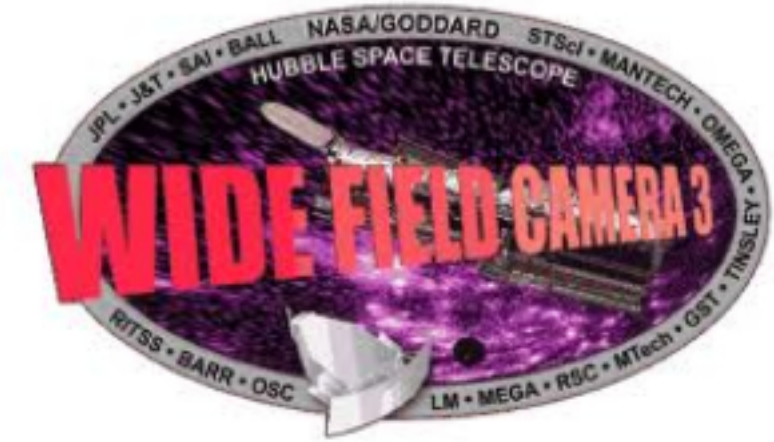


WFC3tools Documentation

Release v1.3.2

wfc3tools

September 22, 2016

Contents

See the team website http://www.stsci.edu/hst/wfc3/ and the WFC3 Data Handbook (http://www.stsci.edu/hst/wfc3/documents/handbooks/currentDHB/) for more information.

CHAPTER 1

Pipeline Software

1.1 calwf3

`calwf3` is the name of the main executable which processes data from the WFC3 instrument onboard Hubble taken with either the UVIS or IR detectors. The code automatically calls the appropriate tasks, but users may also run the tasks independently if they desire special processing for their datasets. *wf3cte*, *wf3ccd* and *wf32d* are used for processing UVIS images, while IR image processing is done with *wf3ir*. The *wf3rej* program is used for both UVIS and IR images to combine multiple exposures contained in a CR-SPLIT or REPEAT-OBS set. Figure 1.3 is the flow diagram for the UVIS pipeline as a whole, while Figure 1.4 contains the flow for the IR pipeline.

During automatic pipeline processing by the STScI archive, `Astrodrizzle` follows `calwf3`. All calibrated images are corrected for geometric distortion correction and associated sets of dithered images are combined into a single product. See the WFC3 Data Handbook for more information, or Astrodrizzle (http://www.stsci.edu/hst/HST_overview/drizzlepac/) .

1.1.1 Where to Find calwf3

`calwf3` is part of HSTCAL package, which can be downloaded from http://www.stsci.edu/institute/software_hardware/stsdas/download-stsdas and is installed along with the STScI distributed package Ureka.

A detailed description of the improved `calwf3`, Version 3.3, which is more generally referred to as the UVIS2.0 update, will be available in a future publication of WFC3 Data Handbook and several ISRs which will accompany the update.

The current WFC3 Data Handbook can be found at http://www.stsci.edu/hst/wfc3/documents/handbooks/currentDHB/ . In the meantime, if you have questions not answered in this documentation, please contact STScI Help Desk (help[at]stsci.edu).

1.1.2 Running calwf3

`calwf3` can be run on a single input raw file or an asn table listing the members of an associtaion. When processing an association, it retrieves calibration switch and reference file keyword settings from the first image listed in the asn table. `calwf3` does not accept a user-defined list of input images on the command line (e.g. `*raw.fits` to process all raw files in the current directory).

The *wf3ccd*, *wf32d*, *wf3cte* and *wf3ir* tasks on the other hand, will accept such user-defined input file lists, but they will not accept an association table(asn) as input.

Running `calwf3` from a python terminal

In Python without TEAL:

```
>>> from wfc3tools import calwf3
>>> calwf3(filename)
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from wfc3tools import calwf3
>>> teal.teal('calwf3')
```

In Pyraf:

```
>>> import wfc3tools
>>> epar calwf3
```

Running many files at the same time

The recommended method for running `calwf3` in batch mode is to use Python and the `wfc3tools` package in the STSDAS distribution (http://www.stsci.edu/institute/software_hardware/stsdas/download-stsdas) .

For example:

```
from wfc3tools import calwf3
from glob import glob

for fits in glob('j*_raw.fits'):
    calwf3(fits)
```

Note: `calwf3()` may raise a RuntimeError if the underlying `calwf3.e` program fails with a non-zero exit code. Review the text output during the calibration call for hints as to what went wrong.

Displaying output from calwf3 in a Jupyter Notebook

When calling `calwf3` from a Jupyter notebook, informational text output from the underlying `calwf3.e` program will be passed through `print` as the calibration runs by default, and show up in the user's cell. This behavior can be customized by passing your own function as the `log_func` keyword argument to `calwf3`. As output is read from the underlying program, the `calwf3` Python wrapper will call `log_func` with the contents of each line. (`print` is an obvious choice for a log function, but this also provides a way to connect `calwf3` to the Python logging system by passing the `logging.debug` function or similar.)

If `log_func=None` is passed, informational text output from the underlying program will be ignored, but the program's exit code will still be checked for successful completion.

Command Line Options

`calwf3` can also be called directly from the OS command line:

```
>>> calwf3.e iaa001kaq_raw.fits [command line options]
```

The command line executable only accepts one file at a time, but you can use os tools like `awk` to process everything in a directory:

```
>>> ls *raw.fits | awk '{print "calwf3.e",$1}' | csh
```

The following options are available

- t: Print verbose time stamps.
- s: Save temporary files.
- v: Turn on verbose output.
- d: Turn on debug output.
- q: Turn on quiet output.
- r: Print the current software version number (with full revision information)
- –version: Print the current software version number only

Types of files used as input to calwf3

- _asn file: name of an association table
- _raw file: name of an individual, uncalibrated exposure
- _crj file: name of any sub-product from an association table

While both CR-SPLIT and REPEAT-OBS exposures from an association get combined using `calwf3`, dithered observations from an association do not.

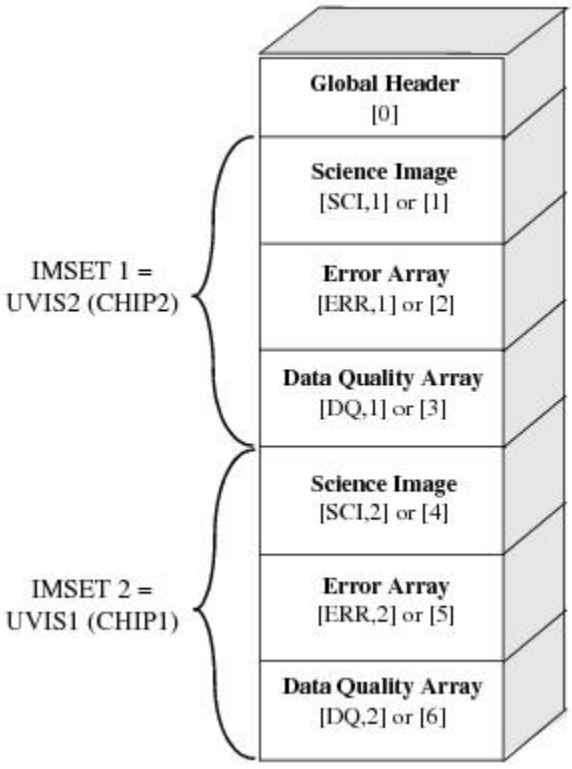


Fig. 1.1: UVIS data raw file format

- The science image contains the data from the focal plane array detectors.

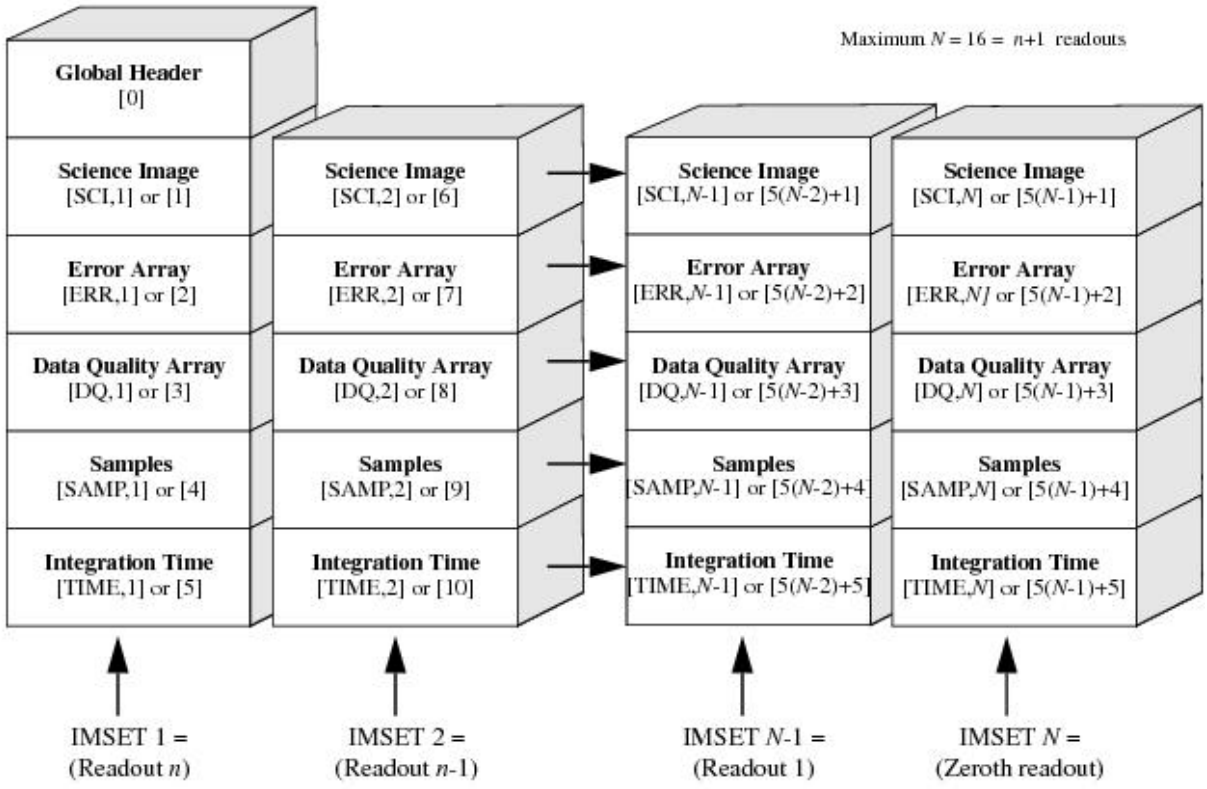


Fig. 1.2: IR data raw file format

- The error array contains an estimate of the statistical uncertainty associated with each corresponding science image pixel
- The data quality array contains independent flags indicating various status and problem conditions associated with each corresponding pixel in the science image
- The sample array (IR ONLY) contains the number of samples used to derive the corresponding pixel values in the science image.
- The time array (IR ONLY) contains the effective integration time associated with each corresponding science image pixel value.

1.1.3 Parameters

input [str] Name of input files

- a single filename (`iaa012wdq_raw.fits`)
- a Python list of filenames
- a partial filename with wildcards (`\*raw.fits`)
- filename of an ASN table (`\*asn.fits`)
- an at-file (`@input`)

output: str Name of the output FITS file.

printtime: bool print a detailed time stamp

save_tmp: bool save temporary files

debug: bool print optionsl debugging statements

parallel: bool run the code with OpemMP parallel processing turned on for the UVIS CTE correction

log_func: func() if not specified, the print function is used for logging to facilitate use in the jupyter notebook

verbose: bool, optional Print verbose time stamps?

quiet: bool, optional Print messages only to trailer file?

Types of output file from calwf3

The suffixes used for WFC3 raw and calibrated data products closely mimic those used by ACS and NICMOS:

SUFFIX	DESCRIPTION	UNITS
_raw	raw data	DN
_rac	UVIS CTE corrected raw data, no other calibration	DN
_asn	association file for observation set	
_spt	telescope and WFC3 telemetry and engineering data	
_blv_tmp	overscan-trimmed UVIS exposure	DN
_blc_tmp	overscan0trimmed UVIS, CTE corrected exposure	DN
_crj_tmp	uncalibrated, cosmic-ray rejected combined	DN
_crc_tmp	uncalibrated, cosmic-rat rejected, CTE cleaned	DN
_ima	calibrated intermediate IR multiaccum image	e^-/s
_flt	UVIS calibrated exposure	e^-
_flc	UVIS calibrated exposure including CTE correction	e^-
_flt	IR calibrated exposure	e^-/s
_crj	UVIS calibrated, cosmic ray rejected image	e^-
_crj	IR calibrated, cosmic ray rejected image	e^-/s
_crc	UVIS calibrated, cr rejected, cte cleaned image	e^-
.tra	trailer file, contains processing messages	

** DRZ and DRC products are produced with Astrodrizzle, see Astrodrizzle (http://www.stsci.edu/hst/HST_overview/drizzlepac/) **

Keyword Usage

`calwf3` processing is controlled by the values of keywords in the input image headers. Certain keywords, referred to as calibration switches, are used to control which calibration steps are performed. Reference file keywords indicate which reference files to use in the various calibration steps. Users who which to perform custom reprocessing of their data may change the values of these keywords in the _raw FITS file headers and then rerun the modified file through `calwf3`. See the WFC3 Data Handbook (http://www.stsci.edu/hst/wfc3/documents/handbooks/currentDHB/wfc3_Ch25.html) for a more complete description of these keywords and their values.

orphan

1.1.4 UVIS Pipleline

As of version 3.3, the calwf3 pipeline processes all UVIS data twice, once with the CTE correction applied as the first step, and a second time without the CTE correction. A short description of the calibration steps, in the order they are performed:

- Calculate and remove and CTE found in the image (PCTECORR)
- Calculate a noise model for each pixel and record in the error array (ERR) of the image (NOISCORR)
- Initialize the data quality(DQ) array of the image (DQICORR)
- Correct fo A-toD conversion errors where necessary, currently skipped (ATODCORR)
- Subtract bias level determined from overscan regions (BLEVCORR)
- Subtract the bias image (BIASCORR)
- Detect and record SINK pixels in the DQ mask (performed if DQICORR is set to PERFORM)
- Perform unit conversion
- Subtract the post-flash image if applicable (FLSHCORR)
- Scale and subtract the dark image (DARKCORR)

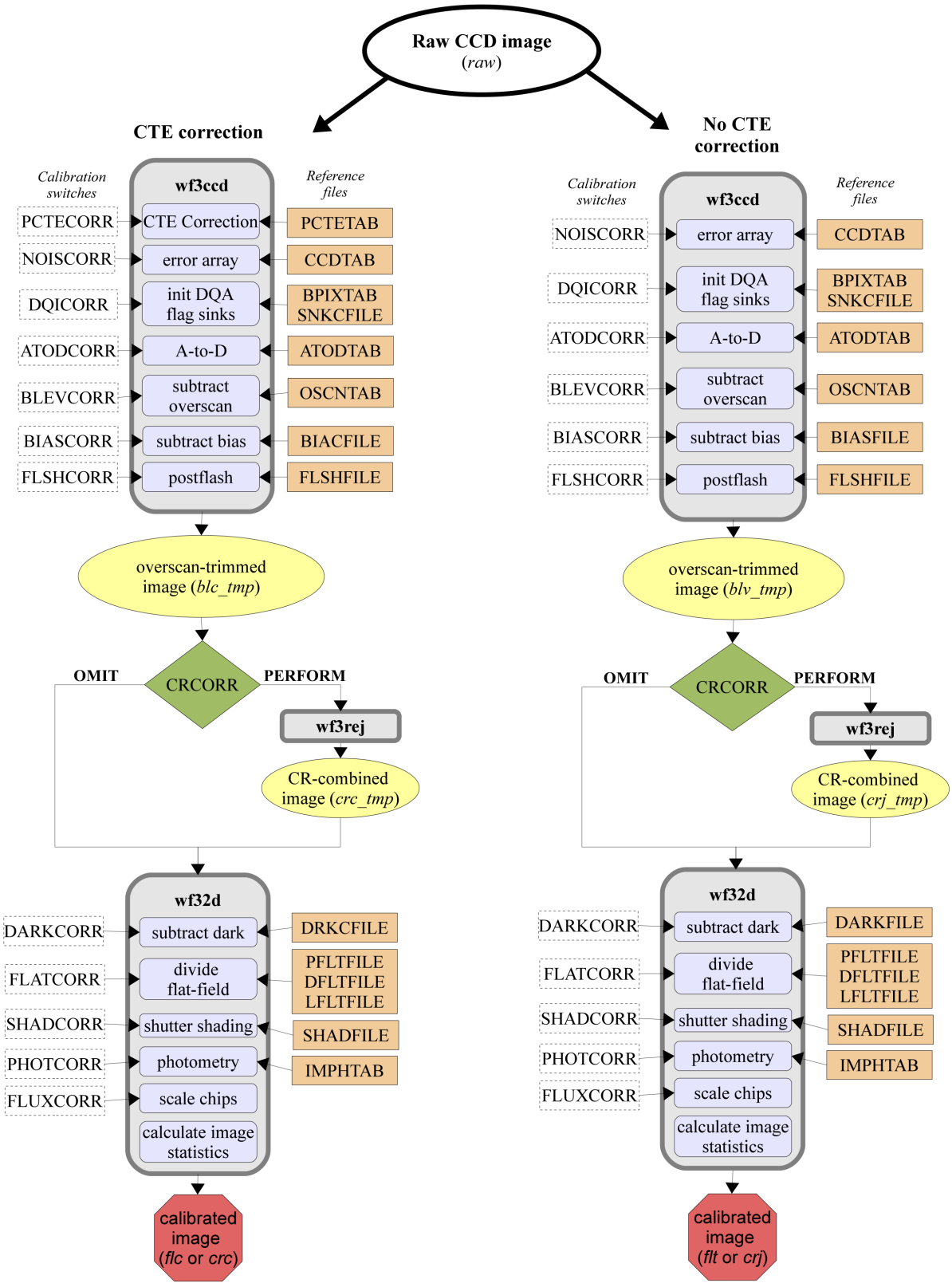


Fig. 1.3: Flow diagram for `calwf3` data. *wf3cte* occurs as the very first step, before *wf3ccd*.

- Perform flatfielding (FLATCORR)
- Perform shutter-shading correction where necessary, currently skipped (SHADCORR)
- Populate photometric header keywords (PHOTCORR)
- Correct chips to be on the same zeropoint (FLUXCORR)
- Calculate image statistics for the header

Correction For Charge Transfer Efficiency (PCTECORR)

The charge transfer (CTE) of the UVIS detector has been declining over time as on-orbit radiation damage creates charge traps in the CCDs. Faint sources in particular can suffer large flux losses or even be lost entirely if observtions are not planned and analyzed carefully. The CTE depends on the morphology of the source, the distribution of electrons in the field of view, and the population of charge traps in the detector column between the source and the transfer register. Further details of the current understanding of the state of the WFC3/UVIS charge transfer efficieny (CTE) are presented in Section 5 (http://www.stsci.edu/hst/wfc3/documents/handbooks/currentIHB/c05_detector5.html#392274) of the data handbook as well as on the WFC3 CTE webpage (http://www.stsci.edu/hst/wfc3/ins_performance/CTE/) . The PCTECORR step aims to mitigate the flux loss incurred from CTE.

More information on this part of the pipeline can be found in the *wf3cte* documentation.

Error Array Initialization

The image error array is initialized. The function examines the RR extension of the input data to determine the state of the array. The input _raw image contains an empty ERR array. If the ERR array has already been expanded and contains values other than zero, then this function does nothing. Otherwise it will intialize the ERR array by assigning pixel values based on a simple noise model. The noise model uses the science (SCI) array and for each pixel calculates the error value σ in units of DN:

$$\sigma_{CCD} = \sqrt{(SCI - bias)/(gain) + (readnoise/gain)^2}$$

The CCDTAB reference file contains the bias, gain and readnoise values used for each CCD aplifier quadrant used in this calculation. The table contains one row for each configuration that can be used during readouse, which is uniquely identified by the list of amplifiers (replicated in the CCDAMP header keyword), the particular chip being read out (CCDCHIP), the commanded gain (CCDGAIN), the commanded bias offset level (CCDOFST) and the pixel bin size (BINAXIS). These commanded values are used to find the table row that matches the characteristics of the image that is being processed and reads each amplifiers characteristics, including readnoise (READNSE), A-to-D gain (ATODGN) and the mean bias level (CCDBIAS).

Data Quality Array Initialization (DQICORR)

This step initializas the data quality array by reading a table of known bad pixels for the detector, as stored in the Bad Pixel reference table BPIXTAB. The types of bad pixels that can be flagged are:

NAME	VALUE	DESCRIPTION
GOODPIXEL	0	OK
SOFTERR	1	Reed-Solomon decoding error
DATALOST	2	data replaced by fill value
DETECTORPROB	4	bad detector pixel or beyond aperture
DATAMASKED	8	masked by occulting bar
HOTPIX	16	hot pixel
CTETAIL	32	UVIS CTE tail
WARMPIX	64	warm pixel
BADBIAS	128	bad bias value
SATPIXEL	256	full-well or a-to-d saturated pixel
BADFLAT	512	bad flatfield value
TRAP	1024	UVIS charge trap, SINK pixel
ATODSAT	2048	a-to-d saturated pixel
ne TBD	4096	reserved for Multidrizzle CR rej
DATAREJECT	8192	rejected during image combination UVIS, IR CR rejection
CROSSTALK	16384	ghost or crosstalk
RESERVED2	32768	can't use

Sink Pixel Detection and Marking

Sink pixels are a type of image defect. These pixels contain a number of charge traps and under-report the number of electrons that were generated in them during an exposure. These pixels can have an impact on nearby upstread or downstream pixels, though they often only impact one or two pixels when the background is high, they can impact up to 10 pixels if the background is low.

Flagging of SINK pixels in the DQ extension of calibrated images is controlled with the DQICORR header keyword, happens after the bias correction has been performed, and is done in the amp-rotated CDAB full image format used and described in the CTE correction. When set to perform, the sink pixels are located and flagged with help from the SNKCFILE reference image. Given the reference image, the procedure for flagging the sink pixel in science data involves:

- Extract the MJD of the science exposure
- Go through the reference image pixel by pixel looking for those pixels with values greater than 999, which indicates that the current pixel is a sink pixel. The value of this pixel in the reference file corresponds to the date at which this pixel exibited the sink behavior.
- If the turn on date of the sink pixel is after the exposure date of the science image, then we ignore the sink pixel in this exposure and move on to the next pixel
- If the turn on date of the sink pixel is before the exposure date of the science image, then this science pixel was compromised at the time of the exposure.The corresponding DQ extension pixel for this science pixel is flagged with the "charge trap" flag of 1024.
- If the pixel "below" the sink pixel in the long format image has a value of -1 in the reference image, then it is also flagged with the "charge trap" value in the DQ extension. We then proceed vertically "up" from the sink pixel and compare each pixel in the reference file to the value of the sink pixel in the science exposure at hand. If the value of the sink pixel in the exposure is below the value of the upstream pixel in the reference image, we flag that pixel with the "charge trap" value in the DQ extension. We continue to flag pixels until the value of the pixel in the reference image is zero or until the value of the sink pixel in the exposure is greater than the value of the upstream pixel in the reference image.

WFC3 ISR 2014-19 (http://www.stsci.edu/hst/wfc3/documents/ISRs/WFC3-2014-19.pdf) has a detailed analysis on detection of the sink pixels, while the strategy for flagging them is discussed in WFC3 ISR 2014-22 (http://www.stsci.edu/hst/wfc3/documents/ISRs/WFC3-2014-22.pdf) .

Sink pixels are currently only flagged in full frame science images, a future release of `calwf3` will also perform flagging in subarray images. **The pipeline currently does no further analysis or correction on pixels which have been flagged as affected by sink pixels**

Unit Conversion to Electrons

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

Bias Correction (BIASCORR)

This step subtracts the two dimensional bias struction from the image using the superbias reference image listed in the header keyword BIASFILE. The dimensions of the image are used to distinguish between full and sub-array images. Because the bias image is already overscan-subtracted, it will have a mean pixel value of less than one. The BIASFILE has the same dimensions as a full-size science image, complete with overscan regions. Only after completion of *wf3ccd* are the science images trimmed to their final calibrated size. The same reference image is used for full-frame and subarray images, `calwf3` will extract the matching region from the full-size bias file and apply it to the subarray image.

Overscan Bias Correction (BLEVCORR)

The location of the overscan regions in a raw image varies, depending upon the type of readout that is performed. The overscan regions are used to monitor the instrument as well as provide a measure of the bias level at the time the detector was exposed. The bias level which is calculated for subtraction is done on a line-by-line basis in the image. If the image has no overscan region the BIAS level to be subtracted is obtained from the CCDTAB reference file. Otherwise, the columns to use for the calculation are referenced in the OSCNTAB reference file. A bias drift calculation is made if there are virtual overscan pixels which exist, if neither of the virtual overscan regions are specfied then the physical overscan region is used.

If there are two sections available to use for the line because only 1 amp was used then they are averaged. The parallel overscan region is split into two if there is more than one amp. If the virtual overscan is used, a straight line is fit as a function of the column number. The fit is evaluated for each line and then subtracted from the data. Iterative sigma clipping is used to reject outliers from the array of bias values.

The mean value of all the bias levels which were subtracted is recorded in the SCI extension output header in MEANBLEV.

Dark Current Subtraction (DARKCORR)

The reference file listed under the DARKFILE header keyword is used as the reference dark image.

In the UVIS, the dark image is scaled by EXPTIME and FLASHDUR.

The reference file pointed to with DARKFILE is used for the non-CTE corrected data.

The reference file pointed to with DRKCFILE is used for the CTE corrected data

Shutter Shading Correction (SHADCORR)

This step corrects the science image for differential exposure time across the detector cased by the amount of time it takes for the shutter to completely open and close, which is a potentially signiciant effect only for images with very

short exposure times (less than ~5 seconds). Pixels are corrected based on the exposure time using the relation:

$$corrected = uncorrected \times EXPTIME \div (EXPTIME + SHADFILE)$$

WFC3 tests have shown that the shutter shading effect is insignificant (< 1%), even for the shortest allowed UVIS exposure time of 0.5 seconds (see WFC3 ISR 2007-17 (http://www.stsci.edu/hst/wfc3/documents/ISRs/WFC3-2007-17.pdf)). Therefore this step is **ALWAYS set to OMIT** in `calwf3`.

Post-Flash Correction (UVIS ONLY) (FLSHCORR)

WFC3 has post-flash capability to provide a means of mitigating the effects of Charge Transfer Efficiency (CTE) degredation. When FLSHCORR=PERFORM, this routine subtracts the post-flash reference image, FLSHFILE, from the science image after DARKCORR in the WF32D step. The success of the post-flash operation during the exposure is first verified by checking the keyword FLASHSTA. The FLSHFILE is renormalized to the appropriate post-flash current level (LOW, MED, HIGH) recorded in the FLASHCUR keyword, and the flash duration (FLASHDUR) and is then subtracted from the science image. The mean value of the scaled post-flash image is written to MEANFLSH in the output SCI extension header. Different members of an association can have different values of SHUTRPOS because it varies by exposure, and this is fine for calibration because the references files are populated separately for each exposure.

KEYWORD	DESCRIPTION
FLSHDUR	is the length of time of the flash exposure
FLSHCUR	is the current that was used to the lamp as calculated by TRANS, which also calculates FLASHEXP, (ZERO, LOW, MED,HIGH)
FLSHFILE	is the flash reference file, which has an illumination pattern for each shutter
SHUTRPOS	says which shutter was used
FLASHSTA	indicates an interrupted exposure (ABORTED, SUCCESSFUL, NOT PERFORMED)
FLASHLVL	post flash level in electrons
MEANFLSH	the mean level which `calwf3` calculated and then subtracted

Futher reading:

- WFC3 Post-Flash Calibration ISR (http://www.stsci.edu/hst/wfc3/documents/ISRs/WFC3-2013-12.pdf)
- CTE-Loss Mitigation Before Data Acquisition (http://www.stsci.edu/hst/wfc3/documents/handbooks/currentIHB/c06_uvis1(

FLATCORR

Correct the image for pixel quantum efficiency using the reference image specified by the FLATFILE keyword in the header. Conversion from DN to ELECTRONS no longer depends on FLATCORR=PERFORM, all images are converted appropriately.

This actually consists of correction using up to 3 reference flat images:

- PFLTCORR: apply a pixel-to-pixel flat (ground flats)
- DFLTCORR: apply a delta flat, applies any needed changes to the small-scale PFLTFILE
- LFLTCORR: apply a low order flat, correcting for large scale sensativity variations (on-orbit)

The pipeline is currently only using the P-flats. If two or more reference files are specified, they are read in line-by-line and multiplied together to form a combined flatfield correction image.

Subarray science images use the same reference file as the full-frame images; `calwf3` will extract the appropriate region from the reference file and apply it to the subarray input image.

Photometry Keywords (PHOTCORR)

The PHOTCORR step is performed using tables of precomputed values instead of calls to SYNPHOT, it uses the reference table specified in the IMPHTTAB header keyword. Each DETECTOR uses a different table.

If you do not wish to use this feature, set the header keyword PHOTCORR to OMIT. However, if you intend to use the FLUXCORR step, then PHOTCORR must be set to PERFORM as well.

- PHOTFNU: the inverse sensitivity in units of :math: Jansky sec electron^{-1}‘
- PHOTFLAM: the inverse sensistiy in units of $ergs\ cm^{-2}A^{-1}electron^{-1}$
- PHOTPLAM: the bandpass pivot wavelength
- PHOTBW: the bandpass RMS width
- PHTFLAM1: the inverse sensistiy in units of $ergscm^{-2}A^{-1}electron^{-1}$
- PHTFLAM2: the inverse sensistiy in units of $ergscm^{-2}A^{-1}electron^{-1}$

For versions 3.3 and beyond, the value PHOTFNU is calculated specific for each UVIS chip, see the section on FLUXCORR for more information.

The SCI headers for each chip contain the PHOTFNU keyword, which is valid for its respective chip, where PHOTFNU is calculated as:

For UVIS 1: $photfnu = 3.33564e^4 * PHTFLAM1 * PHOTPLAM^2$

For UVIS 2: $photfnu = 3.33564e^4 * PHTFLAM2 * PHOTPLAM^2$

The IMPHTTAB file format for WFC3 UVIS is as follows:

EXT#	FITSNAME	FILENAME	EXTVE	DIMENS	BITPI	OBJECT
0	z7n21066i_imp	z7n21066i_imp.fits			16	
1	BINTABLE	PHOTFLAM	1	5Fx256R		
2	BINTABLE	PHOTPLAM	1	5Fx256R		
3	BINTABLE	PHOTBW	1	5Fx256R		
4	BINTABLE	PHTFLAM1	1	5Fx256R		
5	BINTABLE	PHTFLAM2	1	5Fx256R		

where each extension contains the photometry keyword information for that specific header keyword. The rows inside the tables are split on observation mode.

Flux normalization for UVIS1 and UVIS2 (FLUXCORR)

The FLUXCORR step was added in calwf3 v3.1.2 as a way to scale the UVIS chips so that the flux correction over both chips is uniform. This requires new keywords which specify new PHOTFLAM values to use for each chip as well as a keyword to specify the scaling factor for the chips. New flatfields must be used and will replace the old flatfields in CDBS but the change will not be noticable to users. Users should be aware that flatfield images used in conjunction with v3.2.1 of the software should not be used with older versions as the data, and vice versa will be scaled incorrectly.

The new keywords include:

- PHTFLAM1: The FLAM for UVIS 1

- PHTFLAM2: The FLAM for UVIS 2
- PHTRATIO: The ratio: PHTFLAM2 / PHTFLAM1, which is calculated by calwf3 and is multiplied with UVIS2 (SCI,1 in the data file)

In order for FLUXCORR to work properly the value of PHOTCORR must also be set to perform since this populates the header of the data with the keywords FLUXCORR requires to compute the PHTRATIO.

This step is performed by default in the pipeline and the PHOTFLAM keyword will be valid for both chips after the correction has been applied.

orphan

1.1.5 IR Pipeline

IR pipeline output files using the RAW file as input:

- flt.fits: output calibrated, ramp-fitted exposure produced after CRCORR has been run
- ima.fits: output ramp calibrated exposure. Remember that the signal rate recorded in each SCI extension of the ima file represents the average flux between that particular readout and the zero read.
- _crj.fits: a cosmic-ray rejected sub-product produced from images in an association table
- .tra: output text information about the processing

Data Quality Initialization (DQICORR)

Initialize the data quality array for the image using the reference file specified in its header with BPIXTAB. The DQ array is no longer updated to reflect any TDF transition during the sample. If you want to update DQ pixel values yourself before running further processing, do it after this first step has been completed, remembering that the data in this extension is always in units of UNSIGNED INTEGER. The following table lists the DQ flag values and their meanings:

NAME	VALUE	DESCRIPTION
GOODPIXEL	0	OK
SOFTERR	1	Reed-Solomon decoding error
DATALOST	2	data replaced by fill value
DETECTORPROB	4	bad detector pixel or beyond aperture
DATAMASKED	8	masked by occulting bar
BADZERO	8	deviant IR zero-read pixel
HOTPIX	16	hot pixel
UNSTABLE	32	IR unstable pixel
WARMPIX	64	warm pixel
BADBIAS	128	bad bias value
SATPIXEL	256	full-well or a-to-d saturated pixel
BADFLAT	512	bad flatfield value
SPIKE	1024	CR spike detected during cridcalc IR
ZEROSIG	2048	IR zero-read signal correction
ne TBD	4096	reserved for Multidrizzle CR rej
DATAREJECT	8192	rejected during image combination UVIS, IR CR rejection
HIGH_CURVATURE	16384	pixel has more than max CR's
RESERVED2	32768	can't use

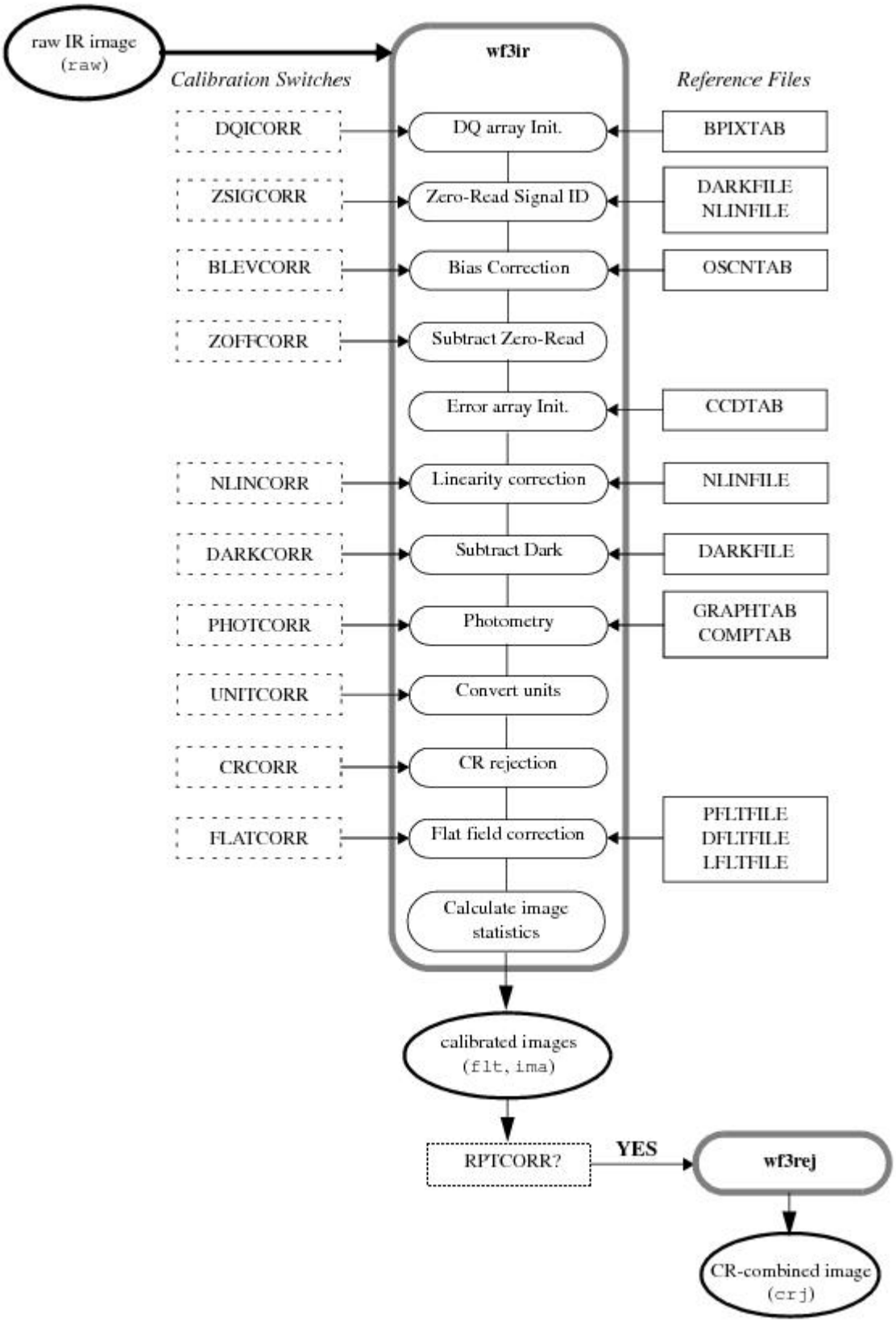


Fig. 1.4: Flow diagram for IR data using wf3ir in calwf3

Estimate the signal in the zero read (ZSIGCORR)

This step measures the signal between the super zero read in the linearity reference file (NLINFILE) and the science zero read exposure, the steps are roughly as follows:

- copy the zero sig image from the linearity reference file
- compute any subarray offsets
- subtract the super zero read reference image from the zero read science image
- compute the noise in the zero image
- pixels which contain more than ZTHRESH*noise of detected signal are flagged and that signal is passed to the NLINCORR step to help judge saturation and linearity, avoiding reference pixels.
- low signal pixels are masked out by setting them to zero
- the NLINCOR file has an extension with saturation values for each pixel which is referenced here. Pixels which are saturated in the zeroth or first reads are flagged in the DQ and the number of found saturated pixels are reported.
- This step works poorly for bright targets which are already begining to saturate in the zeroth and first reads
- This step acutally subtractes the super zero read from the science zero read instead of calculating an estimated signal based on the first read and zero read + estimated exposure time between them so that the difference in readout time for subarrays is not an issue.

Bias Correction (BLEVCORR)

This step subtracts the bias level using the reference pixels around the perimeter of the detector, the boundries fo the reference pixels are defined in the OSCNTAB reference file. There are 5 reference pixels on each end of each row, but 1 is ignored on each side, for a total of 8 being used per row. The resistent mean of the standard deviation of all the reference pixels in the image is subtracted from the entire image and the value is stored in the MEANBLEV keyword in the output image header. The reference pixels are left in place in the IMA output image through processeing, but the final FLT image has been trimmed to just the science pixels.

Zero read subtraction (ZOFFCORR)

The original zero read is subtracted from all groups in the science image, including the zeroth read itself, combining the DQ arrays with a logical OR. The ERR and SAMP arrays are unchanged and the TIME arrays are subtracted from each other. The exposure time for the group being corrected is reduced by an amount equal to the exposure time of the zero-read. At this point we've subtracted the mean bias using the reference pixels (BLEVCORR) and added back in the signal from the super zero read (done at the end of ZSIGCORR). What's left in the zero read of the science image is the superbias subtracted signal. The TIME and SAMP arrays are saved to the FLT image only AFTER the CRCORR step has been completed.

Error array initialization

The errors associated with the raw data are estimated according to the noise model for the detector which currently includes a simple combination of detector readnoise and poisson noise from the pixel. Readnoise and gain are read from the CCDTAB reference file. The ERR array continues to be summed in quadrature as the SCI array is processed. Inside the final FLT image, the ERR array is calculated by CRCORR as the calculated uncertainty of the count-rate fit to the multiaccum samples.

$$\sigma_{IR} = \frac{\sqrt{(readnoise^2) + (counts * gain)}}{gain}$$

Detector Non-linearity Correction (NLINCORR)

The integrated counts in the science images are corrected for the non-linear response of the detectors, flagging pixels which extend into saturation (as defined in the saturation extension of the NLINFILE reference image. The observed response of the detector can be represented by two regimes:

- At low and intermediate signal levels the detector response deviates from the incident flux in a way that is correctable using the following expression

$$F_c = (1 + c_1 + c_2 \times F + c_3 \times F^2 + c_4 \times F^3) \times F$$

where c1, c2, c3, and c4 are the correction coefficients, F is the uncorrected flux in DN and F_c is the corrected flux. The current form of the correction uses a third-order polynomial, but the algoritm can handle an arbitrary number of coefficients. The number of coefficients and error terms are given by the values of the NCOEFF and NERR keywords in the header of the NLINFILE.

- At high signal levels, as saturation sets in, the response becomes highly non-linear and is not correctable to a scientifically useful degree.

The signal in the zero read is temporarily added back to the zeroth read image of the science data before the linearity correction is applied and before the saturation is judged. Once the correction has been applied the signal is once again removed. This only occurs if the ZSIGCORR step is set to PERFORM. Saturation values for each pixel are stored in the NODE extension of the NLINFILE. After each group is corrected, the routine also sets saturation flags in the next group for those pixels that are flagged as saturated in the current group. This is necessary because the SCI image value of saturated pixels will sometimes start to go back down in the subsequent reads after saturation occurs, which means they could go unflagged by normal checking techniques. The SAMP and TIME arrays are not modified during this step.

The format of the linearity reference file:

EXT#	FITSNAME	FILENAME	EXTVE	DIMENS	BITPI	OBJECT
0	u1k1727mi_lin	u1k1727mi_lin.fits			-32	
1	IMAGE	COEF	1	1024x1024	-32	
2	IMAGE	COEF	2	1024x1024	-32	
3	IMAGE	COEF	3	1024x1024	-32	
4	IMAGE	COEF	4	1024x1024	-32	
5	IMAGE	ERR	1	1024x1024	-32	
6	IMAGE	ERR	2	1024x1024	-32	
7	IMAGE	ERR	3	1024x1024	-32	
8	IMAGE	ERR	4	1024x1024	-32	
9	IMAGE	ERR	5	1024x1024	-32	
10	IMAGE	ERR	6	1024x1024	-32	
11	IMAGE	ERR	7	1024x1024	-32	
12	IMAGE	ERR	8	1024x1024	-32	
13	IMAGE	ERR	9	1024x1024	-32	
14	IMAGE	ERR	10	1024x1024	-32	
15	IMAGE	DQ	1	1024x1024	-32	
16	IMAGE	NODE	1	1024x1024	-64	
17	IMAGE	ZSCI	1	1024x1024	-32	
18	IMAGE	ZERR	1	1024x1024	-32	

Dark Current Subtraction (DARKCORR)

The reference file listed under the DARKFILE header keyword is used to subtract the dark current from each sample. Due to potential non-linearities in some of the signal components, such as reet-related effecets in the first one or two reads of an exposure, the dark current subtraction is not aplied by simply scaling a generic reference dark image to the

exposure time and then subtracting it. Instead, a library of dark current images is maintained that includes darks taken in each of the available predefined multiaccum sample sequences, as well as the available sub-array readout modes. The multiaccum dark reference file is subtracted read-by-read from the stack of science image readouts so that there is an exact match in the timings and other characteristics of the dark image and the science image. The subtraction does not include the reference pixel. The ERR and DQ arrays from the reference dark file are combined with the SCI and DQ arrays from the science image, but the SAMP and TIME arrays are unchanged. The mean of the dark image is saved to the MEANDARK keyword in the output science image header.

Photometry Keywords (PHOTCORR)

The PHOTCORR step is 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 calwf3 to perform the PHOTCORR step. The format of the file for the IR detectors is:

```
EXT#  FITSNAME      FILENAME               EXTVE DIMENS       BITPI OBJECT

0     wbj1825ri_imp wbj1825ri_imp.fits                        16
1       BINTABLE    PHOTFLAM               1     5Fx38R
2       BINTABLE    PHOTPLAM               1     5Fx38R
3       BINTABLE    PHOTBW                 1     5Fx38R
```

where each extension contains the photometry keyword information for that specific header keyword. The rows in the tables are split on observation mode.

- PHOTFLAM: the inverse sensistiy in units of $ergs\ cm^{-2}A^{-1}electron^{-1}$
- PHOTPLAM: the bandpass pivot wavelength
- PHOTBW: the bandpass RMS width

Conversion to Signal Rate (UNITCORR)

This step converts the science data from a time-integrated signal to a signal rate by dividing the SCI and ERR arrays for reach readout by the TIME array. No reference file is needed. The BUNIT keyword in the output data header reflects the appropriate data units. The FLATCORR keyword is checked to decide on proper units for BUNIT and skip this step if "countrate" is found. If FLATCORR is set to "complete", then the units should be electrons, otherwise they are counts(the digitized signal from the FPA).

Fit accumulating signal and identify cosmic ray hits (CRCORR)

This step fits the accumulating signal up the image ramp and identifies cosmic-ray hits for each sample using the Fixsen et al (2000) (http://adsabs.harvard.edu/abs/2000PASP..112.1350F) methods.

The process is described below:

- **An iterative fit to the accumulating sample time is calculated for each pixel**
  - **Finding a cosmic ray ends one interval and beins the next; the cosmic ray must be included in the next interval**
    - **intervals are first defined based on existing cosmic rays**
      - CRSIGMAS from the CRREJTAB reference file is used to set the rejection threshold
    - then each interval is fitted separately
    - then each interval is inspected for SPIKES

        * then each interval is inspected for more cosmic rays
    - If any SPIKES or cosmic rays are found then the entire procedure repeats, new intervals are defined, etc ...
    - After the iteration ends because no new SPIKES or cosmic rays are found, each interval is fitted separately once again, with optimum weighting, and then the results for each interval are combined to obtain the final solution for the pixel.
    - The linearity fit includes readnoise in the sample weights and Poisson noise from the source in the final fit uncertainty
    - negative cosmic ray hits are detected and detected SPIKES have their 1024 bit flipped in the DQ extension
    - **If the first read is saturated the output pixels are never zeroed out**
        * The output pixel values in this case are the value in the input zeroth read image, regardless of whether the zero-read image was saturated. If it is saturated in the zero-read, the DQ flag will get carried over to the outpu DQ array to indicate that it's bad.
    - The DATAREJECT DQ flag is set for all samples following a hit. This is done so that anyone looking at the IMA file will know that the absolute value of the pixel is wrong after the firs t hit, but it smears the location of any hits which occurred in addition to the first one.
    - Pixels in the DQ image of the output IMA file are flagged with a value of 8192, the SCI and ERR image arrays are left unchanged
    - DQ values from any sample are carried through to the output pixel if a pixel has no good samples
- The UNSTABLE DQ flag is used to record pixels with higher than max allowed cosmic ray hits recorded.

The result of this step is stored as a single imset in the output FLT file. In the FLT file, the SCI array contains the final slope computed for each pixel, the ERR array contains the estimated uncertainty in the slope, the SAMP array contains the number of non-flagged samples used to compute the slope, and the TIME array contains the total exposure time of those smaples. Pixels for which no unflagged sample exisists (dead pixels for example) still get a slope computed which is recorded in the SCI array of the output FLT image, but the DQ flags in the FLT will reflect their bad status.

Flatfield Correction (FLATCORR)

This step corrects for sensativity variations across the detector by dividing the images by one or more reference flatfields (taken from the PFLTFILE, DFLTFILE or LFLTFILE header keywords). The mean gain from all the amps is used to convert to the image to units of electrons. Errors and DQ flags from the flatfields are combined with the science data errors and flag, the TIME and SAMP arrays are unchanged.

Calculation of image statistics

The min, mean, maxmin and max SNR (for the SCI and ERR) for data values flagged as "good" in the DQ array (i.e. zero) are calculated and stored in the output SCI image header, the reference pixels are not used. This is performed for all samples in the IMA file as well as the FLT image but the input data is not modified in any way. Updated keywords in the science header include:

- NGOODPIX
- GOODMEAN
- GOODMIN
- GOODMAX
- SNRMEAN

- SNRMIN
- SNRMAX

Reject cosmic rays from multiple images (RPTCORR)

Reject cosmic rays from multiple images. CR-SPLIT and REPEAT-OBS exposures get combined, other members of the association file must be combined with Astrodrizzle (see Astrodrizzle (http://www.stsci.edu/hst/HST_overview/drizzlepac/)), which will also correct for geometric distortion. This step is also referred to as *wf3rej*. The task uses the same statistical detection algorithm developed for ACS (acsrej), STIC (ocrrj) and WFPC2(crrej), providing a well-tested and robust procedure.

1.2 wf3cte

The charge transfer efficiency (CTE) of the UVIS detector has inevitably been declining over time as on-orbit radiation damage creates charge traps in the CCDs. Faint sources in particular can suffer large flux losses or even be lost entirely if observations are not planned and analyzed carefully. The CTE loss will depend on the morphology of the source, the distribution of electrons in the field of view (from sources, background, cosmic rays, and hot pixels) and the population of charge traps in the detector column between the source and the transfer register. And the magnitude of the CTE loss increases continuously with time as new charge traps form.

CTE is typically measured as a pixel-transfer efficiency, and would be unity for a perfect CCD. One indicator of CTE is the Extended Pixel Edge Response (EPER). Inefficient transfer of electrons in a flat-field exposure produces an exponential tail of charge in the overscan region. Analysis of monitoring observations through January 2013 shows that CTE continues to decline linearly over time (WFC3 ISR 2013-03 (http://www.stsci.edu/hst/wfc3/documents/ISRs/WFC3-2013-03.pdf)).

The CTE correction step uses it's own dark reference files which have themselves been corrected for CTE. The specific reference file used for any dataset may be found in the image header keyword DRKCFILE. This step also uses a special bias reference file as part of the CTE correction itself, referred to in the header by the BIACFILE keyword. This BIACFILE is only used to facilitate the CTE correction, the resulting corrected image then uses the normal BIASFILE to correct the science frame after the CTE correction has been performed. After the CTE correction has been performed and the data progresses through the rest of the pipeline, the special CTE corrected dark, DRKCFILE in the header, will be used for the dark current correction instead of the DARKFILE.

There is a PCTETAB refrence file which contains extensions of calibration images and tables of parameters used during the CTE correction stage. The header of this file also contains parameters for the CTE correction algorithm. These parameters, and important scalars which are used to correct the data are stored in the output image headers. See the UVIS2.0 reference ISR for more information.

This routine performs the CTE correction on raw data files. The calibration step keyword is PCTECORR, if this is set to PERFORM then the CTE correction will be applied to the dataset. Some caveats for its use:

- CTE corrections can *ONLY* be performed on RAW data which has not been calibrated in any way.
- Data which have already been through BLEVCORR, BIASCORR or DARKCORR will be rejected.
- The CTE correction step in the pipeline is currently only implemented for FULL FRAME images

The standalone call will produce a RAC fits file by default. This contains only the CTE corrected data, no other calibrations have been performed.

For more information the the WFC3 CTE please see the WFC3 CTE webpage (http://www.stsci.edu/hst/wfc3/ins_performance/CTE/) .

1.2.1 Running `wf3cte` from a python terminal

In Python without TEAL:

```
>>> from wfc3tools import wf3cte
>>> wf3cte(filename)
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from wfc3tools import wf3cte
>>> teal.teal('wf3cte')
```

In Pyraf:

```
>>> import wfc3tools
>>> epar wf3cte
```

1.2.2 Displaying output from wf3ccd in a Jupyter Notebook

When calling `wf3cte` from a Jupyter notebook, informational text output from the underlying `wf3cte.e` program will be passed through `print` as the calibration runs by default, and show up in the user's cell. This behavior can be customized by passing your own function as the `log_func` keyword argument to `wf3cte`. As output is read from the underlying program, the `wf3cte` Python wrapper will call `log_func` with the contents of each line. (`print` is an obvious choice for a log function, but this also provides a way to connect `wf3cte` to the Python logging system by passing the `logging.debug` function or similar.)

If `log_func=None` is passed, informational text output from the underlying program will be ignored, but the program's exit code will still be checked for successful completion.

1.2.3 Parameters

input [str]

Name of input files

- a single filename (`iaa012wdq_raw.fits`)
- a Python list of filenames
- a partial filename with wildcards (`\*raw.fits`)
- filename of an ASN table (`\*asn.fits`)
- an at-file (`@input`)

-1 : value, as in minus one, this will make sure only 1 processor/thread is used during processing, otherwise all available are used.

verbose: bool, optional Print verbose time stamps?

The wf3cte function can also be called directly from the OS command line:

```
>>> wf3cte.e input  [-options]
```

Where the OS options include:

- -v: verbose
- -1: turn off multiprocessing

Basic Steps In The CTE Correction

- The reference bias image named in the BIACFILE header keyword is subtracted from the data
- Parameters from the CTE parameter table, referenced in the PCTETAB header keyword, are read and stored
- The date is reformatted so that each quadrant has been rotated such that the readout amp is located at the lower left of the array. The reoriented four quadrants are then arranged into a single 8412x2070 image (including the overscan) with amps CDAB in that order. In this format, the pixels are all parallel-shifted down, then serial-shifted to the left
- An additional bias correction is performed using the residual bias level measured for each amplifier from the steadiest pixels in the horizontal overscan, this value is then subtracted from all the pixels in each respective amp
- The image is corrected for gain
- The smoothest image that is consistent with being the observed image plus readnoise is found and subtracted. This is necessary because we want the CTE correction algorithm to produce the smoothest possible reconstruction, consistent with the original image and the known readnoise. The algorithm then constructs a model that is smoother where there pixel-to-pixel variations aren't too large, then it respects the pixel values, using a 2sigma threshold to mitigate readnoise amplification, iteration is not done when the deblurring is less than the readnoise.
- The CTE correction itself is calculated and then subtracted from the original, raw, uncorrected and uncalibrated image.
- The corrected image is now ready to continue through the rest of the pipeline. When the DARKCORR header keyword is set to perform, the CTE corrected image will use the dark reference file referred to in the DRKCFILE header keyword.

The PCTETAB and Algorithm Parameters

The following are new primary header keywords which will be updated in the data headers during the `wf3cte` step. They are also specified in the PCTETAB reference file.

KEYWORD	DESCRIPTION
CTE_NAME	name of cte algorithm [string]
CTE_VER	version number of cte algorithm [string]
CTEDATE0	date of wfc3/uvis installation in HST, in modified Julian days (MJD)
CTEDATE1	reference date of CTE model pinning, in modified Julian days (MJD)
PCTETLEN	max length of CTE trail
PCTERNOI	readnoise amplitude for clipping
PCTENFOR	number of iterations used in CTE forward modeling
PCTENPAR	number of iterations used in the parallel transfer
PCTENSMD	readnoise mitigation algorithm
PCTETRSH	over-subtraction threshold
PCTEFRAC	cte scaling frac calculated from expstart and used in the algorithm
PCTERNOI	the readnoise clipping level to use
FIXROCR	make allowance for readout cosmic rays

The PCTETAB reference file has 4 extensions, two tables and two images:

```
Filename: wfc3_cte.fits
No.    Name         Type      Cards   Dimensions   Format
0    PRIMARY     PrimaryHDU      21   ()
1    QPROF       BinTableHDU     16   999R x 3C    ['i', 'i', 'i']
2    SCLBYCOL    BinTableHDU     20   8412R x 5C   ['i', 'e', 'e', 'e', 'e']
```

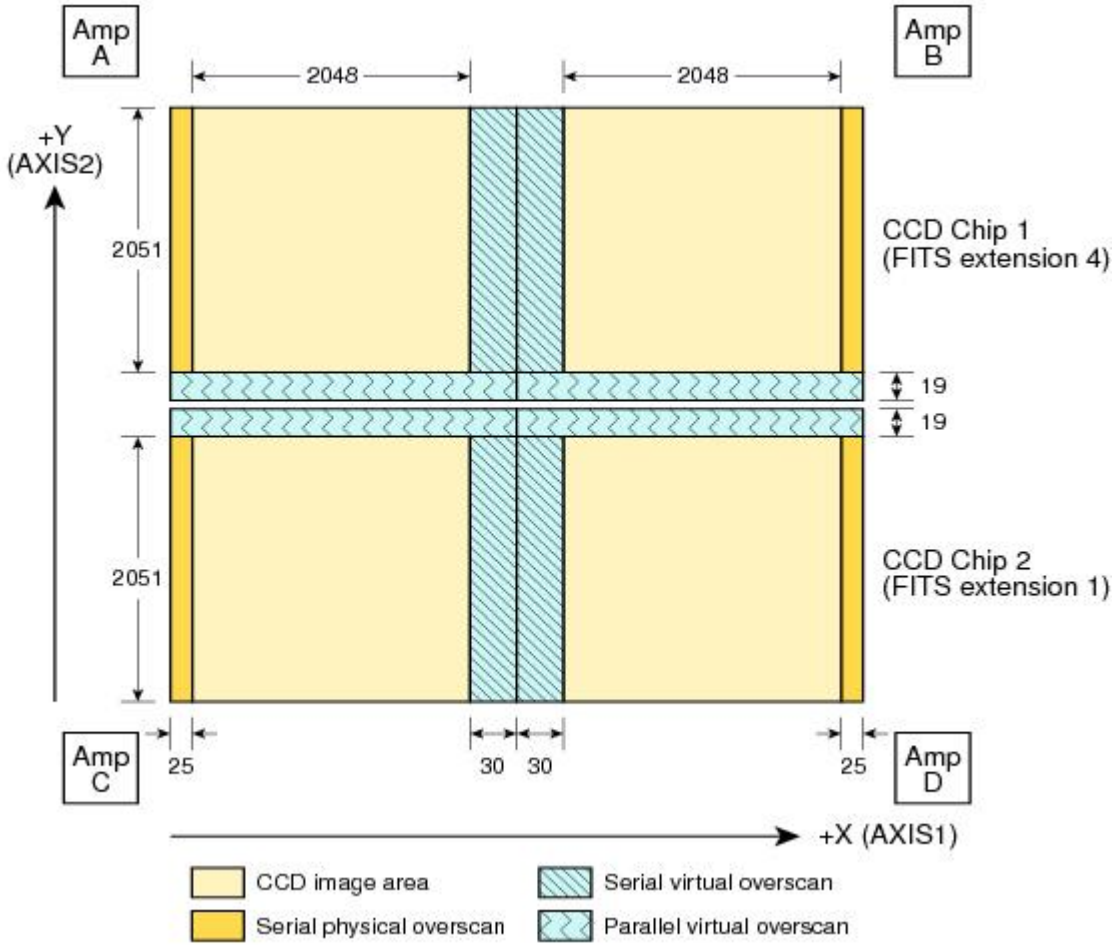


Fig. 1.5: UVIS data raw full-frame file format

```
3    RPROF       ImageHDU        12   (999, 100)   float32
4    CPROF       ImageHDU        12   (999, 100)   float32
```

The first extension lists the charge-trap levels, the columns are respectively the trap number, the charge-packet size it applies to (in electrons), and the size of the trap (also in electrons).

The second extension contains the CTE scalings as a function of column number. There are 5 columns, each with 8412 elements. The first column contains the integer column number in the amp readout-aligned large array. The other columns contain the CTE scaling appropriate for that column at the 512th, 1024th, 1536th and 2048th rows respectively.

The third extension contains the differential CTE trail profile as a function of charge level in the form of an image

The fourth extension contains the cumulative CTE trail profile as a function of charge level, also in the form of an image.

Output Files

If you are running the separate `wf3cte.e` step a _rac.fits file will be output. This the same as a _raw.fits file except the CTE correction has been applied to the data.

If the PCTECORR step is set to PEFORM:

- when the _raw.fits file enters calwf3, then no intermediate _rac.fits file will be saved, unless you specify the `-s` flag, which instructs `calwf3.e` to save all intermediate files.
- the `calwf3` pipeline will produce both CTE calibrated product and non-CTE calibrated products. The CTE products have a 'c' at the end of their extension name, such as _blc, _rac, _crc, _flc, and the non-CTE calibrated products contain the familiar : _blv, _crj, _flt.

1.3 wf3ccd

This routine contains the initial processing steps for all the WFC3 UVIS channel data. These steps are:

- DQICORR - initializing the data quality array
- ATODCORR - perform the a to d conversion correction
- BLEVCORR - subtract the bias level from the overscan region
- BIASCORR - subtract the bias image
- FLSHCORR - subtract the post-flash image

`wf3ccd` first subtracts the bias and trims the overscan regions from the image. If an associated set of UVIS CR-SPLIT or REPEAT-OBS images is being processed, all of the overscan-trimmed images are sent through `wf3rej` to be combined and receive cosmic-ray rejection. The resulting combined image then receives final calibration with `wf32d`, which includes dark subtraction and flat-fielding. If there are multiple sets of CR-SPLIT or REPEAT-OBS images in an association, each set goes through the cycle of `wf3ccd`, `wf3rej` and `wf32d` processing.

If BLEVCORR is performed the output contains the overcan-trimmed region.

Only those steps with a switch value of PERFORM in the input files will be executed, after which the switch will be set to COMPLETE in the corresponding output files.

1.3.1 Running `wf3ccd` from a python terminal

In Python without TEAL:

```
>>> from wfc3tools import wf3ccd
>>> wf3ccd(filename)
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from wfc3tools import wf3ccd
>>> teal.teal('wf3ccd')
```

In Pyraf:

```
>>> import wfc3tools
>>> epar wf3ccd
```

1.3.2 Displaying output from wf3ccd in a Jupyter Notebook

When calling `wf3ccd` from a Jupyter notebook, informational text output from the underlying `wf3ccd.e` program will be passed through `print` as the calibration runs by default, and show up in the user's cell. This behavior can be customized by passing your own function as the `log_func` keyword argument to `wf3ccd`. As output is read from the underlying program, the `wf3ccd` Python wrapper will call `log_func` with the contents of each line. (`print` is an obvious choice for a log function, but this also provides a way to connect `wf3ccd` to the Python logging system by passing the `logging.debug` function or similar.)

If `log_func=None` is passed, informational text output from the underlying program will be ignored, but the program's exit code will still be checked for successful completion.

1.3.3 Parameters

input [str] Name of input files

- a single filename (`iaa012wdq_raw.fits`)
- a Python list of filenames
- a partial filename with wildcards (`\*raw.fits`)
- filename of an ASN table (`\*asn.fits`)
- an at-file (`@input`)

output: str Name of the output FITS file.

dqicorr: str, "PERFORM/OMIT", optional Update the dq array from bad pixel table

atodcorr: str, "PERFORM/OMIT", optional Analog to digital correction

blevcorr: str, "PERFORM/OMIT", optional Subtract bias from overscan regions

biascorr: str, "PERFORM/OMIT", optional Subtract bias image

flashcorr: str, "PERFORM/OMIT", optional Subtract post-flash image

verbose: bool, optional Print verbose time stamps?

quiet: bool, optional Print messages only to trailer file?

The wf3ccd function can also be called directly from the OS command line:

```
>>> wf32ccd.e input output [-options]
```

Where the OS options include:

- -v: verbose
- -f: print time stamps
- -dqi: udpate the DQ array
- -atod: perform gain correction
- -blev: subtract bias from overscan
- -bias: perform bias correction
- -flash: remove post-flash image

1.4 wf32d

Use this function to facilitate batch runs or for the TEAL interface.

The wf32d primary functions include:

- DARKCORR: dark current subtraction
- FLATCORR: flat-fielding
- PHOTCORR: photometric keyword calculations
- FLUXCORR: photometric normalization of the UVIS1 and UVIS2 chips

Only those steps with a switch value of PERFORM in the input files will be executed, after which the switch will be set to COMPLETE in the corresponding output files.

1.4.1 Examples

In Python without TEAL:

```
>>> from wfc3tools import wf32d
>>> wf32d(filename)
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from wfc3tools import wf32d
>>> teal.teal('wf32d')
```

In Pyraf:

```
>>> import wfc3tools
>>> epar wf32d
```

1.4.2 Parameters

input [str] Name of input files

- a single filename (`iaa012wdq_raw.fits`)
- a Python list of filenames

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

output: str Name of the output FITS file.

dqicorr: str, "PERFORM/OMIT", optional Update the dq array from bad pixel table

darkcorr: str, "PERFORM/OMIT", optional Subtract the dark image

flatcorr: str, "PERFORM/OMIT", optional Multiply by the flatfield image

shadcorr: str, "PERFORM/OMIT", optional Correct for shutter shading (CCD)

photcorr: str, "PERFORM/OMIT", optional Update photometry keywords in the header

fluxcorr; str, "PERFORM/OMIT", optional Perform chip photometry normalization

verbose: bool, optional Print verbose time stamps?

quiet: bool, optional Print messages only to trailer file?

The wf32d function can also be called directly from the OS command line:

```
>>> wf32d.e input output [-options]
```

Where the OS options include:

- -v: verbose
- -f: print time stamps
- -d: debug
- -dark: perform dark subtraction
- -dqi: update the DQ array
- -flat: perform flat correction
- -shad: perform shading correction
- -phot: perform phot correction

1.5 wf3ir

Use this function to facilitate batch runs or for the TEAL interface.

This routine contains all the instrumental calibration steps for WFC3 IR channel images. The steps are:

- DQICORR - initialize the data quality array
- ZSIGCORR - estimate the amount of signal in the zeroth-read
- BLEVCORR - subtact the bias level from the reference pixels
- ZOFFCORR - subtract the zeroth read image
- NLINCORR - correct for detector non-linear response
- DARKCORR - subtract the dark current image
- PHOTCORR - compute the photometric keyword values
- UNITCORR - convert to units of count rate

- CRCORR - fit accumulating signal and identify the cr hits
- FLATCORR - divide by the flatfield images and apply gain coversion

The output images include the calibrated image ramp (ima file) and the accumulated ramp image (flt file)

Only those steps with a switch value of PERFORM in the input files will be executed, after which the switch will be set to COMPLETE in the corresponding output files.

1.5.1 Running `wf3ir` from a python terminal

In Python without TEAL:

```
>>> from wfc3tools import wf3ir
>>> wf3ir(filename)
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from wfc3tools import wf3ir
>>> teal.teal('wf3ir')
```

In Pyraf:

```
>>> import wfc3tools
>>> epar wf3ir
```

1.5.2 Displaying output from wf3ir in a Jupyter Notebook

When calling `wf3ir` from a Jupyter notebook, informational text output from the underlying `wf3ir.e` program will be passed through `print` as the calibration runs by default, and show up in the user's cell. This behavior can be customized by passing your own function as the `log_func` keyword argument to `wf3ir`. As output is read from the underlying program, the `wf3ir` Python wrapper will call `log_func` with the contents of each line. (`print` is an obvious choice for a log function, but this also provides a way to connect `wf3ir` to the Python logging system by passing the `logging.debug` function or similar.)

If `log_func=None` is passed, informational text output from the underlying program will be ignored, but the program's exit code will still be checked for successful completion.

1.5.3 Parameters

input [str] Name of input files

- a single filename (`iaa012wdq_raw.fits`)
- a Python list of filenames
- a partial filename with wildcards (`\*raw.fits`)
- filename of an ASN table (`\*asn.fits`)
- an at-file (`@input`)

output: str Name of the output FITS file.

verbose: bool, optional Print verbose time stamps?

quiet: bool, optional Print messages only to trailer file?

The wf3ir function can also be called directly from the OS command line:

```
>>> wf32ir.e input output [-options]
```

Where the OS options include:

- -v: verbose
- -f: print time stamps

1.6 wf3rej

This calls the wf3rej executable. Use this function to facilitate batch runs or for the TEAL interface.

1.6.1 Running `wf3rej` from a python terminal

In Python without TEAL:

```
>>> from wfc3tools import wf3rej
>>> wf3rej(filename)
```

In Python with TEAL:

```
>>> from stsci.tools import teal
>>> from wfc3tools import wf3rej
>>> teal.teal('wf3rej')
```

In Pyraf:

```
>>> import wfc3tools
>>> epar wf3rej
```

1.6.2 Displaying output from wf3rej in a Jupyter Notebook

When calling `wf3rej` from a Jupyter notebook, informational text output from the underlying `wf3rej.e` program will be passed through `print` as the calibration runs by default, and show up in the user's cell. This behavior can be customized by passing your own function as the `log_func` keyword argument to `wf3rej`. As output is read from the underlying program, the `wf3rej` Python wrapper will call `log_func` with the contents of each line. (`print` is an obvious choice for a log function, but this also provides a way to connect `wf3rej` to the Python logging system by passing the `logging.debug` function or similar.)

If `log_func=None` is passed, informational text output from the underlying program will be ignored, but the program's exit code will still be checked for successful completion.

1.6.3 Parameters

input : str, Name of input files

- a single filename (`iaa012wdq_raw.fits`)
- a Python list of filenames
- a partial filename with wildcards (`\*raw.fits`)
- filename of an ASN table (`\*asn.fits`)

- an at-file (`@input`)

output : str, Name of the output FITS file.

crrejtab : string, reference file name

scalense : string, scale factor applied to noise

initgues : string, intial value estimate scheme (min|med)

skysub : string, how to compute the sky (none|mode|mean)

crsigmas : string, rejection levels in each iteration

crradius : float, cosmic ray expansion radius in pixels

crthresh : float, rejection propagation threshold

badinpdq : int, data quality flag bits to reject

crmask : bool, flag CR in input DQ imageS?

shadcorr : bool, perform shading shutter correction?

verbose : bool, optional, Print verbose time stamps?

The wf3rej executable can also be called directly from the OS command line prompt:

```
>>> wf3rej.e input output [-options]
```

Input can be a single file, or a comma-delimited list of files.

Where the OS options include:

- t: print the timestamps
- v: verbose
- shadcorr: perform shading shutter correction?
- crmask: flag CR in input DQ images?
- table <filename>: the crrejtab filename
- scale <number>: scale factor for noise
- init <med|min>: initial value estimate scheme
- sky <none|median|mode>: how to compute sky
- sigmas: rejection leves for each iteration
- radius <number>: CR expansion radius
- thresh <number> : rejection propagation threshold
- pdq <number>: data quality flag bits to reject

1.7 Software Update History for HSTCAL.CALWF3

Warning: IRAF version of WFC3 no longer maintained or delivered, use WFC3TOOLS in HSTCAL or call the executable from your operating system command line. With version 3.3 the pipeline now produces two versions of each calibrated file, one set with the CTE correction applied and one set without the CTE correction applied

Updates for Version 3.3 28-Jan-2016 MLS

- Removed the call to WF3Dth for the CTE data. This is essentially useless since the DTH step has been replaced with astrodrizzle, the only function it has it to concatinate the SPT files from the association members into a new spt file with the product/subproduct name. This is already done for the non-cte processed data, there is no difference between the cte and non-cte SPT files, they are only related to the RAW input file.

Updates for Version 3.3 22-Jan-2016 MLS

- Noticed I missed renaming the crc file with the call to WF3DTH and was setting the input to the crj suffix, but also needed updates to GetAsnTable to include crc filename population for subproducts and sending the correct subproduct through processing, it was always using crj
- some code cleanup as I went
- trailer file concatination update, but it's still not entirely correct

Updates for Version 3.3 19-Jan-2016 MLS

- Updates to fix trailer file output information. Part of the problem is the change in name of the output RAC_TMP file, the functions which create input and output trailer file naming conventions look for the dot to separate extensions so I need to change some expectations.
- Update to check for the raw header value of PCTENSMD and update the cte parameter table accordingly (though only a value of 0 is valid right now)

Updates for Version 3.3 08-Jan-2016 MLS

- changed the name of the output rac file to rac_tmp so that archive could handle deleting it better

Updates for Version 3.3 05-Nov-2015 MLS

- removed explicit downcasts and added calloc returns in procccd

Updates for Version 3.3 04-Nov-2015 MLS

- add more explicity initializations where I could find them

Updates for Version 3.3 24-Oct-2015 MLS

- The nightly build is specifying a higher level of optimization through to the compiler than the debug mode that I have been using for my testing. Building calwf3 with the lower optimization produced no errors or warnings and ran through cleanly, but the high optimization brought on a segfault inside the CTE code in the first openmp section. This was only happening on the Linux cluster, the MAC builds showed no issue. The problem area seemed to be a set of arrays which would rather be doubles than floats. I also changed the remaining floats to doubles, where I could, and removed more of the memcpy statements, making them regular array assignments.
- I also removed a superfluous openmp print statement from maincte.c and cleaned up some more informational print statements.
- I added a time measurement, with verbose on the code prints how long the CTE section took to run, with the specification of number of threads/cpus.

Updates for Version 3.3 21-Oct-2015 MLS

- Editing text for the screen and trailer files
- Formally removed the rac file before the routine ends since archive isn't expecting it

Updates for Version 3.3 16-Oct-2015 MLS

- machine dependent bug, some images were getting nan values on linux machines
- I also removed the temporary image saves we were using for the CTE routines

Updates for Version 3.3 29-Sep-2015 MLS

- bug in original fortran code fixed; the final RAC image should be made by subtracting the CHG image (the net cte effect) from the original non-BIC subtracted raw data. This should remove the additional bias signature that Matthew was seeing in the stacked dark frames. It should NOT make a significant change in the overall output of the code since bias levels are low to begin with.
- I also changed the way the code uses the SCLBYCOL reference file (as called in Jays fortran). The way the fortran code is structured, the reference file information never actually gets used in the calculation. This doesn't make a numerical difference at the moment because the reference file values are all ones, ie. there is no additional scaling done on the CTE pixel other than by using the CTE scaling fraction and the column location. However, if the science team ever delivers a new reference file which has these values updated, they wont actually get used by the code unless this change is implemented.
- Reformatted some code for readability, and fixed SEGFAULT error in reference file checking when iref environment variable not set by user, so can't find file (also when can't find file in general). I made RefExist exit clean the first time it found a missing file, HSTIO was barfing any other way.

Updates for Version 3.3 24-Sep-2015 MLS

- fix for machine dependent precision bug

Updates for Version 3.3 03-Sep-2015 MLS

- One more precision change needed for the nans in the readnoise section

Updates for Version 3.3 28-Aug-2015 MLS

- These updates appear to fix the nan issue in the readnoise step that we ran into with some images
- I also made the cte code a bit more tidy and organized

Updates for Version 3.3 25-Aug-2015 MLS

- changed pow() to powf() in the readnoise calculation to deal with memory overrun producing nans in some cases

Updates for Version 3.3 24-Aug-2015 MLS

- updated the mac os version check in wscript to use sw_vers, the old way was returning junk and we need it for adding the 64bit flags to the compile
- added some initializations the clang compiler complained about

Updates for Version 3.3 20-Aug-2015 MLS

- I changed a float to double in wf3cte readnoise section for added precision
- moved GetGlobalInfo and checkGlobal info higher in the code to reject non-wfc3 datasets
- moved a delete section further out in the logic and that seemed to fix #1220, tests on cte and non-cte data seemed happy

Updates for Version 3.3 18-Aug-2015 MLS

- BuildDthInput has to create the input filename from the asn root, but this can be either FLT or FLC now, have to figure out which one to use.
- Had to add separate DTH pass for IR data and double DTH pass for UVIS data because the input filename for RPTCORR/EXPCORR associations are built in the code from the data rootnames in the ASN table. So the UVIS data coming out of procccd has to take a double pass through DTH when PCTECORR is PERFORM.
- changed the checking order for subarrays in the PCTECORR routine so that it errors out cleanly (has to do with 1 group of images for subarrays)

- added the check for INSTRUMENT == WFC3 back to the code, actually related to a user complainging that calwf3 didn't tell them it couldn't reduce ACS data.
- had to update the procir call to wf3rej_0 signature for the asn update I added to uvis
- updated the mainrej.e calls which were segfaulting (calling wf3rej standalone on input list of images)
- added dynamic memory allocation for trailer file list to initrejtrl
- updated text in wf3rej to report that Astrodrizzle should be used to align images instead of PyDrizzle since that's how it's advertised to users
- found a problem (even in the released version of calwf3) with output file for associations with multiple products, created #1220

Updates for Version 3.3 12-Aug-2015 MLS

- fix for #1215 binned data detection for sink pixel seg faults

Updates for Version 3.3 11-Aug-2015 MLS

- nrej initialized in wf3rej so that REJ_RATE reported consistently correct, see #1214
- fix for #1216, the BIACFILE name was not being populated for bias images with BIASCORR == OMIT
- I also went ahead and added a clean exit for images going to PCTECORR which already have BIASCORR complete

Updates for Version 3.3 21-July-2015 MLS

- Debugged version of the CTE code committed.
- see #1193 ticket for extensive changes

Updates for Version 3.3 31-May-2015 MLS

- UVIS 2.0 added, including CTE correction, Sink Pixel and Photometry updates
- (#1011) New photometry correction for UVIS. This includes a delivery of new flatfields for all filters in CDBS as well as a new IMPHTTAB. The new calibration step is controlled by the FLUXCORR keyword in the image header.
- (#1154) CTE correction for all UVIS data. This is done in conjunction with a full run through of the pipeline code without the CTE correction applied. This correction is for the same reasons as in ACS, but the CTE correction method and code are different, and they are applied to the raw file instead of later in the processing. Some sections of the CTE code support parallel processing with OpenMP. The default for calwf3 is to use all available processors. To restrict processing to 1 cpu use the flag -1 in the call to calwf3.e The cte processing is controlled with the PCTECORR keyword.
- Sink pixels added to the science image DQ mask using the SNKCFILE reference image. This image has 2 extensions, each in the pre-overscan trimmed format. This step is performed if DQICORR is PERFORM, and is done before BLEVCORR while the science image is still untrimmed.
- see #1193 for more detailed information on all the updates

Updates for Version 3.2.1 08-Dec-2014 MLS:

- The FLUXCORR step has been updated, changing how the data is processed in the flow of the pipeline. It was discovered that a chain of requirements meant that the values from the IMPHTTAB were not being read or updated correctly. This is a multifold problem which starts with the way that the IMPHTTAB is read and how it is constructed. Since the file, and it's calling functions, are common to all instruments, the best way around it was to move where the fluxcorr step was done in the pipeline to OUTSIDE the main wf32d loop. The step then reads in the FLT file which was written out and updates the SCI,1 data and headers with the photometry keyword information.

Updates for Version 3.2 09-Dec-2013 MLS:

- A new calibration step was added to the UVIS process, FLUXCORR, can now be run at the end of regular processing. It will scale the chip2 image using the new PHTFLAM1 and PHTFLAM2 values in the IMPHTAB. New flatfields for all filters, as well as a new IMPHTTAB will be delivered by the team for this step to be completely implemented. This is a significant version increase since I had to modify the globablly access GetPhotTab to read the new WFC3 imphttab correctly, as well as touch many routines in the calwf3 process.(see tickets #1088, #1011, #1025)

Updates for Version 3.1.6 15-Nov-2013 MLS:

- Fixed a file i/o issue after change in cfitsio interaction (see #970, #1073 and #1069)

Updates for Version 3.1.5 30-Sep-2013 MLS:

- Fixed the individual task executables for wf3ir, wf3ccd, wf32d to properly used the user specfied output filename when they are called standalone

Updates for Version 3.1.4 09-Sep-2013 MLS:

- Added a couple new functions to deal with user specified subarrays when they start in amp A or C and continue to B or D. In these cases the virtual overscan from the reference postflash file must be avoided, and just incrementing the starting pixel for the array in not a good solution.

Updates for Version 3.1.3 26-Mar-2013 MLS:

- Updated the postflash routine to apply the correct offset for all amps when a user specified subarray is used (no GO users are allowed to do this)
- Some unrelated files will change because I formatted the indentation to make the code easier to decipher

Updates for Version 3.1.2 11-Feb-2013 - MLS:

- Updated the bias subtraction to check for CCDAMP values of SINGLE_AMP and SINGLE_OR_ALL in the reference bias file image when a full frame reference file and a user specified subarray are used so that the correct overscan region is ignored
- Removed check for TDFTRANS per team request, see #980, I'm keeping the same version as the previous change because I havent delivered it yet

Updates for Version 3.1.1 2-Jan-2013 - MLS:

- File I/O in acsrej updated to avoid problems with CFITSIO upcasting file permissions to read/write when not needed. This will allow the hstio code to remove logic that allowed the old code to work but caused problems for CADC when CFITSIO opened ref files in read/write mode because of that hstio logic.

Updates for version 3.1 31-Dec-2012 MLS:

- fixed TrlBufInit problem so it initializes correctly (r21162)

Updates for version 3.1 28-Dec-2012 MLS:

- Updated to account for a memory leak on linux machines during BuildDth when RPTCORR is off and a new spt is being constructed (#967)

Warning: HST CAL DELIVERED, STSDAS+IRAF version no longer maintained, use WFC3TOOLS in HST-CAL

Updates 18Oct 2012 - MLS - Version 2.7.1

- fixed a memory leak in cridcalc that was occuring on linux machines and only affected IR data.
- version date and number updated

Updates for version 2.7 21-May-2012 MLS:

- **cridcal.c/wf3dq.h:**
  - update to help negative cr detections (fabs the comparison)
  - updated the spike flag to 1024 so that those pixels weren't ignored in the rejection routinea
  - Use zero read pixel value for WF3 IR ramp fitting when saturated
- **do2d.c, cr_scaling.c:**
  - update for BUNIT keyword value so it's not case sensitive, BUNIT value now stored as ELECTRONS instead of electrons as well
- wf32d: version update to 07may2012
- wf3rej.cl: version update to 07may2012
- wf3version.h: version update to 07may2012
- wf3main.c: new option r added to print current version and exit

Updates for version 2.6.3 23-Mar-2012 (HAB):

- calwf3.cl: Increment version to 23Mar2012.
- wf3version.h: Increment version to 2.6.3 and date to 23-Mar-2012.
- calwf3/calwf3.c: Upgraded the BuildDthInput function to build file list from names of individual association members when a CRJ sub-product has not been created. (PR 70922; Trac #869)
- calwf3/procir.c: Updated to set CRJ sub-product status to PRESENT after running wf3rej, and report RPTCORR switch status via trlmessage when wf3rej is not run. (PR 70922; Trac #869)

Updates for version 2.6.2 27-Jan-2012 MLS:

- calwf3.cl: Increment version to 27Jan2012.
- wf3version.h: Increment version to 2.6.2 and date to 27-Jan-2012.
- wf3rej/rej.h: Decreased MAX_FILES from 250 to 120 because OPUS is still getting errors when trying to process this many images.

Updates for version 2.6.1 24-Jan-2012 MLS:

- calwf3.cl: Increment version to 24Jan2012.
- wf3version.h: Increment version to 2.6.1 and date to 24-Jan-2012.
- calwf3/procir.c: Added a check for the number of images present when RPTCORR=PERFORM so that wf3rej is not run for singletons.

Updates for version 2.6 - 15-Dec-2011 (HAB):

- calwf3.cl: Increment version to 15Dec2011.
- wf3version.h: Increment version to 2.6 and date to 15-Dec-2011.
- wf3rej/cr_scaling.c: Upgraded to read BUNIT keyword value from first SCI extension header of each input image. (PR 69969; Trac #814)
- wf3rej/rej_do.c: Upgraded to pass new bunit array to and from all functions that need it, in order to handle input data that are in count rates. (PR 69969; Trac #814)
- wf3rej/rej_init.c: Upgraded to rescale input data that are in units of count rates.(PR 69969; Trac #814)
- wf3rej/rej_loop.c: Upgraded to rescale input data that are in units of count rates. (PR 69969; Trac #814)
- wf3rej/rej_sky.c: Upgraded to rescale input data that are in units of count rates. (PR 69969; Trac #814)

Updates for version 2.5.1 - 09-Dec-2011 (HAB):

- calwf3.cl: Increment version to 09Dec2011.
- wf3version.h: Increment version to 2.5.1 and date to 09-Dec-2011.
- calwf3/procir.c: Modified the logic that controls the rptcorr processing so that it's based on the setting of the RPTCORR header keyword switch, instead of just always applying it to every repeat-obs association. (PR 69952; Trac #807)

Updates for version 2.5 - 01-Oct-2011 (HAB):

- calwf3.cl: Increment version to 01Oct2011.
- wf3version.h: Increment version to 2.5 and date to 01-Oct-2011.
- wf3ir/cridcalc.c: Fixed fitsamps routine to correctly accumulate int_time in odd cases where the 1st or 2nd read is bad. (PR 69230; Trac #770)
- wf3ir/rej.h: Increase MAX_FILES from 120 to 250. (PR 63555)
- wf3rej/rej.h: Increased MAX_FILES from 120 to 250. (PR 63555)

Updates for version 2.4.1 - 02-Aug-2011 (HAB):

- calwf3.cl: Increment version to 02Aug2011.
- wf3version.h: Increment version to 2.4.1 and date to 02-Aug-2011.
- lib/wf3info.c: Fixed the logic in the CheckGain routine so that the ref image gets closed before returning when keyval=-1. (PR 68983; Trac #745)
- wf3ir/cridcalc.c: Updated crrej to free memory for tot_ADUs before returning. (PR 68993; Trac #748)

Updates for version 2.4 - 17-Jun-2011 (HAB):

- calwf3.cl: Increment version to 17Jun2011.
- wf3version.h: Increment version to 2.4 and date to 17-Jun-2011.
- calwf3/procccd.c: Modified logic involved in handling error returns from WF3Rej so that WF32d processing still takes place for individual exposures if EXPSCORR=PERFORM. (PR 68593; Trac #722)
- wf3rej/rej_init.c: Added missing call to free(ipts) at end.
- wf3rej/wf3rej.c: Fixed error status return from rej_do so that original status gets passed up for use in caller. (PR 68593; Trac #722)

Updates for version 2.3 - 15-Mar-2011 (HAB):

- calwf3.cl: Increment version to 15Mar2011.
- wf3version.h: Increment version to 2.3 and date to 15-Mar-2011.
- calwf3/calwf3.c: Modified CopyFFile routine to update the FILENAME keyword in created output file. (PR 67225; Trac #646)
- wf3ir/doir.c: No longer load dark ref file for zsigcorr. (PR 67728; Trac #681)
- wf3ir/getirflags.c: Removed zsigcorr checks in checkDark routine, because zsigcorr no longer uses the dark. (PR 67728; Trac #681)
- wf3ir/zsigcorr.c: Modified zsigcorr routine to just subtract the super-zero read image from the science zero read image to estimate zero read signal, rather than scaling the difference between the first and zero reads in the science image. This avoids problems with zoer read exposure time in subarray exposures. Also eliminated use of dark image. (PR 67728; Trac #681)

Updates for Version 2.2 - 01-Dec-2010 (HAB):

- calwf3.cl: Increment version to 01Dec2010.
- wf3version.h: Increment version to 2.2 and date to 01-Dec-2010.
- calwf3/calwf3.c: Modified CalWf3Run and BuildDthInput to skip processing for sub-products that have < 2 members present, because no sub-product is produced in this case. (PR 66366; Trac #622)
- calwf3/getreffiles.c: Modified GetIRRef to correctly check all IR switches, so that re-entrant processing works correctly. (PR 66081; Trac #608)
- calwf3/wf3dth.c: Modified InitDthTrl to return with no action if the input member list is empty, to handle missing asn members. (PR 66366; Trac #622)
- calwf3/wf3table.c: Modified GetAsnTable to turn off CRCORR/RPTCORR if there aren't any sub-products with > 1 member. (PR 66366; Trac #622)
- lib/tabpedigree.c: When tbtopn has a failure, reset status to zero before returning, so that calling routines have a chance to print error messages before shutting down. (PR 65410; Trac #578)
- lib/trlbuf.c: Modified WriteTrlFile to check for non-null pointer before trying to close trailer file (PR 66366; Trac #622).
- wf3ir/cridcalc.c: Changed crrej to always call EstimateDarkandGlow, regardless of darkcorr setting, because for WFC3 we use a static dark value and therefore don't need access to the darkfile. (PR 66081; Trac #608)
- wf3ir/doir.c: Upgraded crimage header updates to include check of flatcorr status when updating BUNIT values. Also modified noisMsg routine to print noiscorr switch value and have trailer message printed from noiscorr routine itself. Both changes are to support re-entrant processing. (PR 66081; Trac #608)
- wf3ir/getirflags.c: Fixed a reference to dqicorr in checkCRRej that should've been crcorr.
- wf3ir/getirsw.c: Modified GetSw routine to not reset cal switches to OMIT if they have a value other than PERFORM, in order to support re-entrant processing where some switches are COMPLETE. (PR 66081; Trac #608)
- wf3ir/groupinfo.c: Upgraded getDataUnits routine to recognize BUNIT values of ELECTRONS, to support re-entrant processing. (PR 66081; Trac #608)
- wf3ir/irhist.c: Upgraded noisIRHistory routine to first check setting of noiscorr switch before adding history keyword, to support re-entrant processing. (PR 66081; Trac #608)
- wf3ir/noiscalc.c: Modified doNoisIR to print trailer message and noiscorr value, and also give a message saying that noiscorr is skipped if noiscalc returns with an error. Noiscalc was modified to see if the ERR array is already populated before doing the calculation, to support re-entrant processing. (Pr 66081; Trac #608)
- wf3ir/pixcheck.c: Updated the WFC3 IR DQ value assignments. (PR 66080; Trac #607)
- wf3ir/unitcorr.c: Upgraded unitcorr routine to check flatcorr status to decide proper units for BUNIT keyword value update, to support re-entrant processing. (PR 66081; Trac #608)
- wf3ir/zsigcorr.c: Modified to no longer call pixOK function before operating on a pixel. Instead, do the calculation for all pixels. (PR 66080; Trac #607)

Updates for Version 2.1 - 15 May 2010 (HAB):

- calwf3.cl, wf32d.cl, wf3ccd.cl, wf3rej.cl, wf3ir.cl: Increment version to 07May2010.
- wf3version.h: Increment version to 2.1 and date to 07-May-2010.
- calwf3/procccd.c: Modified logic and processing flow so that if CRCORR=PERFORM and EXPSCORR=PERFORM, run wf32d on the individual exposures *after* crcorr is complete, so that CR flags

inserted into blv_tmp files by crcorr will show up in final flt images produced by wf32d. (PR 64963; Trac #545)

- wf3rej.cl: Modified to place the user-supplied crsigmas param string in quotes when appending to the command line, so that embedded blanks don't cause problems for the parser. (PR 64941; Trac #544)
- wf3rej/readpar.c: Updated the strtor routine to match the one in the calstis lib, which skips over leading and embedded blanks in the string. (PR 64941; Trac #544)
- wf3ir/darkcorr.c: Updated the darkcorr routine to compute and populate the MEANDARK keyword. (PR 65151; Trac #560)
- wf3ir/doir.c: Swapped the execution order of darkcorr and nlincorr, so now nlincorr goes first. (PR 64854; Trac #536)

Updates for Version 2.0 - 08 Mar 2010 (HAB):

- calwf3.cl, wf32d.cl, wf3ir.cl, wf3rej: Increment version to 08Mar2010.
- wf3version.h: Increment version to 2.0 and date to 08-Mar-2010.
- calwf3/wf3dth.c: Eliminated the creation of dummy drz products. Now done with PyDrizzle. (PR 64261; Trac #495)
- lib/mkspt.c: Modified to allow for the case where there are no input spt files, in which case don't try to create or update the output spt header. (PR 64260; Trac #494)
- wf32d/doflat.c: Modified divFlat to use mean_gain for all images, including grisms. (PR 64259; Trac #493)
- wf3ir/blevcorr.c: Swapping order of zsig and blev such that zsig occurs first requires sending zoff image to blevcorr to be processed. (PR 64262; Trac #496)
- **wf3ir/cridcalc.c:**
  - Added check for pixels already saturated in zeroth read (detected by zsigcorr), in which case outputs set to zero.
  - Switch from using commanded ccdgain to mean_gain.
  - Modified linfit to include readnoise in sample weights and Poisson noise from source in final fit uncertainty.
  - Added SPIKE_THRESH in RejSpikes to use a separate rejection threshold from CR thresh.
  - Updated hardwired dark and readnoise to use SMOV results. Some general cleanup. (PR 64630; Trac #518)
- **wf3ir/doir.c:**
  - **Changed order of processing so that doZsig is called before doBlev. This also requires passing zoff image to**
    - Compute zero-read sample time (sampzero) here instead of in zsigcorr. (PR 63711; Trac #457)
- wf3ir/flatcorr.c: Modified mult_gain to use mean_gain for all images, including grisms. (PR 64259; Trac #493)
- wf3ir/refdata.c: Fixed initialization of maxcrsplit variable.
- wf3ir/unitcorr.c: No longer need to check status of ZSIGCORR before using sampzero, because sampzero is always computed in doIR. (PR 63711; Trac #457)
- **wf3ir/zsigcorr.c:**

– Set ZEROSIG DQ values along with SATPIXEL flags. Set and count pixels as saturated in first read if they'

* Moved computation of sampzero into doIR. (PR 63711; Trac #457)

Updates for Version 1.8.1 - 27 Oct 2009 (HAB):

- calwf3.cl, wf3ir.cl: Increment version to 27Oct2009.
- wf3version.h: Increment version to 1.8.1 and date to 27-Oct-2009.
- wf3ir/cridcalc.c: Fixed the crrej routine to use the logical OR of all input flags for the output DQ flag value for pixels that have all samples rejected, instead of simply flagging them all as SATURATED. (PR 63806; Trac #459)

Updates for Version 1.8 - 26 Oct 2009 (HAB):

- calwf3.cl: Increment version to 26Oct2009.
- wf3version.h: Increment version to 1.8 and date to 26-Oct-2009.
- wf3info.h: Added new "mean_gain" parameter to WF3Info structure. (PR 63788; Trac #458)
- lib/getccdtab.c: Added computation of mean_gain to GetCCDTab. (PR 63788; Trac #458)
- lib/wf3info.c: Added initialization of new mean_gain parameter. (PR 63788; Trac #458)
- wf32d/doflat.c: Upgraded divFlat to use mean_gain when applying gain calibration, except for grism images, which still use the amp-dependent gain values.(PR 63788; Trac #458)
- wf3ir/flatcorr.c: Upgraded mult_gain to use mean_gain when applying gain calibration, except for grism images, which still use the amp-dependent gain values. (PR 63788; Trac #458)

Updates for Version 1.7 - 14 Oct 2009 (HAB):

- calwf3.cl: Increment version to 14Oct2009.
- wf3version.h: Increment version to 1.7 and date to 14-Oct-2009.
- wf3ir/cridcalc.c: Updated the crrej routine to use the badinpdq value from the CRREJTAB to set the DQIGNORE constant, which is used to reject samples, rather than having it hardwired to a value in the code. The hardwired value had been set to just SATPIXEL, which meant that pixels flagged with other values such as DETECTORPROB (4), BADZERO (8), HOTPIX (16), and UNSTABLE (32) were not being blanked out in the output flt image. (PR 63556; Trac ticket #454)
- wf3ir/refdata.c: Updated the crrpar_in routine to report the value of badinpdq, now that it's being used in cridcalc. (PR 63556; Trac ticket #454)

Updates for Version 1.6 - 17 Aug 2009 (HAB):

- calwf3.cl: Increment version to 17Aug2009.
- wf3version.h: Increment version to 1.6 and date to 17-Aug-2009.
- calwf3/wf3dth.c: Updated to set NEXTEND=3 in header of dummy drz file for IR images. (PR 63286; Trac ticket #436)
- ib/mkspt.c: Updated mkNewSpt to count total number of input spt extensions before updating NEXTEND in output spt file. (PR 63286; Trac ticket #436)
- wf3ir/flatcorr.c: Updated flatcorr routine to set BUNIT to electrons after gain correction has been applied. (PR 63063; Trac ticket #435)
- wf3rej/cr_history.c: Updated to set NEXTEND=3 in header of output crj file for IR images. (PR 63286; Trac ticket #436)

Updates for Version 1.5 - 24 Jun 2009 (HAB):

- calwf3.cl: Increment version to 24Jun2009.
- wf3version.h: Increment version to 1.5 and date to 24-Jun-2009.
- calwf3/procccd.c: Added logic to always use CRCORR=PERFORM internally for both CRJ and RPT associations, instead of using CRCORR for one and RPTCORR for the other.
- wf3rej/rej_check.c: Added logic to getampxy routine to reset ampx/ampy to correct values for IR subarray images. (PR 62948; Trac ticket #424)
- wf3rej/rej_sky.c: Commented out print statement that had inadvertanly been left active in a previous delivery that was only intended for debugging use.

Updates for Version 1.4.1 - 27 Apr 2009 (HAB):

- calwf3.cl: Increment version to 27Apr2009.
- wf3version.h: Increment version to 1.4.1 and date to 27-Apr-2009.
- wf3ccd/doblev.c: Added verbose mode print statements to indicate the overscan column limits being used in the overscan calculations. (Trac ticket #405)
- wf3ccd/findover.c: Fixed the logic that was used to compute the biassect values when dealing with a subarray that includes the physical overscan on the amp B/D edge of the image. (Trac ticket #405)

Updates for Version 1.4 - 14 Apr 2009 (HAB):

- calwf3.cl: Increment version to 14Apr2009.
- wf3version.h: Increment version to 1.4 and date to 14-Apr-2009.
- lib/interpinfo.c: Added checks to make sure pixel fractions "q" and "p" come out between 0.0 and 1.0. (Trac ticket #325)
- lib/unbin2d.c: Added checks to make sure pixel fractions "q" and "p" come out between 0.0 and 1.0. (Trac ticket #325)
- lib/unbinline.c: Added checks to make sure argument of sqrt() is positive. (Trac #325)
- wf32d/doflat.c: Fixed bugs that were causing the routine to crash when trying to interpolate a binned LFLTFILE and also added the capability to do a direct division into science image if they are the same size. Added forced return if LFLTFILE is binned, until we upgrade the interpolation routines to work better. (Trac ticket #325)
- wf3ir/refdata.c: Fixed bugs in crrpar_in routine for calls to c_tbegti to read value of IRRAMP column in each row of the crrejtab. (Trac ticket #392)
- wf3ir/resistmean.c: Upgraded computations of mean and standard deviation to use double precision variables. Original single-precision calculations were giving different results on different computer platforms. Also did some general code clean-up. (Trac ticket #391)

Updates for Version 1.3 - 13 Mar 2009 (HAB):

- calwf3.cl: Increment version to 13Mar2009.
- wf3version.h: Increment version to 1.3 and date to 13-Mar-2009.
- wf3info.h: Added "crrej" to WF3Info structure for the CRREJTAB ref table, now that it's being used within calwf3 in wf3ir/cridcalc step. Previously, it was only accessed from within wf3rej. (Trac ticket #352)
- wf3ccd/dobias.c: Updated to compute correct x-offset values for subarrays in the amp B and D quadrants, which need to take into account the columns of serial virtual overscan that are in the middle of a 4-amp bias reference image. (Trac ticket #378)
- **wf3ir/cridcalc.c:**

  - Added use of CRREJTAB to allow user input of CR rejection threshold instead of having it hardwired in the code.
  - Decreased max_CRs from 6 to 4. Reinstated old loop limits code that excludes reference pixels from ramp fitting. Fixed bug in logic that identifies pixels already saturated in first read.
  - Don't set HIGH_CURVATURE flag in ouput DQ arrays, use UNSTABLE instead, and change messages to say UNSTABLE.
  - Also don't set ZEROSIG value in output crimage (flt file) DQ array, because those pixels are still OK (assuming no other flag also set).
  - Removed unnecessary call to EstimateDarkandGlow at end of processing.
  - Fixed calculation of output SAMP and TIME values.
  - Fixed bug in logic that identifies pixels with only 1 good sample.
  - Fixed bug in computation of "firstgood" and "lastgood" assignments for pixels with no acceptable samples. (Trac tickets #352, 365, 376, 377, 381)

- wf3ir/getirflags.c: Added new checkCRRej routine to check for the existence and correctness of the CRREJTAB ref table, for use in CRCORR. (Trac ticket #352)
- wf3ir/refdata.c: Added crrpar_in routine to load parameters from CRREJTAB ref table, for use in CRCORR. (Trac ticket #352)

Updates for Version 1.2a - 20 Feb 2009 (HAB):

- calwf3.cl: Increment version to 20Feb2009.
- wf3version.h: Increment version to 1.2a and date to 20-Feb-2009.
- wf3rej/rej_loop.c: Fixed bug in test to exclude flagged pixels from being tested for CR's so that pixels previously marked as SPILL still get tested to see if they qualify as a CR. (PR 62005)

Updates for Version 1.2 - 29 Jan 2009 (HAB):

- calwf3.cl: Increment version to 29Jan2009.
- wf3version.h: Increment version to 1.2 and date to 29-Jan-2009.
- wf3.h: Added new parameter "type" to RefImage and RefTab structures, which contains the value of the FILETYPE keyword for each reference file. (PR 61608)
- wf3dq.h: New WFC3 UVIS and IR DQ flag assignments. (PR 61741)
- lib/div1d.c: Changed CALIBDEFECT macro to BADFLAT, to coincide with WFC3 DQ assignment changes. (PR 61741)
- lib/getkeys.c: Eliminated use of default values for FILTER and CCDGAIN keywords, which means it will now be an error if they are missing. (PR 61608)
- lib/imgpedigree.c: Upgraded to retrieve FILETYPE along with PEDIGREE/DESCRIP keywords. (PR 61608)
- lib/tabpedigree.c: Upgraded to retrieve FILETYPE along with PEDIGREE/DESCRIP keywords, and to retrieve these keywords from the primary HDU, not the table HDU. (PR 61608)
- lib/trlbuf.c: Fixed bug in CloseTrlBuf causing an IRAF segv, which was due to a call to fclose with a NULL pointer as argument. Removed the call. (PR 61164)
- lib/wf3info.c: Added initialization of new 'type' parameter in InitRefImg and InitRefTab. Also added new CheckImgType, CheckTabType, CheckFilter, CheckDetector, and CheckGain routines. (PR 61608)
- lib/whicherror.c: Added case of an invalid ref file to error message. (PR 61608)

- wf32d/dophot.c: Added computation of PHOTFNU keyword value, to be consistent with IR photcorr process. Also removed some old ACS-specific code that is not used for WFC3. (PR 61138)
- wf32d/get2dflags.c: Upgraded all the checkNNNN routines to verify correct FILETYPE for each reference file, as well as correct selection criteria such as DETECTOR, FILTER, and CCDGAIN. (PR 61608)
- wf32d/photmode.c: Modified construction of photmode string to use separate UVIS1/UVIS2 keywords for the CCD chips, to add the new "cal" keyword for UVIS exposures, and to remove the "DN" keyword for IR exposures because they're now in units of electrons. (PR 61497)
- wf3ccd/blevdrift.c: Upgraded the cleanDriftFit routine to only use the good values returned by VMedianY in the computation of statistics and rejection of outliers in the array of bias values. Also added checks for potential divide-by-zero conditions. (PR 61698)
- wf3ccd/doblev.c: Upgraded the cleanBiasFit routine to only use the good values returned by FindBlev in the computation of statistics and rejection of outliers in the array of bias values. Also added checks for potential divide-by-zero conditions. (PR 61698)
- wf3ccd/getflags.c: Upgraded all the checkNNNN routines to verify the correct FILETYPE for reference file, as well as correct selection criteria such as DETECTOR, FILTER, and CCDGAIN. (PR 61608)
- wf3ir/cridcalc.c: Reinstated code that had been inadvertantly removed from the calnica code ported to calwf3, which propogates CR DQ flags to all samples following a hit. (PR 61425)
- wf3ir/dqicorr.c: Updated to check for missing CCDGAIN and CCDAMP columns in BPIXTAB and default to a match with the science data (same logic as in lib/dodqi.c). (PR 61436)
- wf3ir/flatcorr.c: Fixed bug in mult_gain routine that was doing out of bounds array access for subarray images. (PR 61428)
- wf3ir/getirflags.c: Upgraded all of the checkNNNN routines to verify the correct FILETYPE for each reference file, as well as verifying a match with selection criteria such as DETECTOR and FILTER. (PR 61608)
- wf3ir/refdata.c: Removed FILTER check from getFlatImage because that's now handled by checkFlat in getIRFlags. (PR 61608)
- wf3ir/zsigcorr.c: Fixed bug in zsigcorr routine to compute correct zeroth read exposure time for subarray images, rather than using SAMPZERO keyword value, which is only correct for full-frame images. (PR 61347)
- wf3rej/rej_loop.c: Fixed problems with DQ flags written to input and output DQ arrays, including not setting any SPILL flags (CR only), not setting CR flags in the ouput CRJ file for pixels that had at least 1 good input, and not propagating CR flags set for one input file into the remaining files in the input list. (PR 61819)

Updates for Version 1.1 - 10 Oct 2008 (HAB):

- calwf3.cl: Increment version to 10Oct2008.
- wf3version.h: Increment version to 1.1 and date to 10-Oct-2008.
- calwf3/procccd.c: Fixed handling of EXPSCORR=PERFORM so that WF32D gets called for all images, and fixed save_tmp setting so that blv_tmp files get deleted after EXPSCORR processing.
- wf32d/doflat.c: Added 'applygain' switch to divFlat to turn on/off the gain correction so that the gain will only be used to correct one ref file and not both, otherwise the gain will be applied twice to the science data.
- **wf3ccd/blevdrift.c:**
  - **Added new routine cleanDriftFit to reject outliers from parallel overscan array before fitting (as in serial ro**

        * Added readnoise as an input argument to use in cleanDriftFit.

- wf3ccd/blevfit.c: Modified fit report in BlevFit to indicate that results are for the serial overscan fit.
- wf3ccd/doblev.c: Added readnoise as an input argument to BlevDrift. Modified cleanBiasFit to use different clip values on each pass through data.
- wf3ir/flatcorr.c: Upgraded to convert data to units of electrons by multiplying by the gain after flat field has been applied. Uses new function "mult_gain".
- wf3rej/rej_sky.c: Added capabilities for "mean" sky calculation, using resistmean.

Updates for Version 1.0 - 11 Sep 2008 (HAB):

- calwf3.cl: Increment version to 11Sep2008.
- wf3version.h: Increment version to 1.0 and date to 11-Sep-2008.
- wf3info.h: Added ncoeff and nerr to NlinData structure.
- wf3sizes.h: Removed this old include file, which isn't used anywhere.
- wf3ir/blevcorr.c: Modified to use statistics from all ref pixels in each readout, rather than working quad-by-quad. Uses new statistics module "resistmean".
- wf3ir/mkpkg: Added new module "resistmean.c" to library list.
- wf3ir/nlincorr.c: Modified to use 3rd-order coeffs and new ncoeff, nerr members of NlinData struct.
- wf3ir/refdata.c: Modified getNlinData and freeNlinData to use new ncoeff, nerr members of NlinData struct.
- wf3ir/resistmean.c: New statistics module to compute resistant mean.

Updates for Version 0.99 - 28 Aug 2008 (HAB):

- calwf3.cl: Increment version to 28Aug2008.
- wf3version.h: Increment version to 0.99 and date to 28-Aug-2008.
- calwf3/wf3init.c: Changed all occurences of "_dth" to "_drz".
- calwf3/wf3table.c: Changed all occurences of "_dth" to "_drz".
- lib/detnsegn.c: Removed unnecessary old ACS/HRC code and added WFC3/IR functionality.
- lib/findroot.c: Changed all occurences of "_dth" to "_drz".
- lib/getccdtab.c: Modified to only reset ampx for UVIS subarrays, not for IR.
- lib/mkspt.c: Changed all occurences of "_dth" to "_drz".
- wf3ir/doir.c: Added call to GetGrp at beginning of processing to load LTV offsets.
- wf3ir/noiscalc.c: Upgraded to use separate gain and readnoise values for each amp quadrant of the images. Includes support for IR subarrays.
- wf3rej/rej_do.c: Set non_zero=nimgs for case where all images have exptime=0, so that they'll still process using exptimes reset to 1.
- wf3rej/rej_init.c: Fixed indexing of SQ(noise.val[0]) to SQ(noise.val[k]) in loop over amps so that appropriate readnoise values get used for each amp.

Updates for Version 0.95 - 21 Jul 2008 (MS):

- calwf3.cl: Increment version to 21Jul2008.
- wf3version.h: Increment version to 0.96 and date to 21-Jul-2008.

- wf3ir/cridcal.c: Major rewrite to incorporate new CR rejection and err computation methods from latest calnica/n_cridcalc.c.

Updates for Version 0.9 - 19 Jun 2008 (HAB):

- calwf3.cl: Increment version to 19Jun2008.
- wf3version.h: Increment version to 0.9 and date to 19-Jun-2008.
- calwf3/procir.c: Added logic and supporting functionality to call WF3Rej_0 to combine IR Repeat-Obs images into a crj product.
- calwf3/wf3dth.c: Restored old acsdth code for creating dummy dth output products, until MultiDrizzle capability is added to WFC3 pipeline.
- lib/mkspt.c: Corrected the calculation of the number of extensions in output spt files for WFC3 (IR files have a pair of extensions for each nsamp).
- wf3rej/cr_history.c: Update RPTCORR, instead of CRCORR, for IR images. Required adding detector as input argument.
- wf3rej/rej_do.c: Pass detector to cr_history.
- wf3rej/rej_sky.c: Avoid arithmetic overflow in binning calculations.
- wf3rej/wf3rej.c: Various updates to properly ID and handle IR images.

Updates for Version 0.8 - 21 Dec 2007 (HAB):

- calwf3.cl: Increment version to 21Dec2007.
- wf3dq.h: Change ZEROSIG DQ value from 4096 to 64, to leave 4096 free for Multidrizzle CR flag.
- wf3version.h: Increment version to 0.8 and date to 21-Dec-2007. lib/dodqi.c: Use new FirstLast routine (provided by P. Hodge) to fix problems with indexing in binned images.
- lib/mkspt.c: Added handling of SNAP1 extensions, in addition to UDL extensions, including appropriate mods to output NEXTEND.
- wf3ir/blevcorr.c: Removed code put in place in previous version to swap quad indexes for images processed before a certain date, because all old images have now been reprocessed to latest orientation. Also updated quad numbering scheme to latest (1 in upperleft and going counter-clockwise from there).
- wf3rej/rej_init.c, rej_loop.c, rej_sky.c: Added calls to hstio getHeader before each call to getShortLine, in order to prevent getShortLine from crashing on null input DQ arrays. In order to handle null arrays, getShortLine needs to access the image header.

Updates for Version 0.7 - 09 May 2007 (HAB):

- calwf3.cl: Increment version to 09May2007.
- wf3info.h: Added "subtype" to WF3Info structure for use with IR subarrays.
- wf3version.h: Increment version to 0.7 and date to 09-May-2007.
- calwf3/getinfo.c: Changed default gain for IR channel from 2.0 to 2.5 in GetIRInfo routine.
- lib/dodqi.c: Modified to allow for wildcard values in BPIXTAB Amp, Gain, and Chip columns (following CALACS change).
- lib/getkeys.c: Updated default gain for IR channel from 2.0 to 2.5. Added 'subtype' to list of IR keywords loaded. Changed default sampzero value to 2.911755 sec.
- wf32d/do2d.c: Modified call to PhotMode to use science extension header, rather than primary header, because that's where phot keywords are.
- **wf32d/photmode.c:**

  - Changed UVIS channel detector keyword to always use "UVIS1".
  - Changed use of "A2Dx" gain keyword to "DN" and eliminated use of it for UVIS images because flatfielding leaves them in units of electrons, not counts.

- wf3ir/darkcorr.c: Eliminated use of RebinRef, because we don't want to extract subarrays from a full-frame dark ref image, we want to instead use a matching subarray dark ref image.
- wf3ir/getirflags.c: Added logic to checkDark to turn off zsigcorr if dark=dummy.
- **wf3ir/imageio.c:**
  - **Enhanced copyGroup to only copy filename if input name is not Null.**
    * Added new putCalDataSect routine.
- **wf3ir/refdata.c:**
  - Reduced ALLOWDIFF from 0.1 to 0.01 for use with IR subarray exptimes.
  - Added check for SUBTYPE in getDarkInfo.
- wf3ir/wf3ir.c: Added use of new putCalDataSect routine to write out calibrated images that have the ref pixels trimmed off.

Updates for Version 0.61 - 01 Aug 2006 (HAB):

- calwf3.cl: Increment version to 01Aug2006.
- wf3version.h: Increment version to 0.61 and date to 01-Aug-2006.
- wf3ccd/doblev.c: Fixed logic used to select the appropriate readnoise value to pass to the FitToOverscan routine and to convert the readnoise value to units of DN, so that it matches the science data.
- wf3ir/blevcorr.c: Enhanced the blevcorr routine to swap the quad indexes around for raw images generated before and after the date on which OPUS starting transposing the raw IR images.
- wf3ir/flatcorr.c: Switched routine from multiplying by (inverse) flats to dividing by flats.
- wf3ir/math.c: Upgraded adiv and adiv_noref routines to avoid divide by zero errors when computing output err values.

Updates for Version 0.6 - 17 Jul 2006 (HAB):**

- calwf3.cl: Increment version to 17Jul2006.
- wf3version.h: Increment version to 0.6 and date to 17-Jul-2006.
- calwf3/calwf3.c: CalWf3Run routine modified to remove updateAsnStat routine, because only OPUS should update the ASN_STAT keyword in asn tables.
- calwf3/procccd.c: ProcessCCD routine modified to use new "wf3rej_msgtext" string variable to hold (potentially) very long list of input file names for printing. Sometimes too long for regular MsgText string variable.
- calwf3/refexist.c: RefExist routine modified to include check for ref file names that are null (""), in addition to existing check for "N/A".
- calwf3/wf3dth.c: InitDthTrl routine modified to fix "trl_in" memory allocation problem for holding long list of trailer file names.
- calwf3/wf3table.c: getAsnTable routine modified to only populate sub-products if at least one input exists for that product.
- lib/dodqi.c: DoDQI routine modified to properly handle binned images, and to adjust flagged pixel coords read from BPIXTAB for presence of serial virtual overscan in WFC3 raw images.

- lib/mkoutname.c: MkOutName routine modified to include calls to "free", to free local memory before all error returns.
- wf3ccd/blevdrift.c: VMedianY routine modified to fix bug in "if"-statement logic being used to reject flagged pixels from the parallel overscan region. Flawed logic was allowing flagged pixels to remain in computation.
- wf3ccd/findblev.c: FindBlev routine modified to fix bug in "if"-statement logic being used to reject flagged pixels from the serial overscan regions. Flawed logic was allowing flagged pixels to remain in computation.

Updates for Version 0.5 - 08 Nov 2005 (HAB):

- calwf3.cl: Increment version to 08Nov2005.
- wf3version.h: Increment version to 0.5 and date to 08-Nov-2005.
- wf32d/do2d.c: Modified logic in OscnTrimmed routine to make it compatible with WFC3 binned images.
- wf3ir/blevcorr.c: Fixed bug in calculation of j2 loop limit for reference pixel regions for quads 3 and 4.
- wf3ir/nlincorr.c: Fixed bug in calculation of nlin ref image pixel indexes.
- wf3ir/noiscalc.c: Fixed bug in noise computation by adding use of "noise2" variable to temporarily store value of readnoise-squared.
- wf3ir/zsigcorr.c: Fixed bug in calculation of nlin ref image pixel indexes.

Updates for Version 0.4 - 14 Feb 2005 (HAB):

- calwf3.cl: Increment version to 14Feb2005.
- wf3rej.cl: Increment version to 14Feb2005.
- wf3version.h: Increment version to 0.4 and date to 14-Feb-2005.
- wf3ccd/findover.c: Enhanced FindOverscan routine to handle IR images differently than UVIS, selecting oscntab row based on image size (nx,ny) instead of binning.
- wf3ir/blevcorr.c: Enhanced to set reference pixel statistics computation limits based on biassect values in oscntab, rather than image trim values.
- wf3rej/wf3rej.c: Fixed memory reallocation in InitRejTrl that was causing a crash for very large numbers of input images. Made reallocation increment much larger, so that it doesn't get called repeatedly.

Updates for Version 0.3 - 20 Feb 2004 (HAB):

- calwf3.cl: Increment version to 0.3.
- wf3.h: Added ATOD_SATURATE macro definition.
- wf3dq.h: Added ATODSAT dq value of 2048 and changed existing ZEROSIG from 2048 to 4096.
- wf3version.h: Incremented version to 0.3 and date to 20-Feb-2004.
- lib/dodqi.c: Modified to make CCDAMP and CCDGAIN columns optional when looking for matching rows in BPIXTAB. Added handling of new ATODSAT dq flag.
- lib/donoise.c: Fixed use of amp boundaries to take into account WFC3 serial virtual overscan regions.
- lib/getccdtab.c: Changed use of wf3->binaxis to wf3->bin to make it work properly for binned science images.
- lib/getgrp.c: Eliminated the ACS practice of hardwiring wf3->bin to 1 and instead populate it by reading BINAXIS keywords from sci extension header.
- lib/getkeys.c: Eliminated attempt to read BINAXIS keywords from primary header because for WFC3 they're in the sci extension header.

- lib/loadhead.c: Minor code cleanup.
- **wf3ccd/doblev.c:**
  - Implemented limit on sdev to be sqrt(mean) for first pass in CleanBiasFit and use readnoise as value of sdev for second pass.
  - Added readnoise ('rn') as input to cleanBiasFit.
- wf3ccd/doccd.c: Minor comment change.
- wf3rej/rej_loop.c: Commented out unused LoadHdr function declaration. Removed SQ(scale*val) from sumvar computation. Changed AllocBitBuff to work with arbitrary buffer sizes rather than only those evenly divisible by 8.

Updates for Version 0.2 - 28 Oct 2003 (HAB):

- **wf3info.h:**
  - **Changed datatype of 'ccdgain' from int to float.**
    - Added 'blev(NAMPS)' to WF3Info struct so WF3CCD can remember all blev values for all extensions/amps.
  - Added 'expscorr' to WF3Info struct for use in WF32D.
- wf3version.h: Incremented version to 0.2 and date to 28-Oct-2003.
- wf3wild.h: Added 'FLT_WILDCARD' and 'FLT_IGNORE' macros for use in floating-pt get/put keyword functions.
- calwf3/calwf3.h: Changed datatype of 'scigain' from int to float.
- calwf3/calwf3.c: Removed unique code for RPTCORR processing and made it same as CRCORR for UVIS images.
- calwf3/getinfo.c: Changed datatype of 'scigain' values from int to float.
- calwf3/getrefffiles.c: Load 'CRREJTAB' ref table if RPTCORR is turned on (to make it same as CRCORR for UVIS images).
- calwf3/getswitches.c: Changed to handle RPTCORR switch the same as CRCORR for UVIS images.
- **calwf3/procccd.c:**
  - **Changed to handle RPTCORR processing same as CRCORR for UVIS images.**
    - Added check on status value returned from WF3Rej. If set to 'NO_GOOD_DATA', it will reset 'wf3hdr->sci_basic_2d' to 'SKIPPED' so that no further processing will be performed. It then resets the status value to 'WF3_OK' for continuing normally.
- calwf3/wf3table.c: Changed to handle RPTCORR processing same as CRCORR for UVIS images.
- lib/wf3info.c: Added initialization of new wf3->blev array.
- lib/dodqi.c: Updated to treat commanded gain values as float datatype instead of int.
- lib/donoise.c: Added logic to use Amp C/D bias values from new blev array for UVIS Chip 2 instead of relying on 'AMPY' logic.
- lib/getccdtab.c: Updated to treat commanded gain values as float datatype instead of int.
- lib/getkeys.c: Updated to treat commanded gain values as float datatype instead of int.
- lib/mkspt.c: Updated a couple of printf statements to use trlmessage so that the comments on creating the SPT file also make it to the trailer file.
- lib/sameint.c: Added new 'SameFlt' routine for use with gain keyword values.

- lib/trlbuf.c: Increased 'trldata' buffer size from 'SZ_FNAME' to 'SZ_LINE'.
- lib/key.c: Changed putKeyBool function type from Bool to int.
- wf32d/wf32d.c: Added 'expscorr' switch as command-line argument for wf32d.
- wf32d/do2d.c: Update final state of 'expscorr' switch in output header.
- wf32d/photmode.c: Updated to treat gain values as float datatype instead of int.
- wf3ccd/wf3ccd.c: Populate BIASLEV[abcd] keywords in output header using new 'BiasKeywords' function.
- wf3ccd/blevfit.c: Added BlevResults function to return the values of the slope and intercept computed for the bias fit. Also, the fit reports the values to the user in a trailer message.
- wf3ccd/doatod.c: Updated to treat commanded gain values as float datatype instead of int.
- **wf3ccd/doblev.c:**
  - Added 'cleanBiasFit' routine to do sigma-clipping on bias measurements before computing fit.
  - Set default ccdbias value to be AMP C/D value for UVIS Chip 2 data where no overscan was available for computing the bias level.
  - Modified to load the 'biassect' array with indexes corresponding to the serial physical overscan regions, instead of serial virtual overscan regions, when processing UVIS subarray images (which have noserial virtual overscan).
- **wf3ccd/doccd.c:**
  - **Added processing msg's giving info on bias levels for each amp.**
    - Upgraded to do correct overscan trimming of output image for UVIS subarray modes, in which there's no serial virtual overscan to remove, and variable amounts of serial physical overscan.
- wf3ccd/findover.c: Modified to zero-out all serial and parallel virtual biassect and trim values when processing UVIS subarray images (which don't have any virtual overscan). Also fixed a bug in which one of the biassect values was not being converted from 1-indexed to 0-indexed in the case of subarray images.
- wf3ir/dqicorr.c: Updated to treat commanded gain values as float datatype instead of int.
- wf3ir/getirflags.c: Modified to load DARKCORR and NLINCORR switch settings and DARKFILE and NLINFILE ref file info if ZSIGCORR is set to PERFORM.
- wf3ir/nlincorr.c: Modified to use just 1 node array from the NLINFILE ref data, which is the saturation value. There won't be another node array specifying the lower bound of the nlin correction as with NICMOS.
- **wf3ir/refdata.c:**
  - Modified to load just 1 node array from the NLINFILE ref file.
  - Also modified to combine all of the PFLT, DFLT, and LFLT ref file data (if present) into a master flat, as is done for UVIS processing.
- wf3ir/zsigcorr.c: Modified to use just 1 node array from the NLINFILE ref data, which is the saturation value.
- wf3rej/wf3rej.c: Added call to 'mkNewSpt' within error condition for wf3rej_do to always produce a new SPT file for product when possible. This also involved remembering the value of the error condition, setting it to WF3_OK, calling 'mkNewSpt', then resetting to old value in order to allow 'mkNewSpt' to work successfully.
- **wf3rej/rej_do.c:**

  - Added code to count number of inputs with exptime>0. If some are zero, new code will insure that first good image gets used to initialize the initial guess image.
  - Revised to handle cases where 0,1,or more input are valid. If none have exptime>0, skips wf3rej_loop altogether and output a blank image with DQ values of 1 and ERR values of 0 with the exception of the 0,0 pixel, which have values of 8 and -1 respectively, to forces HSTIO to write out the image arrays. It now returns status=NO_GOOD_DATA if there are no inputs with exptime>0.

- **wf3rej/rej_init.c:**
  - Added code to count number of inputs with exptime>0.
  - Also now checks whether exptime!=0 when building initial guess image.
- wf3rej/rej_loop.c: Added code to avoid crashing when exp[n]=0 for an input image. It will now skip all the detection code when exp[n]=0.
- wf3rej/cr_scaling.c: Added trailer file comments to better describe how exptime=0 cases are handled.

Updates for Version 0.1 - 26 Nov 2002 (HAB):

- Initial installation of baseline CALWF3 into stlocal$testwf3 pkg.

CHAPTER 2

Analysis Tools

2.1 embedsub

Given an image specified by the user which contains a subarray readout, return a full-frame image with the subarray implanted at the appropriate location.

2.1.1 Usage

```
>>> python
>>> from wfc3tools import embedsub
>>> embedsub(files)
```

2.1.2 Parameters

files [file] Input image name or list of image names. The rootname will be used to create the output name

2.1.3 Returns

Return the full-frame location of the subarray coordinates using a file specified by the user.

2.1.4 Example Output

This method calls wfc3tools.sub2full to calculation the subarray position on the full frame image.

This is the default output:

```
> wfc3tools.embedsub.embedsub('ic5p02eeq_flt.fits')
> Subarray image section [x1,x2,y1,y2] = [2828:3339,215:726]
> Image saved to: ic5p02eef_flt.fits
```

2.2 pstat

Plot statistics for a specified image section up the stack of a an IR MultiAccum image. Sections from any of the SCI, ERR, DQ, image extensions can be plotted. A choice of mean, median, mode, standard deviation, minimum and maximum statistics is available. The total number of samples is determined from the primary header keyword

NSAMP and all samples (excluding the zeroth-read) are plotted. The SCI, ERR, DQ statistics are plotted as a function of sample time. The sample times are read from the SAMPTIME keyword in the SCI header for each readout.

SAMP and TIME aren't generally populated until the FLT image stage To plot the samptime vs sample, use wfc3tools.pstat and the "time" extension

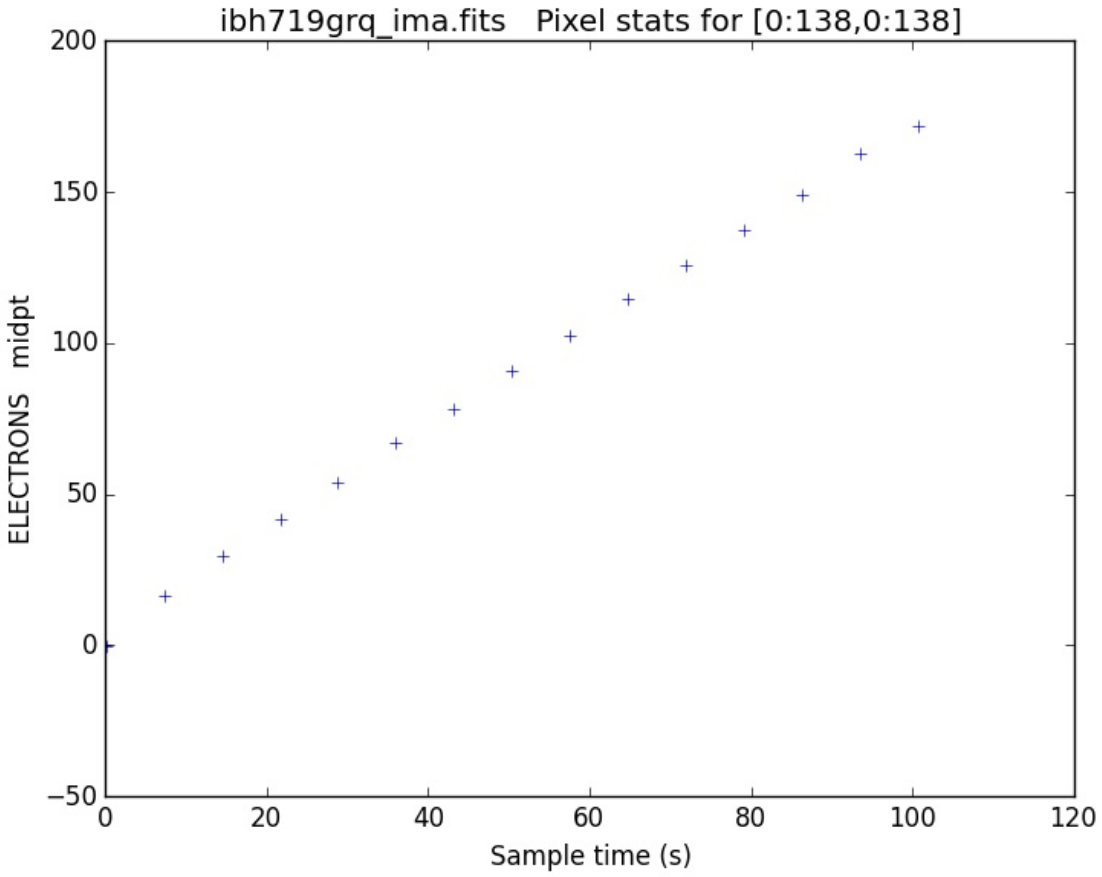


Fig. 2.1: An example of the default plot from pstat

```
In [1]: pstat('ibh719grq_ima.fits')
```

The plotting data is returned as two arrays:

```
Out[2]:
(array([ 100.651947,   93.470573,   86.2892  ,   79.107826,   71.926453,
          64.745079,   57.563702,   50.382328,   43.200954,   36.019581,
          28.838205,   21.65683 ,   14.475455,    7.29408 ,    0.112705,
           0.      ]),
 array([ 171.85813415,  162.44643275,  148.99918831,  137.60458714,
         125.91510743,  114.71149769,  102.63038466,   90.97130078,
          78.26479365,   67.16825321,   53.6897243 ,   41.71639092,
          29.36628817,   16.47151355,   -0.20821257,    0.        ]))
```

If you want to save the plotting data into the variables `time` and `counts` you can issue the command like this:

```
In [1]: time,counts =pstat.pstat('ibh719grq_ima.fits')

In [2]: time
Out[3]:
array([ 100.651947,   93.470573,   86.2892  ,   79.107826,   71.926453,
         64.745079,   57.563702,   50.382328,   43.200954,   36.019581,
         28.838205,   21.65683 ,   14.475455,    7.29408 ,    0.112705,
          0.      ])

In [4]: counts
Out[5]:
array([ 171.85813415,  162.44643275,  148.99918831,  137.60458714,
        125.91510743,  114.71149769,  102.63038466,   90.97130078,
         78.26479365,   67.16825321,   53.6897243 ,   41.71639092,
         29.36628817,   16.47151355,   -0.20821257,    0.        ])
```

Warning: Note that the arrays are structured in SCI order, so the final exposure is the first element in the array

2.2.1 Parameters

- **filename [file]** Input MultiAccum image name with optional image section specification. If no image section is specified, the entire image is used. This should be either a _raw or _ima file, containing all the data from multiple readouts. You must specify just the file name and image section, with no extname designation.
- **extname = "sci" [string, allowed values: sci | err | dq]** Extension name (EXTNAME keyword value) of data to plot.
- **units = "counts" [string, allowed values: counts | rate]** Plot "sci" or "err" data in units of counts or countrate ("rate"). Input data can be in either unit; conversion will be performed automatically. Ignored when plotting "dq", "samp", or "time" data.
- **stat = "midpt" [string, allowed values: mean|midpt|mode|stddev|min|max]** Type of statistic to compute.
- **title = "" [string]** Title for the plot. If left blank, the name of the input image, appended with the extname and image section, is used.
- **xlabel = "" [string]** Label for the X-axis of the plot. If left blank, a suitable default is generated.
- **ylabel = "" [string]** Label for the Y-axis of the plot. If left blank, a suitable default based on the plot units and the extname of the data is generated.
- plot = True [bool] set plot to false if you only want the data returned

2.2.2 Usage

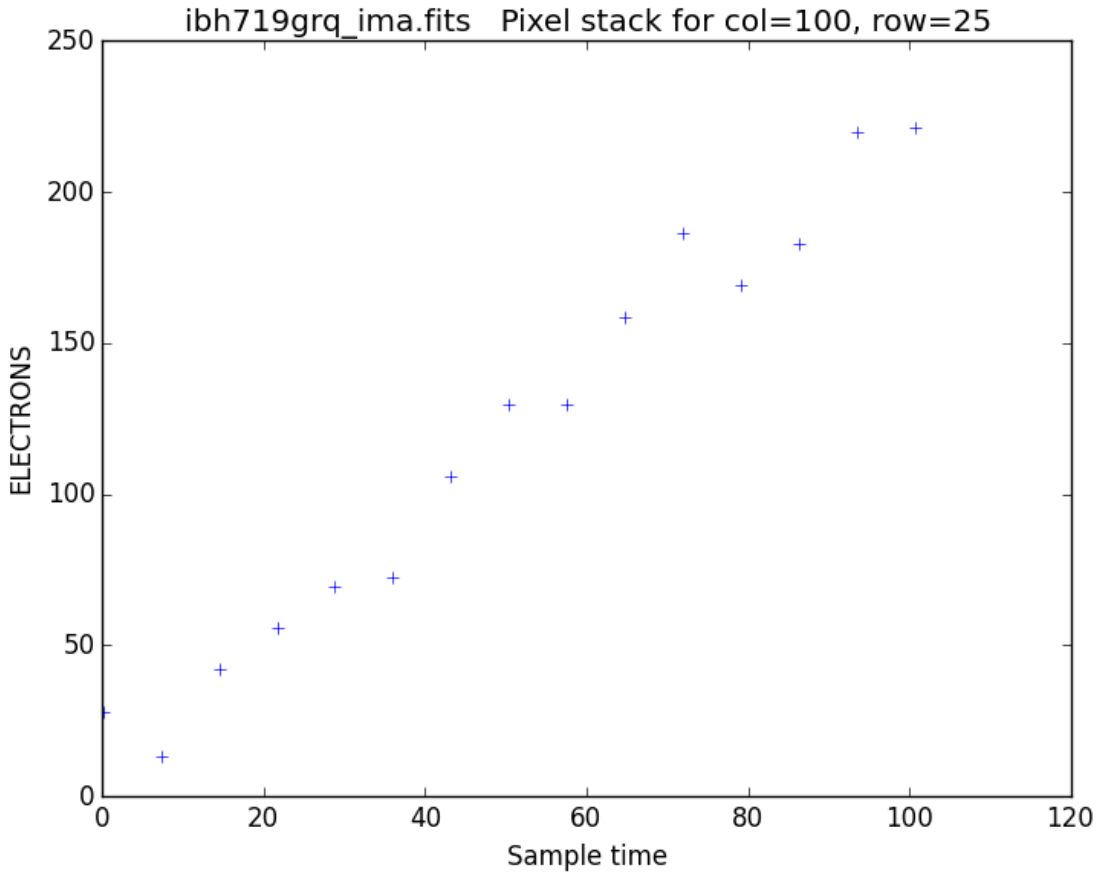
```
pstat.py  inputFilename [pixel range]


>>> python
>>> from wfc3tools import pstat
>>> pstat(inputFilename,extname="sci",units="counts",stat="midpt",title="",xlabel="",ylabel="" )
```

2.3 pstack

Plot the stack of MultiAccum sample values for a specified pixel in an IR multiaccum image. Pixels from any of the SCI, ERR, DQ, or TIME image extensions can be plotted. The total number of samples is determined from the primary header keyword NSAMP and all samples (excluding the zeroth read) are plotted. The SCI, ERR, DQ, values are plotted as a function of sample time, while TIME values are plotted as a function of sample number. The sample times are read from the SAMPTIME keyword in the SCI header for each readout. If any of the ERR, DQ, SAMP, or TIME extensions have null data arrays, the value of the PIXVALUE extension header keyword is substituted for the pixel values. The plotted data values can be saved to an output text table or printed to the terminal.

The BUNIT keyword value is used to determine the starting units of the data, but you can plot either counts or rate using the optional keyword `units`.




```
In [1]: xdata,ydata=pstack.pstack('ibh719grq_ima.fits',column=100,row=25,extname='sci')
```

The plotting data is returned as two arrays, to save them into variables issue the command as above:

```
In [1]: xdata
Out[2]:
array([ 100.651947,   93.470573,   86.2892  ,   79.107826,   71.926453,
         64.745079,   57.563702,   50.382328,   43.200954,   36.019581,
         28.838205,   21.65683 ,   14.475455,    7.29408 ,    0.112705,
          0.      ])
```

```
In [3]: ydata
Out[4]:
array([ 221.36606389,  219.5396653 ,  182.63100095,  169.178308  ,
        186.44084352,  158.3105126 ,  129.46997895,  129.92935701,
        106.14521852,   72.71721539,   69.68652119,   55.98828663,
         42.30755279,   13.12659422,   27.71404187,    0.        ])
```

Warning: Note that the arrays are structured in SCI order, so the final exposure is the first element in the array

2.3.1 Parameters

- **input [file]** Input MultiAccum image name. This should be either a _ima or _raw file, containing all the data from multiple readouts. You must specify just the file name, with no extension designation.
- **col [integer]** The column index of the pixel to be plotted.
- **row [integer]** The row index of the pixel to be plotted.
- **extname = "sci" [string, allowed values: sci | err | dq | samp | time]** Extension name (EXTNAME keyword value) of data to plot.
- **units = "counts" [string, allowed values: counts | rate]** Plot "sci" or "err" data in units of counts or countrate ("rate"). Input data can be in either unit; conversion will be performed automatically. Ignored when plotting "dq", "samp", or "time" data.
- **title = "" [string]** Title for the plot. If left blank, the name of the input image, appended with the extname and column and row being plotted, is used.
- **xlabel = "" [string]** Label for the X-axis of the plot. If left blank, a suitable default is generated.
- **ylabel = "" [string]** Label for the Y-axis of the plot. If left blank, a suitable default based on the plot units and the extname of the data is generated.
- plot = True [bool] set plot to false if you only want the data returned

2.3.2 Usage

```
>>> python
>>> from wfc3tools import pstack
>>> xdata,ydata=pstack(inputFilename,column=x,row=y,extname="sci",units="counts|rate",title="",ylabe
```

2.4 sampinfo

Sampinfo prints information about a WFC3/IR MultiAccum image, including exposure time information for the individual samples (readouts). The global information listed (and the names of the header keywords from which it is retrieved) includes:

- the total number of image extensions in the file (NEXTEND)
- the name of the MultiAccum exposure sample sequence (SAMP_SEQ)
- the total number of samples, including the "zeroth" read (NSAMP)
- the total exposure time of the observation (EXPTIME).

Information that is listed for each sample is the IMSET number (EXTVER), the sample number (SAMPNUM), the sample time, which is the total accumulated exposure time for a sample (SAMPTIME), and the delta time, which is the additional exposure time accumulated since the previous sample (DELTATIM).

Note that the samples of a MultiAccum exposure are stored in the FITS file in reverse time order. The initial, or "zeroth" read, appears last in the FITS file, with IMSET=NSAMP, SAMPNUM=0, SAMPTIME=0, and DELTATIM=0. The final read of the exposure appears first in the file and has IMSET=1, SAMPNUM=NSAMP-1 (SAMPNUM is zero-indexed), and SAMPTIME=EXPTIME.

2.4.1 Options

This version will accept a single image name or a python list of images. The list of images should be a python style list, such as:

```
>>> ["image1.fits","image2.fits"]
```

add_keys=list(): You can also supply a supplimental list of keywords to print for each sample, if the key isn't found in the sample the global header will be checked.If a key is not found the "NA" string will be printed. Additionally you can ask for the median or mean of the datavalues for each sample using the appropriate switch.

median=False: Set to True in order to report the median pixel value for each sample

mean=False: Set to True in order to report the mean pixel value for each sample (as measured with np.min and np.max)

2.4.2 Usage

Typical:

```
>>> python
>>> from wfc3tools import sampinfo
>>> sampinfo(imagename)
```

Where imagename can be a single filename or a python list() of names

To get the median value for each sample:

```
>>> sampinfo.sampinfo(imagename, median=True)
```

To print additional keys for information:

```
>>> sampinfo.sampinfo(imagename,add_keys=["DETECTOR"])
```

To get the average balue for each sample:

```
>>> sampinfo.sampinfo(imagename, mean=True)
```

2.4.3 Example Output

Default output:

```
In [3]: wfc3tools.sampinfo('ibcf02faq_raw.fits')
IMAGE                       NEXTEND SAMP_SEQ        NSAMP   EXPTIME
ibcf02faq_raw.fits  80      STEP50          16      499.234009

IMSET       SAMPNUM SAMPTIME        DELTATIM
1   15      499.234009      50.000412
2   14      449.233582      50.000412
```

```
3   13      399.233154      50.000412
4   12      349.232727      50.000412
5   11      299.2323        50.000412
6   10      249.231873      50.000412
7   9       199.231461      50.000412
8   8       149.231049      50.000412
9   7       99.230637       50.000412
10  6       49.230225       25.000511
11  5       24.229715       12.500551
12  4       11.729164       2.932291
13  3       8.796873        2.932291
14  2       5.864582        2.932291
15  1       2.932291        2.932291
16  0       0.0     0.0
```

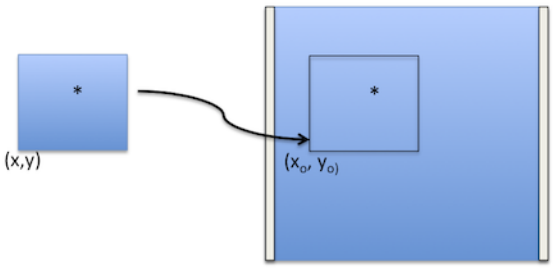
with median=True:

```
In [4]: wfc3tools.sampinfo('ibcf02faq_raw.fits',median=True)
IMAGE                       NEXTEND SAMP_SEQ        NSAMP   EXPTIME
ibcf02faq_raw.fits  80      STEP50          16      499.234009

IMSET       SAMPNUM SAMPTIME        DELTATIM
1   15      499.234009      50.000412       MedPixel: 11384.0
2   14      449.233582      50.000412       MedPixel: 11360.0
3   13      399.233154      50.000412       MedPixel: 11335.0
4   12      349.232727      50.000412       MedPixel: 11309.0
5   11      299.2323        50.000412       MedPixel: 11283.0
6   10      249.231873      50.000412       MedPixel: 11256.0
7   9       199.231461      50.000412       MedPixel: 11228.0
8   8       149.231049      50.000412       MedPixel: 11198.0
9   7       99.230637       50.000412       MedPixel: 11166.0
10  6       49.230225       25.000511       MedPixel: 11131.0
11  5       24.229715       12.500551       MedPixel: 11111.0
12  4       11.729164       2.932291        MedPixel: 11099.0
13  3       8.796873        2.932291        MedPixel: 11097.0
14  2       5.864582        2.932291        MedPixel: 11093.0
15  1       2.932291        2.932291        MedPixel: 11090.0
16  0       0.0     0.0     MedPixel: 11087.0
```

2.5 sub2full

Given an image specified by the user which contains a subarray readout, return the location of the corner of the subarray in a full frame reference image (including the full physical extent of the chip), in 1-indexed pixels. If the user supplies an X and Y coordinate, then the translated location of that point will be returned.



2.5.1 Usage

```
>>> python
>>> from wfc3tools import sub2full
>>> coords=sub2full(filename,x=None, y=None,fullExtent=False)
```

2.5.2 Parameters

- **filename [file]** Input image name or list of image names. The rootname will be used to find the _SPT files in the same directory, the SPT file has all the necessary information for the transform.
- **x = None [integer] Optional** Specify an x coordinate in the subarray to translate. If an x and y are specified, the fullExtent option is turned off and only the translated x,y coords are returned
- **y = None [integer] Optional** Specify a y coordinate in the subarray to translate. If an x and y are specified, the fullExtent option is turned off and only the translated x,y coords are returned
- **fullExtent = False [bool] Optional** If set, the returned values will include the full extent of the subarray in the reference image, for example: (x0,x1,y0,y1)

2.5.3 Returns

A list of tuples which specify the translated coordinates, either (x0,y0) for each image or the full extent sections

2.5.4 Example Output

Default output:

```
> sub2full('ibbso1fdq_flt.fits')
> [(3584.0, 1539)]
```

Optional output:

```
Specify a list of images:

>im = ['ic5p02e0q_spt.fits',
       'ic5p02e1q_spt.fits',
       'ic5p02e2q_spt.fits',
       'ic5p02e3q_spt.fits',
       'ic5p02e4q_spt.fits']
>
> sub2full(im)
> [(1062.0, 1363),
> (1062.0, 1363),
> (1410.0, 1243),
> (1410.0, 1243),
> (1402.0, 1539)]


Return the full extent of the subarray:

> sub2full('ibbso1fdq_flt.fits',fullExtent=True)
> [(3584.0, 4096, 1539, 2050)]
```

2.5.5 More information on header keywords

The task uses header keywords from the SPT file of the associated image in order to calculate the offset for the subarray. The keywords it uses are:

Keyword	Meaning
SS_DTCTR	To get the detector for the image
SS_SUBAR	To make sure the image is a subarray
XCORNER	The x corner of the subarray
YCORNER	The y corner of the subarray
NUMROWS	Subarray x size
NUMCOLS	Subarray y size

The UVIS full frame detector has 2051 rows, with 25 pixels of serial overscan. The IR detector has 1024 rows and 5 pixels of overscan.

- genindex
- modindex