of filenames

- a partial filename with wildcards ('*flt.fits')
- filename of an ASN table ('j12345670_asn.fits')
- an at-file (`@input`)

suffix : str

The string to use to add to each input file name to indicate an output product. This string will be appended to the _flt suffix in each input filename to create the new output filename. For example, setting *suffix='csck'* will create '*_flt_csck.fits' images.

maxiter : int

This parameter controls the maximum number of iterations to perform when computing the statistics used to compute the row-by-row corrections.

sigrej : float

This parameters sets the sigma level for the rejection applied during each iteration of statistics computations for the row-by-row corrections.

clobber : bool

Specify whether or not to 'clobber' (delete then replace) previously generated products with the same names.

9.2 Global Variables

`acstools.acs_destripe.`**`MJD_SM4 = 54967`**

int(x=0) -> int or long int(x, base=10) -> int or long

Convert a number or string to an integer, or return 0 if no arguments are given. If x is floating point, the conversion truncates towards zero. If x is outside the integer range, the function returns a long instead.

If x is not a number or if base is given, then x must be a string or Unicode object representing an integer literal in the given base. The literal can be preceded by '+' or '-' and be surrounded by whitespace. The base defaults to 10. Valid bases are 0 and 2-36. Base 0 means to interpret the base from the string as an integer literal. >>> int('0b100', base=0) 4

9.3 Version

`acstools.acs_destripe.`**`__version__ = '0.4.1'`**

str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`acstools.acs_destripe.`**`__vdate__ = '05-Feb-2014'`**

str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

9.4 Author

`acstools.acs_destripe.`**`__author__ = 'Norman Grogin, STScI, March 2012.'`**

str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

CHAPTER

TEN

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

BIBLIOGRAPHY

[Anderson] Anderson J. & Bedin, L.R., 2010, PASP, 122, 1035

PYTHON MODULE INDEX

a

PYTHON MODULE INDEX

a

INDEX

Symbols

A

C

M

Y