wfc3tools Documentation

Release 1.0

Megan Sosey

May 03, 2013

Modules for WFC3.

http://www.stsci.edu/hst/wfc3/

Contents:

CHAPTER

ONE

CALWFC3

The wfc3tools module contains a function `calwf3` that calls the CALWF3 executable. Use this function to facilitate batch runs or for the TEAL interface.

1.1 Examples

In Python without TEAL:

```
>>> from wfc3tools import calwf3
>>> 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
```

A detailed description of this new and improved `calwf3` will be available in a future publication of WFC3 Data Handbook. 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).

1.2 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`, 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.

1.3 Where to Find calwf3

`calwf3` is now part of HSTCAL package, which can be downloaded from http://www.stsci.edu/institute/software_hardware/stsdas/download-stsdas

1.4 Usage

From the command line:

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

1.5 Command Line Options

`calwf3` supports several command line options:

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

1.6 Batch calwf3

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
import glob

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

1.7 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.

1.8 Dark Current Subtraction (DARKCORR)

It uses DARKFILE for the reference dark image.

The UVIS Dark image is now scaled by EXPTIME and FLASHDUR.

1.9 Post-Flash Correction (FLSHCORR)

Post-flash correction is now performed after DARKCORR in the WF32D step. When FLSHCORR=PERFORM, it uses FLSHFILE (the post-flash reference file).

1.10 FLATCORR

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

1.11 Photometry Keywords (PHOTCORR)

The PHOTCORR step is now performed using tables of precomputed values instead of calls to SYNPHOT. The correct table for a given image must be specified in the IMPHTTAB header keyword in order for calwf3 to perform the PHOTCORR step. By default, it should be in the `iref` directory and have the suffix `_imp.fits`. Each DETECTOR uses a different table.

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

1.12 calwf3 Output

Using RAW as input:

- flt.fits: output calibrated exposure.
- ima.fits: output ramp calibration IR exposure.

Using ASN as input with WF3REJ:

- crj.fits: cosmic ray rejected image

`wfc3tools.calwf3.`**`calwf3`**(*input*, *printtime=False*, *save_tmp=False*, *verbose=False*, *debug=False*)

calwf3 Version 1.0 updated on 03-Jan-2013

In Python without TEAL:

```
>>> from wfc3tools import calwf3
>>> 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
```

A detailed description of this new and improved `calwf3` will be available in a future publication of WFC3 Data Handbook. 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).

`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`, 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.

`calwf3` is now part of HSTCAL package, which can be downloaded from http://www.stsci.edu/institute/software_hardware/stsdas/download-stsdas

From the command line:

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

`calwf3` supports several command line options:

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

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
import glob

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

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

It uses DARKFILE for the reference dark image.

The UVIS Dark image is now scaled by EXPTIME and FLASHDUR.

Post-flash correction is now performed after DARKCORR in the WF32D step. When FLSHCORR=PERFORM, it uses FLSHFILE (the post-flash reference file).

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

The PHOTCORR step is now performed using tables of precomputed values instead of calls to SYNPHOT. The correct table for a given image must be specified in the IMPHTTAB header keyword in order for calwf3 to perform the PHOTCORR step. By default, it should be in the `iref` directory and have the suffix `_imp.fits`. Each DETECTOR uses a different table.

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

Using RAW as input:

- flt.fits: output calibrated exposure.
- ima.fits: output ramp calibration IR exposure.

Using ASN as input with WF3REJ:

- crj.fits: cosmic ray rejected image

CHAPTER

TWO

WF32D

The wfc3tools module contains a function `wf32d` that calls the WF32D executable. Use this function to facilitate batch runs or for the TEAL interface.

The wf32d primary functions include:

- dark current subtraction
- flat-fielding
- photometric keyword calculations

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.

2.1 Examples

In Python without TEAL:

```
>>> from wfc3tools import wf32d
>>> calwf3.wf32d(filename)
```

In Python with TEAL:

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

In Pyraf:

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

`wfc3tools.wf32d.`**`wf32d`**(*input*, *output=''*, *dqicorr='PERFORM'*, *darkcorr='PERFORM'*, *flatcorr='PERFORM'*, *shadcorr='PERFORM'*, *photcorr='PERFORM'*, *verbose=False*, *quiet=True*)

wf32d Version 1.0 updated on 03-Jan-2013

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

The wf32d primary functions include:

- dark current subtraction
- flat-fielding
- photometric keyword calculations

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.

In Python without TEAL:

```
>>> from wfc3tools import wf32d
>>> calwf3.wf32d(filename)
```

In Python with TEAL:

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

In Pyraf:

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

CHAPTER

THREE

WF3CCD

The wfc3tools module contains a function `wf3ccd` that calls the wf3ccd executable. Use this function to facilitate batch runs or for the TEAL interface.

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

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.

3.1 Examples

In Python without TEAL:

```
>>> from wfc3tools import wf3ccd
>>> calwf3.wf3ccd(filename)
```

In Python with TEAL:

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

In Pyraf:

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

`wfc3tools.wf3ccd.`**`wf3ccd`**(*input*, *output=''*, *dqicorr='PERFORM'*, *atodcorr='PERFORM'*, *blevcorr='PERFORM'*, *biascorr='PERFORM'*, *flashcorr='PERFORM'*, *verbose=False*, *quiet=True*)

wf3ccd Version 1.0 updated on 03-Jan-2013

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

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.

In Python without TEAL:

```
>>> from wfc3tools import wf3ccd
>>> calwf3.wf3ccd(filename)
```

In Python with TEAL:

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

In Pyraf:

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

CHAPTER

FOUR

WF3IR

The wfc3tools module contains a function `wf3ir` that calls the wf3ir executable. 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.

4.1 Examples

In Python without TEAL:

```
>>> from wfc3tools import wf3ir
>>> calwf3.wf3ir(filename)
```

In Python with TEAL:

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

In Pyraf:

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

`wfc3tools.wf3ir.`**`wf3ir`**`(`*input*, *output=''*, *verbose=False*, *quiet=True*`)`

wf3ir Version 1.0 updated on 03-Jan-2013

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.

In Python without TEAL:

```
>>> from wfc3tools import wf3ir
>>> calwf3.wf3ir(filename)
```

In Python with TEAL:

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

In Pyraf:

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

CHAPTER

FIVE

WF3REJ

The wfc3tools module contains a function `wf3rej` that calls the wf3rej executable. Use this function to facilitate batch runs or for the TEAL interface.

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

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.

5.1 Examples

In Python without TEAL:

```
>>> from wfc3tools import wf3rej
>>> calwf3.wf3rej(filename)
```

In Python with TEAL:

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

In Pyraf:

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

`wfc3tools.wf3rej.`**`wf3rej`**(*input*, *output=''*, *crrejtab=''*, *scalense=''*, *initgues=''*, *skysub=''*, *crsigmas=''*, *crradius=-1*, *crthresh=-1*, *badinpdq=-1*, *crmask=False*, *shadcorr=False*, *verbose=False*)

wf3rej Version 1.0 updated on 03-Jan-2013

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

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.

In Python without TEAL:

```
>>> from wfc3tools import wf3rej
>>> calwf3.wf3rej(filename)
```

In Python with TEAL:

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

In Pyraf:

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

CHAPTER SIX

INDICES AND TABLES

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

PYTHON MODULE INDEX

W

INDEX