

Modified: Sept 8, 2005

These are step-by-step notes for processing and reducing data for mini-mosaic images – in some cases, the details may be redundant, but they are here for completeness.

Initial data processing

IRAF

The values on the different functions are those that we found to work pretty well for mini mosaic images; however that can be modified, depending on individual preferences.

Step 1

Overscan correction

Go to ccdproc, and set:

(**images** = *imagename.fits* (or to process all the .fits files, use *.fits)
(**output** = *imagename.proc.fits* (new name of the image)

(or, to process all the files at the same time, use: *.%fits%.proc.fits)

make sure that:

(**oversca=** **yes**)
(**trim** = **yes**)
(**functio=** **spline3**) --> for cubic spline
(**order** = **5**)

(the order of the

Should look like:

I R A F

Image Reduction and Analysis Facility

PACKAGE = mscrd

TASK = ccdproc

images = *.fits List of Mosaic CCD images to proc
(output = *.%fits%proc.fits) List of output processed images
(ccdtype=) CCD image type to process
(noproc = no) List processing steps only?
(xtalkco= no) Apply crosstalk correction?
(oversca= yes) Apply overscan strip correction?
(trim = yes) Trim the image?
(fixpix = no) Apply bad pixel mask correction?
(zerocor= no) Apply zero level correction?
(darkcor= no) Apply dark count correction?
(flatcor= no) Apply flat field correction?
(sflatco= no) Apply sky flat field correction?
(merge = no) Merge amplifiers from same CCD?
(xtalkfi=) Crosstalk file
(biassec= image) Overscan strip image section
(trimsec= image) Trim data section
(fixfile=) List of bad pixel masks
(zero =) List of zero level calibration im
(dark =) List of dark count calibration im
(flat =) List of flat field images
(sflat =) List of secondary flat field imag
(minrepl= 1.) Minimum flat field value
(interac= no) Fit overscan interactively?
(functio= spline3) Fitting function
(order = 5) Number of polynomial terms or spl
(sample = *) Sample points to fit
(naverag= 1) Number of sample points to combin
(niterat= 1) Number of rejection iterations
(low_rej= 3.) Low sigma rejection factor
(high_re= 3.) High sigma rejection factor
(grow = 0.) Rejection growing radius
(fd =)
(fd2 =)
(mode = ql)

Run:

ms> ccdproc

should get (output on the screen) :

List of Mosaic CCD images to process (obj070.fits):
obj070.fits[jim1]: Aug 22 20:52 Trim is [1:1024,1:4096]
obj070.fits[jim1]: Aug 22 20:52 Overscan is [1039:1088,1:4096], mean 1685.524
obj070.fits[jim2]: Aug 22 20:52 Trim is [65:1088,1:4096]
obj070.fits[jim2]: Aug 22 20:52 Overscan is [1:50,1:4096], mean 1660.425
obj070.fits[jim3]: Aug 22 20:52 Trim is [1:1024,1:4096]
obj070.fits[jim3]: Aug 22 20:52 Overscan is [1039:1088,1:4096], mean 1574.498
obj070.fits[jim4]: Aug 22 20:52 Trim is [65:1088,1:4096]
obj070.fits[jim4]: Aug 22 20:52 Overscan is [1:50,1:4096], mean 1714.08

Step 2

examine the biases:

do an ***mscstat*** on the bias frames, and look at the *mean* (= mean of the pixel distribution). For the non-overscan corrected images, the mean should have a value of the order of 1000; for example:

```
mc> mscstat zero125.fits
```

#	IMAGE	NPIX	MEAN	STDDEV	MIN	MAX
	zero125.fits[im1]	4456448	1735.	8.892	1642.	7065.
	zero125.fits[im2]	4456448	1693.	11.39	1657.	9670.
	zero125.fits[im3]	4456448	1599.	10.43	1560.	9690.
	zero125.fits[im4]	4456448	1748.	9.941	1708.	12796.

for the overscan corrected, this value should be reduced by a factor of 3 (~10). A typical example of the result of ***mscstat*** on a corrected bias frame is:

```
mc> mscstat zero125.proc.fits
```

#	IMAGE	NPIX	MEAN	STDDEV	MIN	MAX
	zero125.proc.fits[im1]	4194304	7.074	8.878	-85.73	5338.
	zero125.proc.fits[im2]	4194304	6.918	11.52	-27.67	7983.
	zero125.proc.fits[im3]	4194304	9.021	10.42	-27.78	8099.
	zero125.proc.fits[im4]	4194304	10.27	9.828	-30.59	11058.

The IRAF help pages really enlightening when it comes to explaining the different parameters; in summary:

image = the image name

npix = the number of pixels used to do the statistics

mean = the mean of the pixel distribution

midpt = estimate of the median of the pixel distribution

mode = the mode of the pixel distribution

stddev = the standard deviation of the pixel distribution

skew = the skew of the pixel distribution

kurtosis = the kurtosis of the pixel distribution (=the peakedness or flatness of the graph of a frequency distribution especially with respect to the concentration of values near the mean as compared with the normal distribution)

min - minimum pixel value

max - maximum pixel value

It is always useful to plot a column of the processed zero image. For mini mosaic images, the ***mscexam*** task + ***l*** plots a lines of all the amplifiers from a single image:

```
ms> mscexam zero3.fits
```

Some times, there is structure on the column profile, which is good o be eliminated. In those cases, it is useful to use higher order polynomials. It is recommended to try several different orders (5, 10, 15, 20, 25, 30) to one of the bias frames, in order to determine the best one before chosing the one that is best for a specific data set. Our experience is that there is no difference between them, but, still, it is good to check.

It is always useful to check the values of individual amplifiers, and make sure that the overscan subtraction was properly performed. This can be done by doing an implot on each amplifier separately:

```
ms> implot zero030.proc.fits[1]
```

where [1] implies the first amplifier. You can zoom in the plot by using your cursor; the value should be centered around the mean value, with a +/- 10 deviation. If you want more acurate fit, you can change the value of the fitting function ("function") or the order of the polynomial ("order") on the ccdproc option file. For the function, the options are: legendre polynomial, chebyshev polynomial, linear spline, cubic spline. Feel free to adjust the values on the demands on your data; we present an example of parameters that we found to work pretty well and lead to good reduction. For more information on the different functions, use the online IRAF help file on ccdproc.

Step 3

combine bias frames from a single night for zero corection on the images: use zerocombine

make sure that:

```
(combine=      median)
(reject =      none)
(process=      no)
```

There is a choice of "average" or "median" for the "combine" option. The average just takes the average of all 5 values of the pixels, where the "median" use the average of the two central values. It is recommended to use the median (for obvious reasons...)

Your zerocombine parameter file should look like:

I R A F

Image Reduction and Analysis Facility

PACKAGE = mscred

TASK = zerocombine

input = zero03*.proc.fits List of zero level images to combine
(output = zero1) Output zero level name
(combine= median) Type of combine operation
(reject = none) Type of rejection
(ccdtype= zero) CCD image type to combine
(process= no) Process images before combining?
(delete = no) Delete input images after combining?
(scale = none) Image scaling
(statsec=) Image section for computing statistics
(nlow = 0) minmax: Number of low pixels to reject
(nhigh = 1) minmax: Number of high pixels to reject
(nkeep = 1) Minimum to keep (pos) or maximum to reject (neg)
(mclip = yes) Use median in sigma clipping algorithms?
(lsigma = 3.) Lower sigma clipping factor
(hsigma = 3.) Upper sigma clipping factor
(rdnoise= 0.) ccdclip: CCD readout noise (electrons)
(gain = 1.) ccdclip: CCD gain (electrons/DN)
(snoise = 0.) ccdclip: Sensitivity noise (fraction)
(pclip = -0.5) pclip: Percentile clipping parameter
(blank = 0.) Value if there are no pixels
(mode = ql)

typing zero*.proc.fits allows you to process a number of bias frames (overscan corrected) without typing their individual names (p.ex zero001.proc.fits, zero002.proc.fits, zero003.proc.fits etc). On the other hand, make sure to combine the bias frames of a single night; they shouldn't change from night to night, but some times they do!

Then create a master bias frame from each night:

Run:

```
ms> zerocomb
```

List of zero level images to combine (zero03*.proc.fits) and then run zerocomb. If everything works fine. you should see on your screen something like:

Aug 23 10:33: IMCOMBINE

combine = median, scale = none, zero = none, weight = none
blank = 0.

Images

zero030.proc.fits[1]
zero031.proc.fits[1]
zero032.proc.fits[1]
zero033.proc.fits[1]
zero034.proc.fits[1]

Output image = zero1[inherit], ncombine = 5

Aug 23 10:33: IMCOMBINE

combine = median, scale = none, zero = none, weight = none
blank = 0.

Images

zero030.proc.fits[2]
zero031.proc.fits[2]
zero032.proc.fits[2]
zero033.proc.fits[2]
zero034.proc.fits[2]

Output image = zero1[inherit], ncombine = 5

Aug 23 10:34: IMCOMBINE

combine = median, scale = none, zero = none, weight = none
blank = 0.

Images

zero030.proc.fits[3]
zero031.proc.fits[3]
zero032.proc.fits[3]
zero033.proc.fits[3]
zero034.proc.fits[3]

Output image = zero1[inherit], ncombine = 5

Aug 23 10:34: IMCOMBINE

combine = median, scale = none, zero = none, weight = none
blank = 0.

Images

zero030.proc.fits[4]
zero031.proc.fits[4]
zero032.proc.fits[4]
zero033.proc.fits[4]
zero034.proc.fits[4]

Output image = zero1[inherit], ncombine = 5

Step 4

correct the flat fields: use ccdproc and the same master bias from the same night, in order to bias correct the flat field images. The ccdproc option file should look like:

```

                I R A F
            Image Reduction and Analysis Facility
PACKAGE = mscrd
    TASK = ccdproc

images =   dflat*.proc.fits List of Mosaic CCD images to process
(output = dflat*.%proc.fits%bias.fits) List of output processed images
(ccdtype=          ) CCD image type to process
(noproc =          no) List processing steps only?

(xtalkco=          no) Apply crosstalk correction?
(oversca=          no) Apply overscan strip correction?
(trim  =           no) Trim the image?
(fixpix =          no) Apply bad pixel mask correction?
(zerocor=          yes) Apply zero level correction?
(darkcor=          no) Apply dark count correction?
(flatcor=          no) Apply flat field correction?
(sflatco=          no) Apply sky flat field correction?
(merge =           no) Merge amplifiers from same CCD?

(xtalkfi=          ) Crosstalk file
(biassec=          image) Overscan strip image section
(trimsec=          image) Trim data section
(fixfile=          ) List of bad pixel masks
(zero  =           zero.fits) List of zero level calibration images
(dark  =           ) List of dark count calibration images
(flat  =           ) List of flat field images
(sflat =           ) List of secondary flat field images
(minrepl=          1.) Minimum flat field value

(interac=          no) Fit overscan interactively?
(funcio=          spline3) Fitting function
(order  =          40) Number of polynomial terms or spline pieces
(sample =          *) Sample points to fit
(naverag=          1) Number of sample points to combine
(niterat=          1) Number of rejection iterations
(low_rej=          3.) Low sigma rejection factor
(high_re=          3.) High sigma rejection factor
(grow  =           0.) Rejection growing radius
(fd    =           )
(fd2   =           )
(mode  =           ql)
```

and the output on the screen:

```
dflat086.proc.fits[im1]: Aug 26 14:00 Zero is zero.fits[im1]
dflat086.proc.fits[im2]: Aug 26 14:00 Zero is zero.fits[im2]
dflat086.proc.fits[im3]: Aug 26 14:00 Zero is zero.fits[im3]
dflat086.proc.fits[im4]: Aug 26 14:00 Zero is zero.fits[im4]
dflat087.proc.fits[im1]: Aug 26 14:00 Zero is zero.fits[im1]
dflat087.proc.fits[im2]: Aug 26 14:00 Zero is zero.fits[im2]
dflat087.proc.fits[im3]: Aug 26 14:00 Zero is zero.fits[im3]
dflat087.proc.fits[im4]: Aug 26 14:00 Zero is zero.fits[im4]
dflat088.proc.fits[im1]: Aug 26 14:00 Zero is zero.fits[im1]
dflat088.proc.fits[im2]: Aug 26 14:01 Zero is zero.fits[im2]
dflat088.proc.fits[im3]: Aug 26 14:01 Zero is zero.fits[im3]
dflat088.proc.fits[im4]: Aug 26 14:01 Zero is zero.fits[im4]
dflat089.proc.fits[im1]: Aug 26 14:01 Zero is zero.fits[im1]
dflat089.proc.fits[im2]: Aug 26 14:01 Zero is zero.fits[im2]
dflat089.proc.fits[im3]: Aug 26 14:01 Zero is zero.fits[im3]
dflat089.proc.fits[im4]: Aug 26 14:01 Zero is zero.fits[im4]
```

The results are annotated by dflat#.bias.fits

examine the flat fields from the same night:

It is good to make sure that the mean of all the flat fields for an individual amplifier in a specific filter are the same. This is going to be used in the combination of the flat field images in order to create a "master" flat field. It is also good to keep the results on a file. Therefore, do an

```
ms> mscstat dflat*.bias.fits > dflatstat
```

dflatstat is the file created, containing the statistics-results of the ccdproc process. You should check the dflatstat file to make sure that there are not big deviations between the mean values of the same amplifier of the different flat field exposures (same filter, same night). If there ARE big differences, then a "median" SHOULD be used for the flatcombine procedure. If the differences are not that then an average can be used. In general, though, it is recommended that a median is used for the combining.

File: dflatstat

#	IMAGE	NPIX	MEAN	STDDEV	MIN	MAX
	dflat086.bias.fits[jim1]	4194304	23607.	1144.	-26.01	31682.
	dflat086.bias.fits[jim2]	4194304	22427.	609.7	510.6	63923.
	dflat086.bias.fits[jim3]	4194304	20666.	763.5	-25.86	33067.
	dflat086.bias.fits[jim4]	4194304	21203.	302.9	2277.	61353.
	dflat087.bias.fits[jim1]	4194304	23313.	1130.	-29.44	30729.
	dflat087.bias.fits[jim2]	4194304	22142.	416.8	481.7	27569.
	dflat087.bias.fits[jim3]	4194304	20409.	752.7	-23.32	32572.
	dflat087.bias.fits[jim4]	4194304	20940.	298.7	2251.	61351.
	dflat088.bias.fits[jim1]	4194304	23033.	1116.	-30.39	28488.
	dflat088.bias.fits[jim2]	4194304	21881.	412.	492.5	54695.
	dflat088.bias.fits[jim3]	4194304	20164.	742.4	-24.76	32488.
	dflat088.bias.fits[jim4]	4194304	20688.	295.4	2216.	61350.
	dflat089.bias.fits[jim1]	4194304	23018.	1117.	-34.42	63769.
	dflat089.bias.fits[jim2]	4194304	21869.	411.3	474.2	29170.
	dflat089.bias.fits[jim3]	4194304	20154.	741.9	-30.91	32440.
	dflat089.bias.fits[jim4]	4194304	20677.	294.8	2124.	61349.
	dflat090.bias.fits[jim1]	4194304	23057.	1117.	-31.3	28850.
	dflat090.bias.fits[jim2]	4194304	21901.	412.1	479.2	28284.
	dflat090.bias.fits[jim3]	4194304	20188.	743.4	-29.3	32529.
	dflat090.bias.fits[jim4]	4194304	20713.	295.2	2189.	61348.

.
.
From our experience, there is a slight increase of the pixel value between successive exposures of the same filter for the same amplifier. For example, from the list above, notice that in:

#	IMAGE	NPIX	MEAN	STDDEV	MIN	MAX
	dflat086.bias.fits[jim1]	4194304	23607.	1144.	-26.01	31682.
	dflat087.bias.fits[jim1]	4194304	23313.	1130.	-29.44	30729.
	dflat088.bias.fits[jim1]	4194304	23033.	1116.	-30.39	28488.
	dflat089.bias.fits[jim1]	4194304	23018.	1117.	-34.42	63769.
	dflat090.bias.fits[jim1]	4194304	23057.	1117.	-31.3	28850.

the mean value decreases from 23607. (dflat086.bias.fits) to 23057. (dflat090.bias.fits) for amp1. To take this into account in the data reduction procedure, it is good to do a scaling when combining the flat fields (look next step).

Step 5

Combine the flat field images of the same night/filter: flatcombine

make sure that:

combine = median
reject = none
scale = mode

The options for the scale are
none|mode|median|mean|exposure

A "mode" can correct for large, time dependent changes of the intensity of the lamps. The options for the pixel rejection (option "reject") is
none|minmax|ccdclip|crreject|sigclip|avsigclip|pclip.

For more info on the functions themselves, look the IRAF help manual. In general, it is good to use a "median" function, with "none" at the reject options.

the flatcombine file should look like:

```

                I R A F
          Image Reduction and Analysis Facility
PACKAGE = mscrd
  TASK = flatcombine

input =    dflat*.mb.fits List of flat field images to combine
(output =    Flat) Output flat field root name
(combine=    median) Type of combine operation
(reject =    none) Type of rejection
(ccdtype=    ) CCD image type to combine
(process=    no) Process images before combining?
(subsets=    yes) Combine images by subset parameter?
(delete =    no) Delete input images after combining?
(scale =    mode) Image scaling
(statsec=    ) Image section for computing statistics
(nlow =    1) minmax: Number of low pixels to reject
(nhigh =    1) minmax: Number of high pixels to reject
(nkeep =    1) Minimum to keep (pos) or maximum to reject (neg)
(mclip =    yes) Use median in sigma clipping algorithms?
(lsigma =    3.) Lower sigma clipping factor
(hsigma =    3.) Upper sigma clipping factor
(rdnoise=    0.) ccdclip: CCD readout noise (electrons)
(gain =    1.) ccdclip: CCD gain (electrons/DN)
(snoise =    0.) ccdclip: Sensitivity noise (fraction)
(pclip =    -0.5) pclip: Percentile clipping parameter
(blank =    1.) Value if there are no pixels
(mode =    ql)
```

Step 6

process the images: flat field and zero correction on the images, by using

ccdproc:

(output: obj031.correct.fits)

Step 7

correct the bad columns: Mini mo has several bad columns on each amplifiers; it is good to correct them. There are 4 different mask.dat files, one for each of the amplifiers, for mini-mo. For some purposes, it is advisable to create single pixel masks, but they are not necessary in general. The correction takes place at one amplifier at a time, therefore it is good to have the amplifiers split

Step 7

correct for ghost images:

The mini mosaic images have ghost images due to saturated stars. The ghost signature of a bright saturated star is very profound as mirror signals on the adjacent amplifiers. On the other hand, when a small number of pixels is saturated, the ghost images can be hidden from the eye, but still be there, and may overlap with stars. This way, stars can appear to vary for image to image, and appear as variable stars, which can be misleading in variability searches. The unfortunate is that the images can not be corrected, which may have some impact to the science, but they can be masked so that they will be excluded from the science. IDL routines do exist for the cosmetic improvement of an image (create "pretty pictures"), but this can not be used for scientific purposes. There is a way on IRAF to mask the ghost images from each amplifier. This is done by using the IRAF imexpr task. The command syntax is as follows:

```
imexpr "(a > 65534 | b> 65534 | c>65534 ? 0 : d)" newimage obj080[1][-*,*]
obj080[2] obj080[3][-*,*] obj080[4]
```

```
ms> imexpr
expression ((a > 65535 | b> 65535 | c>65535 ? -32768 : d)): (a > 65534 |
b> 65534 | c>65534 ? -32768 : d)
output image (junk): junk1
operand a (obj080[1][-*,*]):
operand b (obj080[2]):
operand c (obj080[3][-*,*]):
operand d (obj080[4]):
10% 20% 30% 40% 50% 60% 70% 80% 90% 100% - done
```

```
ms> msdisplay junk1 3
Amp:      Individual      Display (zcombine=minmax)
1:      2394.9 2837.1      2394.9 2837.1
```

Image Reduction and Analysis Facility

PACKAGE = imutil

TASK = imexpr

expr = (a > 65534 | b > 65534 | c > 65534 ? -32768 : d) expression

output = junk_all[im2][append+] output image

(dims = auto) output image dimensions

(intype = auto) minimum type for input operands

(outtype= auto) output image pixel datatype

(refim = auto) reference image for wcs etc

(bwidth = 0) boundary extension width

(btype = nearest) boundary extension type

(bpixval= 0.) boundary pixel value

(rangech= yes) perform range checking

(verbose= yes) print informative messages

(exprdb = none) expression database

(lastout= junk_all[im2]) last output image

a = obj080[3][-*,*] operand a

b = obj080[4] operand b

c = obj080[1][-*,*] operand c

d = obj080[2] operand d

e = operand e

f = operand f

g = operand g

h = operand h

i = operand i

j = operand j

k = operand k

l = operand l

m = operand m

n = operand n

o = operand o

p = operand p

q = operand q

r = operand r

s = operand s

t = operand t

u = operand u

v = operand v

w = operand w

x = operand x

y = operand y

z = operand z

(mode = ql)

Set 1: corrected output = amp1

output = obj116.gh_1.fits[1][append+] output image

- a = obj116.correct.fits[2][-*,*] operand a
- b = obj116.correct.fits[3] operand b
- c = obj116.correct.fits[4][-*,*] operand c
- d = obj116.correct.fits[1] operand d

Set 2: corrected output = amp2

output = obj116.gh_2.fits[2][append+] output image

- a = obj116.correct.fits[3][-*,*] operand a
- b = obj116.correct.fits[4] operand b
- c = obj116.correct.fits[1][-*,*] operand c
- d = obj116.correct.fits[2] operand d

Set 3: corrected output = amp3

output = obj116.gh_3.fits[3][append+] output image

- a = obj116.correct.fits[4][-*,*] operand a
- b = obj116.correct.fits[1] operand b
- c = obj116.correct.fits[2][-*,*] operand c
- d = obj116.correct.fits[3] operand d

Set 4: corrected output = amp4

output = obj116.gh_4.fits[4][append+] output image

- a = obj116.correct.fits[1][-*,*] operand a
- b = obj116.correct.fits[2] operand b
- c = obj116.correct.fits[3][-*,*] operand c
- d = obj116.correct.fits[4] operand d

or:

for amplifier 1:

```
imexpr "(a > 65534 | b > 65534 | c > 65534 ? 0 : d)" obj116_1.gh.fits  
obj116.correct.fits[2][-*,*] obj116.correct.fits[3] obj116.correct.fits[4][-*,*]  
obj116.correct.fits[1]
```

for amplifier 2:

```
imexpr "(a > 65534 | b> 65534 | c>65534 ? 0 : d)" obj116_2.gh.fits
obj116.correct.fits[3][-*,*] obj116.correct.fits[4] obj116.correct.fits[1][-*,*]
obj116.correct.fits[2]
```

for amplifier 3:

```
imexpr "(a > 65534 | b> 65534 | c>65534 ? 0 : d)" obj116_3.gh.fits
obj116.correct.fits[4][-*,*] obj116.correct.fits[1] obj116.correct.fits[2][-*,*]
obj116.correct.fits[3]
```

for amplifier 4:

```
imexpr "(a > 65534 | b> 65534 | c>65534 ? 0 : d)" obj116_4.gh.fits
obj116.correct.fits[1][-*,*] obj116.correct.fits[2] obj116.correct.fits[3][-*,*]
obj116.correct.fits[4]
```

As an output, individual images of the different amplifiers will be created, in each one of which the ghost images will be masked.

Step 7

correct the bad columns: Mini mo has several bad columns on each amplifiers; it is good to correct them. There are 4 different mask.dat files, one for each of the amplifiers, for mini-mo. For some purposes, it is advisable to create single pixel masks (to correct for small pixel defects, but they are not necessary in general. The individual amplifiers can be corrected for bad columns, using the appropriate mask.dat files for each amplifier on ccdproc:

for amp1 (mask1.dat):

```
200 201 3468 4096
319 320 1108 4096
443 444 1451 4096
444 445 1451 4096
214 216 680 787
```

for amp2 (mask2.dat):

```
941 946 2385 4096
```

for amp3 (mask3.dat)

912 913 1605 4096
932 933 692 4096
536 540 3554 4096

for amp4 (mask4.dat)

545 560 214 4096

ccdproc option file:

```

                I R A F
          Image Reduction and Analysis Facility
PACKAGE = mscred
  TASK = ccdproc

images =  obj116_2.gh.fits List of Mosaic CCD images to process
(output = obj116.mask_2.fits) List of output processed images
(ccdtype=          ) CCD image type to process
(noproc =          no) List processing steps only?
(xtalkco=         no) Apply crosstalk correction?
(oversca=         no) Apply overscan strip correction?
(trim  =          no) Trim the image?
(fixpix =         yes) Apply bad pixel mask correction?
(zerocor=         no) Apply zero level correction?
(darkcor=         no) Apply dark count correction?
(flatcor=         no) Apply flat field correction?
(sflatco=        no) Apply sky flat field correction?
(merge  =         no) Merge amplifiers from same CCD?
(xtalkfi=         ) Crosstalk file
(biassec=         image) Overscan strip image section
(trimsec=         image) Trim data section
(fixfile=         mask2.dat) List of bad pixel masks
(zero   =         zero.fits) List of zero level calibration images
(dark   =         ) List of dark count calibration images
(flat   = BflatB_Harris.fits) List of flat field images
(sflat  =         ) List of secondary flat field images
(minrepl=         1.) Minimum flat field value
(interac=         no) Fit overscan interactively?
(funcio=         spline3) Fitting function
(order  =         5) Number of polynomial terms or spline piec
(sample =         *) Sample points to fit
(naverag=         1) Number of sample points to combine
(niterat=         1) Number of rejection iterations
(low_rej=         3.) Low sigma rejection factor
(high_re=         3.) High sigma rejection factor
(grow   =         0.) Rejection growing radius
(fd     =         )
(fd2    =         )
(mode   =         ql)
```

make sure to have the appropriate mask.dat file on the "fixfile" option, and set the "fixpix" option to "yes" whereas the rest of the options should be sent into "no"

In some cases, it is useful to work with individual amplifiers; therefore our initial data processing end here, and the images are ready for photometry. However, some cases require the whole image (all 4 amplifiers) to be used. For this reason, we need to combine the individual amplifiers into one. This can be done by using the imcombine task. The parameter file looks like:

```

          I R A F
    Image Reduction and Analysis Facility
PACKAGE = immatch
  TASK = imcombine

input = junk116_1.mask,junk116_2.mask List of images to combine
output =      outjunk List of output images
(rejmask=      ) List of rejection masks (optional)
(plfile =      ) List of pixel list files (optional)
(sigma =      ) List of sigma images (optional)
(logfile=      STDOUT) Log file

(combine=      average) Type of combine operation
(reject =      none) Type of rejection
(project=      no) Project highest dimension of input images?
(outtype=      real) Output image pixel datatype
(offsets=      offset) Input image offsets
(masktyp=      none) Mask type
(maskval=      0.) Mask value
(blank =      1.) Value if there are no pixels

(scale =      none) Image scaling
(zero =      none) Image zero point offset
(weight =      none) Image weights
(statsec=      ) Image section for computing statistics
(expname=      ) Image header exposure time keyword

(lthresh=      INDEF) Lower threshold
(hthresh=      INDEF) Upper threshold
(nlow =      1) minmax: Number of low pixels to reject
(nhigh =      1) minmax: Number of high pixels to reject
(nkeep =      1) Minimum to keep (pos) or maximum to reject
(mclip =      yes) Use median in sigma clipping algorithms?
(lsigma =      3.) Lower sigma clipping factor
(hsigma =      3.) Upper sigma clipping factor
(rdnoise=      0.) ccdclip: CCD readout noise (electrons)
(gain =      1.) ccdclip: CCD gain (electrons/DN)
(snoise =      0.) ccdclip: Sensitivity noise (fraction)
(sigscal=      0.1) Tolerance for sigma clipping scaling corre
(pclip =      -0.5) pclip: Percentile clipping parameter
(grow =      0.) Radius (pixels) for neighbor rejection
(mode =      ql)

```

or, just type

```
ms> imcomb Bshort2_3.mask.fits,Bshort2_4.mask.fits Bshort2.chip2.fits
```

junk116_1.mask & junk116_2.mask are amplifiers 1 and 2 of the same image, and imout is the output image that contains the two amplifiers combined. Make sure that:

(combine= average) Type of combine operation
(reject = none) Type of rejection
(project= no) Project highest dimension of input image
(outtype= real) Output image pixel datatype
(blank = 1.) Value if there are no pixels
(scale = none) Image scaling
(zero = none) Image zero point offset
(weight = none) Image weights

The "offset" file in the "offsets" option is a file containing the offsets of one of the images with respect to the others. Each line of the file corresponds to one image, and each column corresponds to a dimension (x or y). For example, the offset file use above is the following:

```
0 0
1024 0
```

This means that for the first image (which, in this case, is junk116_1) there is no offset on x and y. For the second image (junk116_2) there is an offset of 1024 on the x dimension and 0 on the y. This is necessary for the final image to have dimensions of 2048 x 4096 (two amplifiers together).
