

Data Reductions with OPTIC at WIYN (Getting the data into the proper format)

June 22, 2005

Questions/Comments to Heidi Schweiker (heidis@wiyn.org) or Steve Howell (showell@wiyn.org).

OPTIC has 2 CCD's with 2 amplifiers each, but when readout the image is a "flat" format as opposed to a multiextension FITS format (like MiniMo and Mosaic). It's relatively simple to convert the image into MEF format using the task **mkmsc** within the **mscred** package. Once converted, reductions are similar to MiniMo or Mosaic. Note that additional steps need to be done if OT guiding was used.

Step 1 – getting the *.fits* extension

The OPTIC images are saved as FITS files at the telescope, but do not have the *.fits* extension. You'll need to copy the original files so that they have the *.fits* extension in order to be able to work with them in IRAF. You can do this manually:

```
cl> imcopy a.001 a001.fits
```

Or you can use the perl script **appendfits.pl** (available off the WIYN web site) to rename all of your optic images at once. If you are using this script on MrBill please first copy it to another file and edit that as opposed to editing the original **appendfits.pl** file. If using this script on your computer you will need to do a couple things to make the script run:

1. Change the top line to point to perl on your system.
2. `chmod u+x` to make it executable.

In either case you will need to make a list of the filenames that you would like to change:

3. make a list of filenames to change:

```
ls x.??? > infiles
```

4. run the script: `appendfits.pl infiles`

The script makes a copy of all input files with a *.fits* suffix. You can change the line

```
system ("cp $infile ${infile}.fits");
```

to

```
system ("mv $infile ${infile}.fits");
```

so that you rename the file rather than copy it.

Step 2 – getting the correct information into your headers

There are important keywords that may be missing or incorrect in the OPTIC fits headers, such as 'obstype', 'object', 'filter', 'telfilter'. It's easiest to edit the headers using hedit while the images are still flat (i.e. before running mkmsc), otherwise you'll need to edit all 5 extensions [0-4] (extension 0 is a short header that has information common to the 4 extensions). The keywords 'obstype' or 'object' may not exist in your headers (unless you explicitly set these at the telescope) so when using hedit be sure to edit the hedit parameter file to set the "add" and "addonly" parameters to yes as show here:

```
cl> epar hedit
```

```
PACKAGE = imutil
TASK = hedit
```

```
images =          images to be edited
fields =          fields to be edited
value =          zero value expression
add =            yes) add rather than edit fields
addonly=         yes) add only if field does not exist
delete =         no) delete rather than edit fields
verify =         yes) verify each edit operation
show =           yes) print record of each edit operation
update =         yes) enable updating of the image header
mode =           ql)
```

Now that the hedit parameters are set correctly you can run hedit on your images:

```
cl> hedit @zeros.list obstype zero verify+ (where zeros.list is a list of all bias frames)
cl> hedit @flats.list obstype flat verify+ (where flats.list is a list of all flats)
cl> hedit @objects.list obstype object verify+ (where objects.list is a list of all object frames)
```

It's not necessary to change the 'object' keyword value in your headers, but if you'd like to change it you can do so in the same manner (using lists of files) or individually. For example:

```
cl> hedit a001.fits object M42 verify+
```

At WIYN, the correct filter name in an OPTIC header is stored in the TELFILT header keyword. There is an additional FILTER keyword, which never updates. The mscred package uses the FILTER keyword when processing images so you may want to use hedit again to edit this keyword in all of your data to note the correct filter. Though it's not used, you may want to also check the TELFILT keyword to be sure the correct filter was written here.

cl> epar hedit

PACKAGE = imutil

TASK = hedit

images =	a001.fits	images to be edited
fields =	FILTER	fields to be edited
value =	V	value expression
add =	no)	add rather than edit fields
addonly=	yes)	add only if field does not exist
delete =	no)	delete rather than edit fields
verify =	yes)	verify each edit operation
show =	yes)	print record of each edit operation
update =	yes)	enable updating of the image header
mode =	ql)	

The GAIN and READNOISE keywords for each amp are also not populated correctly in the headers, however the *mkmsc* task below will populate the headers with the correct values for GAIN and READNOISE so we do not need to address that now.

Step 3 – Correcting the Bias frames

A. Getting your images into Multi-Extension Fits (MEF) format

Your images should now have all of the necessary information in the headers. The next step is to convert these "flat" images into MEF format. To do this you need to load the `mscred` package in IRAF. Use the `setinstrument` task to set all of the correct parameters in the necessary tasks. This is also where you will note the location of the database file which contains the information needed to convert your images to MEF format.

```
cl> setinst
```

```
Site (? for menu): ?
```

```
Sites:
```

```
kpno Kitt Peak National Observatory  
ctio Cerro Tololo Inter-American Observatory
```

```
Site (? for menu or q to quit): kpno
```

```
Telescope (? for menu): ?
```

```
Telescopes:
```

```
36inch 36inch/0.9m telescope  
4meter Mayall 4meter telescope  
wiyw WIYN telescope
```

```
Telescope (? for menu or q to quit): wiyw
```

```
Instrument (? for a list): ?
```

```
Instruments:
```

```
minimosaic WIYN MiniMosaic  
wttm WIYN Tip-Tilt  
optic1 OPTIC full format (no guide regions)
```

```
Instrument (? for menu or q to quit): optic1
```

There is a 104 pixel gap between the 2 chips in OPTIC. To have this gap appear when displaying your MEF image you need to specifically set the gap size in the parameter file **mimpars**; `xgap=104, ygap=0`. Edit **mimpars** to reflect this.

```
cl> epar mimpars
```

```
PACKAGE = mscred
TASK = mimpars
```

```
(extname = "")           extension name pattern
(exttmplt = "_![1-9]*.*") extension template for separate images\n
(xgap = 104)            minimum X gap between images
(ygap = 0)              minimum Y gap between images\n
(process = no)          do calibration processing?
(overscan = yes)        do line-by-line overscan subtraction?
(flatfield = yes)       do flat field correction?
(caldir = "mscdb$noao/kpno/wiyn/caldir/") calibration directory
(filter = "!filter")    filter
(mode = "ql")
```

The next step is to use the task **mkmsc** to divide each image into extensions [0-4]. Edit the parameters for the **mkmsc** task:

```
cl> epar mkmsc
```

```
PACKAGE = mscred
TASK = mkmsc
```

```
input = @zeros.list List of input images
output = @msczeros.list List of output mosaic MEF files
(descrip= mscdb$noao/kpno/wiyn/Optic/optic1.mkmsc) Description file
(verbose= yes) Verbose input?
(mode= ql)
```

Zeros.list is just a list of the original bias frames with the .fits extension. To distinguish the MEF zeros from the original zeros you can use whatever naming convention you like. The msczeros.list file is just a list of the MEF images you will be creating.

This mkmsc task will use the optic1.mkmsc file to add specific keywords to the header of each extension. Below is the optic1.mkmsc file within the mscred database:

MKMSC description file for a basic OPTIC exposure.

```
imageid(ccd0) 1
datasec(ccd0) [1:2048,1:2052]
trimsec(ccd0) [1:2048,1:2052]
biassec(ccd0) [4097:4128,1:2052]
ccdsec(ccd0) [1:2048,1:2052]
detsec(ccd0) [1:2048,1:2052]
ccdname(ccd0) "CCD1"
ampname(ccd0) "AMP1"
rdnoise(ccd0) 4.
gain(ccd0) !CCD0GAIN
optfilt(ccd0) !FILTER
filter(ccd0) !TELFILT
dtv1(ccd0) 0.
dtv2(ccd0) 0.
dtm1_1(ccd0) 1.
dtm2_2(ccd0) 1.

imageid(ccd1) 2
datasec(ccd1) [1:2048,2053:4104]
trimsec(ccd1) [1:2048,2053:4104]
biassec(ccd1) [4097:4128,2053:4104]
ccdsec(ccd1) [1:2048,2053:4104]
detsec(ccd1) [1:2048,2053:4104]
ccdname(ccd1) "CCD1"
ampname(ccd1) "AMP2"
rdnoise(ccd1) 4.
gain(ccd1) !CCD1GAIN
optfilt(ccd1) !FILTER
filter(ccd1) !TELFILT
dtv1(ccd1) 0.
dtv2(ccd1) 0.
dtm1_1(ccd1) 1.
dtm2_2(ccd1) 1.

imageid(ccd2) 3
datasec(ccd2) [2049:4096,1:2052]
trimsec(ccd2) [2049:4096,1:2052]
biassec(ccd2) [4129:4160,1:2052]
ccdsec(ccd2) [1:2048,1:2052]
detsec(ccd2) [2049:4096,1:2052]
ccdname(ccd2) "CCD2"
ampname(ccd2) "AMP1"
rdnoise(ccd2) 4.
gain(ccd2) !CCD2GAIN
optfilt(ccd2) !FILTER
filter(ccd2) !TELFILT
dtv1(ccd2) 2048.
dtv2(ccd2) 0.
dtm1_1(ccd2) 1.
dtm2_2(ccd2) 1.
```

```

imageid(ccd3)    4
datasec(ccd3)   [2049:4096,2053:4104]
trimsec(ccd3)   [2049:4096,2053:4104]
biassec(ccd3)   [4129:4160,2053:4104]
ccdsec(ccd3)    [1:2048,2053:4104]
detsec(ccd3)    [2049:4096,2053:4104]
ccdname(ccd3)   "CCD2"
ampname(ccd3)   "AMP2"
rdnoise(ccd3)   4.
gain(ccd3)      !CCD3GAIN
optfilt(ccd3)   !FILTER
filter(ccd3)    !TELFILT
dtv1(ccd3)      2048.
dtv2(ccd3)      0.
dtm1_1(ccd3)    1.
dtm2_2(ccd3)    1.

```

There are some bad regions in the overscan. It's a good idea to edit the biassec section of this optic1.mkmsc file before running the mkmsc task. First copy it to your local directory.

```
cl> cp mscdb$noao/kpno/wiyn/Optic/optic1.mkmsc optic1_edit.mkmsc
```

You should check your own images, but an example of good regions is:

```

biassec(ccd0) [4099:4123,1:2052]
biassec(ccd1) [4099:4114,2053:4104]
biassec(ccd2) [4141:4157,1:2052]
biassec(ccd3) [4130:4145,2053:4104]

```

If you do edit the bias sections, be sure to update the mkmsc parameter to note the location of your edited file.

```
cl> epar mkmsc
```

```

PACKAGE = mscred
TASK = mkmsc

input = @zeros.list List of input images
output = @msczeros.list List of output mosaic MEF files
(descrip= optic1_edit.mkmsc) Description file
(verbose= yes) Verbose input?
(mode= ql)

```

Now you can run mkmsc on your zeros. After doing so, be sure to check the headers to make sure correct information was populated to all extensions.

B. Trimming the biases

You will then need to use `ccdproc` to trim the zeros – turn off all corrections except trim.

```
cl> epar ccdproc
```

```
    images = @zeros.list      List of Mosaic CCD images to process
    (output = " ")           List of output processed images
    (bpmmasks = " ")         List of output bad pixel masks
    (ccdtype = " ")          CCD image type to process
    (noproc = no)             List processing steps only?\n
    (xtalkcor = no)           Apply crosstalk correction?
    (fixpix = no)             Apply bad pixel mask correction?
    (overscan = no)          Apply overscan strip correction?
    (trim = yes)              Trim the image?
    (zerocor = no)           Apply zero level correction?
    (darkcor = no)           Apply dark count correction?
    (flatcor = no)           Apply flat field correction?
    (sflatcor = no)          Apply sky flat field correction?
    (split = no)             Use split images during processing?
    (merge = no)             Merge amplifiers from same CCD?\n
    (xtalkfile = " ")         Crosstalk file
    (fixfile = "BPM")         List of bad pixel masks
    (saturation = "INDEF")    Saturated pixel threshold
    (sgrow = 0)               Saturated pixel grow radius
    (bleed = "INDEF")         Bleed pixel threshold
    (btrail = 20)            Bleed trail minimum length
    (bgrow = 0)               Bleed pixel grow radius
    (biassec = "!biassec")    Overscan strip image section
    (trimsec = "!trimsec")    Trim data section
    (zero = " ")              List of zero level calibration images
    (dark = "Dark")           List of dark count calibration images
    (flat = " ")              List of flat field images
    (sflat = "Sflat*")        List of secondary flat field images
    (minreplace = 1.)         Minimum flat field value\n
    (interactive = no)        Fit overscan interactively?
    (function = "legendre")    Fitting function
    (order = 1)               Number of polynomial terms or spline
    (sample = "**")            Sample points to fit
    (naverage = 1)            Number of sample points to combine
    (niterate = 1)            Number of rejection iterations
    (low_reject = 3.)         Low sigma rejection factor
    (high_reject = 3.)        High sigma rejection factor
    (grow = 0.)               Rejection growing radius
    (fd = "")                 
    (fd2 = "")                 
    (mode = "ql")
```

C. Zerocombine

You can now use zerocombine to combine all of your bias frames into one master zero.

cl> epar zerocombine

```
    input = @msczeros.list  List of zero level images to combine
  (output = "Zero")        Output zero level name
  (combine = "median")     Type of combine operation
  (reject = "crreject")    Type of rejection
  (ccdtype = " ")          CCD image type to combine
  (process = yes)          Process images before combining?
  (delete = no)            Delete input images after combining?
  (scale = "none")         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
  (mclip = yes)            Use median in sigma clipping algorithm
  (lsigma = 3.)            Lower sigma clipping factor
  (hsigma = 3.)            Upper sigma clipping factor
  (rdnoise = "rdnoise")    ccdclip: CCD readout noise (electrons)
  (gain = "gain")          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 = "q1")
```

Step 4 – Correcting the Flats

Before combining flats, ideally you should run `fixpix` on the flats first.

A. Making Bad Pixel Masks

There are currently no bad pixel masks on file for OPTIC. However it's not hard to make your own.

Ideally you'd like to have 2 flatfield images in the same filter, each with different counts. You would then use the task **imarith** and **ccdmask** to produce the (4) bad pixel masks – one for each section. The syntax for **imarith** is:

```
cl> imarith flatlow[1] /flathigh[1] ratio1
```

This will use 2 flat field images (flatlow with significantly lower counts than flathigh) and produce the image `ratio1.fits`. You will then use this `ratio1.fits` file along with the task `ccdmask` to produce the bad pixel mask for extension 1. You'll need to load the `imred` and `ccdred` packages to get to the **ccdmask** task. The syntax is:

```
cl> ccdmask ratio1 mask1
```

This will produce a `mask1.pl` file which you will use as your bad pixel mask. However, most likely it is not perfect. You will want to display the bad pixel mask over at least one of your flatfield images to see if there are additional pixels that need to be added to the mask or areas that need to be removed from the mask. To display a mask over an image:

```
cl> disp flathigh[1] 1 bpmask=mask1.pl bpdisplay=overlay
```

If you do find regions that need to be added to or subtracted from your mask you can do so using **imreplace**.

```
cl> imreplace mask1.pl[1:2,1:2052] value=1
```

This will add columns 1 and 2 to the mask. A `value=0` will subtract.

Keep in mind, you will most likely need to do additional editing of your masks for your conflat'd flats.

Once you are satisfied with your mask you will need to use **hedit** again to note the name of your bad pixel mask in your header. To do this:

```
cl> hedit a*.fits[1] BPM mask1.pl verify+
```

Now you are ready to combine all of the flats in each filter.

If you used OT guiding you will need to run the task `conflat` on your combined flatfields. The `conflat` task is expecting images of type `ushort`, `flatcombine` will not allow you to set this parameter, so you need to use `imcombine` instead (if no OT guiding was used, feel free to use the task `flatcombine`). Be sure to set the parameter `outtype` to `ushort`:

cl> epar imcombine

input = "@Iflats1"	List of images to combine
output = "IFlat"	List of output images
(headers = "")	List of header files (optional)
(bpmsk = "mask")	List of bad pixel masks (optional)
(rejmsk = "")	List of rejection masks (optional)
(nrejmsk = "")	List of number rejected masks
(expmsk = "")	List of exposure masks (optional)
(sigmas = "")	List of sigma images (optional)
(logfile = "STDOUT")	Log file\n
(combine = "average")	Type of combine operation
(reject = "ccdclip")	Type of rejection
(project = no)	Project highest dimension of input
(outtype = "ushort")	Output image pixel datatype
(outlimits = "")	Output limits (x1 x2 y1 y2 ...)
(offsets = "none")	Input image offsets
(masktype = "none")	Mask type
(maskvalue = 0.)	Mask value
(blank = 1.)	Value if there are no pixels\n
(scale = "mode")	Image scaling
(zero = "none")	Image zero point offset
(weight = "none")	Image weights
(statsec = "")	Image section for computing statistics
(expname = "")	Image header exposure time keyword\n
(lthreshold = INDEF)	Lower threshold
(hthreshold = 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
(mclip = no)	Use median in sigma clipping algorithms
(lsigma = 3.)	Lower sigma clipping factor
(hsigma = 3.)	Upper sigma clipping factor
(rdnoise = "4")	ccdclip: CCD readout noise (electrons)
(gain = "1.4")	ccdclip: CCD gain (electrons/DN)
(snoise = "0.")	ccdclip: Sensitivity noise (fraction)
(sigscale = 0.1)	Tolerance for sigma clipping scaling
(pclip = -0.5)	pclip: Percentile clipping parameter
(grow = 0.)	Radius (pixels) for neighbor rejection
(mode = "q1")	

B. Fixpix

Once the flats have been combined, use `fixpix` with your bad pixel mask on the combined flats.

```
images = Iflat.fits      List of images to be fixed
masks = "conflatbpml.pl" List of bad pixel masks
(linterp = "INDEF")     Mask values for line interpolation
(cinterp = "INDEF")     Mask values for column interpolation
(verbose = yes)          Verbose output?
(pixels = no)           List pixels?
(mode = "ql")
```

C. Using Conflat

If you have used OT guiding you will need to use the `conflat` script to correct your flats to your ot guided images. See section 6.5 of the OPTIC at WIYN manual for a full description of `conflat`.

To use `conflat` you will need the `*_ot.***` files and the combined flat for that filter. For example, if you want to convolve the I flat to match the OT shifting for image 100 you would use the syntax:

```
cl> ! ./conflat Iflat.fits a_ot.100 Flat100.fits
```

to produce the convolved flat `Flat100.fits`

Once you have convolved flats for each of your OT guided images you can then use the `mkmsc` task to break them up into multiextension fits files.

```
mkmsc Flat100.fits mscFlat100.fits descrip=optic1.mkmsc
```

D. Overscan subtracting and trimming the flats

Then you will need to again use `ccdproc` to overscan subtract and trim all of the Flats. One could zero correct the flats instead of overscan subtracting, if desired.

```
cl> epar ccdproc
```

```
    images = "mscFlat100"    List of Mosaic CCD images to process
  (output = "mscFlat100ot") List of output processed images
  (bpmmasks = " ")         List of output bad pixel masks
  (ccdtype = " ")         CCD image type to process
  (noproc = no)           List processing steps only?\n
  (xtalkcor = no)         Apply crosstalk correction?
  (fixpix = no)           Apply bad pixel mask correction?
  (overscan = yes)        Apply overscan strip correction?
  (trim = yes)            Trim the image?
  (zerocor = no)          Apply zero level correction?
  (darkcor = no)          Apply dark count correction?
  (flatcor = no)          Apply flat field correction?
  (sflatcor = no)         Apply sky flat field correction?
  (split = no)            Use split images during processing?
  (merge = no)            Merge amplifiers from same CCD?
  (xtalkfile = " ")       Crosstalk file
  (fixfile = "BPM")       List of bad pixel masks
  (saturation = "INDEF")  Saturated pixel threshold
  (sgrow = 0)              Saturated pixel grow radius
  (bleed = "INDEF")       Bleed pixel threshold
  (btrail = 20)           Bleed trail minimum length
  (bgrow = 0)              Bleed pixel grow radius
  (biassec = "!biassec")  Overscan strip image section
  (trimsec = "!trimsec")  Trim data section
  (zero = " ")            List of zero level calibration images
  (dark = "Dark")         List of dark count calibration images
  (flat = " ")            List of flat field images
  (sflat = "Sflat*")      List of secondary flat field images
  (minreplace = 1.)       Minimum flat field value\n
  (interactive = no)      Fit overscan interactively?
  (function = "legendre") Fitting function
  (order = 1)             Number of polynomial terms or spline
  (sample = "*" )         Sample points to fit
  (naverage = 1)          Number of sample points to combine
  (niterate = 1)          Number of rejection iterations
  (low_reject = 3.)       Low sigma rejection factor
  (high_reject = 3.)      High sigma rejection factor
  (grow = 0.)             Rejection growing radius
  (fd = "")               (fd = "")
  (fd2 = "")              (fd2 = "")
  (mode = "q1")          (mode = "q1")
```

Step 5 - Correcting object frames

A. Fixpix

First you will need to edit the parameter file `fixpix` to note your images and the name of your bad pixel mask. Then run the task to correct the bad regions in your images.

```
cl> epar fixpix
```

```
images = "a100f.fits"    List of images to be fixed
masks = "conflatbpm.pl" List of bad pixel masks
(linterp = "INDEF")     Mask values for line interpolation
(cinterp = "INDEF")     Mask values for column interpolation
(verbose = yes)         Verbose output?
(pixels = no)          List pixels?
(mode = "ql")
```

B. Mkmisc

Once again run the `mkmisc` task, but this time on the object frames.

```
cl> epar mkmisc
```

```
PACKAGE = mscred
TASK = mkmisc

input =      a100f.fits List of input images
output =    msca100f.fits List of output mosaic MEF files
(descrip=   optic1.mkmisc) Description file
(verbose=   yes) Verbose input?
(mode=     ql)
```

C. Overscan subtracting and trimming object frames

Once again run the task ccdproc to overscan subtract (or zero correct) and trim all object frames.

```
images = "msca100f"      List of Mosaic CCD images to process
(output = "msca100otf") List of output processed images
(bpmasks = " ")         List of output bad pixel masks
(ccdtype = " ")         CCD image type to process
(noproc = no)           List processing steps only?\n
(xtalkcor = no)         Apply crosstalk correction?
(fixpix = no)           Apply bad pixel mask correction?
(overscan = yes)        Apply overscan strip correction?
(trim = yes)            Trim the image?
(zeroeor = no)          Apply zero level correction?
(darkcor = no)          Apply dark count correction?
(flatcor = no)          Apply flat field correction?
(sflatcor = no)         Apply sky flat field correction?
(split = no)            Use split images during processing?
(merge = no)            Merge amplifiers from same CCD?\n
(xtalkfile = " ")      Crosstalk file
(fixfile = "BPM")       List of bad pixel masks
(saturation = "INDEF") Saturated pixel threshold
(sgrow = 0)             Saturated pixel grow radius
(bleed = "INDEF")       Bleed pixel threshold
(btrail = 20)           Bleed trail minimum length
(bgrow = 0)             Bleed pixel grow radius
(biassec = "!biassec") Overscan strip image section
(trimsec = "!trimsec") Trim data section
(zero = " ")            List of zero level calibration images
(dark = "Dark")         List of dark count calibration images
(flat = " ")            List of flat field images
(sflat = "Sflat*")     List of secondary flat field images
(minreplace = 1.)       Minimum flat field value\n
(interactive = no)      Fit overscan interactively?
(function = "legendre") Fitting function
(order = 1)             Number of polynomial terms or spline
(sample = "*" )         Sample points to fit
(naverage = 1)          Number of sample points to combine
(niterate = 1)          Number of rejection iterations
(low_reject = 3.)       Low sigma rejection factor
(high_reject = 3.)      High sigma rejection factor
(grow = 0.)             Rejection growing radius
(fd = "")
(fd2 = "")
(mode = "ql")
```

D. Flatfield correcting the object frames.

Run `ccdproc` again, this time flatfield correcting the object frames. If you used OT guiding, be sure to use the convolved flats for each image. Remember, you will be using a unique flat for each image – set the `flat` flag to the appropriate convolved flat for that image in `ccdproc`.

```
cl> epar ccdproc
```

```
images = "msca100otf" List of Mosaic CCD images to process
(output = "msca100otff") List of output processed images
(bpmsk = " ") List of output bad pixel masks
(ccdtype = " ") CCD image type to process
(noproc = no) List processing steps only?\n
(xtalkcor = no) Apply crosstalk correction?
(fixpix = no) Apply bad pixel mask correction?
(overscan = no) Apply overscan strip correction?
(trim = no) Trim the image?
(zerocor = no) Apply zero level correction?
(darkcor = no) Apply dark count correction?
(flatcor = yes) Apply flat field correction?
(sflatcor = no) Apply sky flat field correction?
(split = no) Use split images during processing?
(merge = no) Merge amplifiers from same CCD?\n
(xtalkfile = " ") Crosstalk file
(fixfile = "BPM") List of bad pixel masks
(saturation = "INDEF") Saturated pixel threshold
(sgrow = 0) Saturated pixel grow radius
(bleed = "INDEF") Bleed pixel threshold
(btrail = 20) Bleed trail minimum length
(bgrow = 0) Bleed pixel grow radius
(biassec = "!biassec") Overscan strip image section
(trimsec = "!trimsec") Trim data section
(zero = " ") List of zero level calibration images
(dark = "Dark") List of dark count calibration images
(flat = "mscFlat100ot") List of flat field images
(sflat = "Sflat*") List of secondary flat field images
(minreplace = 1.) Minimum flat field value\n
(interactive = no) Fit overscan interactively?
(function = "legendre") Fitting function
(order = 1) Number of polynomial terms or spline
(sample = "*" ) Sample points to fit
(naverage = 1) Number of sample points to combine
(niterate = 1) Number of rejection iterations
(low_reject = 3.) Low sigma rejection factor
(high_reject = 3.) High sigma rejection factor
(grow = 0.) Rejection growing radius
(fd = "")
(fd2 = "")
(mode = "ql")
```

You are now the proud owner of fully reduced OPTIC data – analyze away!