
Basic SparsePak Reductions Guideline v2.1

This document is based on a compilation of reduction notes from D. Andersen, M. Verheijen, K. Westfall, and M. Bershady. Recommendations are given for book-keeping format based on what is required for the DiskMass Survey SparsePak SVD analysis.

last update: 06 Feb 2008 [MAB]

Planned upgrades:

I. Directory structure for processing

In the following instructions and steps, the square brackets, "[]", are place-holders and are not meant to be typed.

Raw and reduced (rdx) data are stored in directories for each run (run_name) and by night:

```
[run_name]/n[x]/raw             raw data files
[run_name]/n[x]/rdx             reduced data files
```

where x = 1,...number of nights for that run. We copy raw data from archive or other site provided into [run_name]/n[x]/raw via a tar command.

Example:

a. Observing run to process is [run_name], with five nights (n1,n2,n3,n4,n5).

b. The archive location is:

```
[archive_path]
```

where it is assumed here that [obs_run] sits in the top level of [archive_path].

c. You copy the data to: [your_rdx_disk_path]

d. You copy the data by executing (in a linux shell, not iraf):

```
-> cd [archive_path]
-> tar cf - [run_name] | (cd [your_rdx_disk_path]; tar xf - )
```

When you are done processing, delete raw data directories in your copy. We make archive copies of the reduced data directories in the same way as we make copies of the raw data, so it is important to keep the same format.

II. Overview of processing

(i) Reduce data from runs on a night-by-night basis.

(ii) In each rdx directory for each night of each run, create and keep a "Notes" file which contains:

a. your name
b. date(s) of processing (what steps)

c. any notes of problems, oddities or challenges
d. image-groupings for each object for crclean and dohydra steps
e. parameter lists for hydra.dohydra and hydra.params

(iii) Begin by identifying the different frames, and make lists for each, as appropriate, as you work your way through the reduction steps.

Start with some general lists for each night's rdx directory:

```
all_raw.lst - "OT" CCDPROC input - see step 1 below (Section III)
all.lst     - "OT" CCDPROC output - see step 1 below (Section III)
all_ot.lst  - "Z" CCDPROC step input - copy all.lst and
              delete bias frames from list.
```

a. bias (or zero):

zero_n[x].lst, list of ot-corrected images input to zerocombine

b. dome flats

flat_n[x].lst - list of otz-corrected images input to flatcombine

c. line-lamps or comps (ThAr, CuAr, etc.):

comp_n[x].lst - list of otz-corrected images input for CR-cleaning

d. object (standards, targets, sky)

```
[obj]_[group].[band].lst - list of otz-corrected images input for
                            CR-cleaning.
                           [obj],[group],[band] defined below.

pointings.lst - input list to dohydra with names of CR-cleand and
                combined images.
```

You will inspect these images as you go through the reductions.

Note conditions and when the dewar was filled. (If there was a dewar fill in the middle of the night you will need to create two line-lamp masters and two wavelength solutions for different sets of data before and after the fill.) Keep a record of notes for your steps and file names; this will make it easier to redo things if you need to later on.

(iv) List of processing steps:

1. "OT" ccdproc: Overscan [O] correct and trim [T] *all* raw images.

2. zerocombine: Construct the Bias frames from the OT'ed zero frames. Exclude all zero frames with funky statistics. These usually occur at the beginning of a zero-series. if there is no discernable difference between zero frames taken before and after a dewar fill, then use all of them to create a single Bias image. A single bias image is fine for the entire night.

3. "Z" ccdproc: Subtract the appropriate Bias frame from the OT'ed flat, comp and obj frames. This takes out any 2-dimensional structure in the Bias frame. It is important to use as many zero frames as possible to minimize the addition of noise.

4. flatcombine: Construct the Flat frames from the OTZ'ed flats. As with the zero frames, exclude frames with unusual statistics (combine fames with the same

exposure and lamp levels).

5. crclean: and/or imclean — Remove the cosmics and combine the OTZ'ed comp and obj frames. With crclean For the strong emission lines in the comp frames it is best to use a threshold of 12-sigma, while for the obj frames it is best to use a 6-sigma threshold. However, you may want to inspect a difference image between the crclean result and the input images.

   Grouping object and comp frames together at this stage to run through crclean is important. At this stage inspect the individual images to make sure they have the same mean intensity (exposure length should be the same; conditions and telescope pointing should be the same). Look for spectral shifts in the comp frames, or for guiding errors in the obj frames which were suppossed to have been taken at the same position (this is manifested by a different illumination pattern from fiber-to-fiber. Add any notes and comments to the Notes file.

   When in doubt, combine fewer frames into crclean. A minimum of two will work; with three, it works well.

   If only 1 image, use imclean. If several images with the same pointing, but can't be combined with crclean, use multimclean. If more than 7 images, use imcombine with crreject option.

6. colclean: Fix the bad colums of the chip. This requires an input control file indicating where the bad columns are. Note that the position of the bad columns vary from run to run because at each run, a different part of the chip was read out. This is only relevant to earlier data taken with T2KC; T2KA has no bad columns.

7. dohydra: Extract the spectra and determine the dispersion function.

8. dispcor: Apply the dispersion solutions to the ms-file in order to create rectified linear and logarithmic spectra. make sure that all the pointings of the same galaxy have identical w1, w2 and dw values. Mask files are produced here as well.

9. errors: Calculate error spectra based on first-principles for the specific, combined, sampled data processed through dohydra. This uses proprietary code in the ifupkg.

10. skysub: Substract sky and calculate errors in sky-subtracted data using the hydra.skysub program and proprietary code in the ifupkg for the errors. Sky-subtraction here is the straight mean of the good sky fibers.

(v) Products - stored in each night's rdx directory

Calibration images : bias (step 2), dome (step 4) and line-lamps (step 5)

Use these standard names for the output of zerocombine and flatcombine, which should be stored in each night's rdx directory.

Zero_n[x]	x = 1,2, ... number of nights
Flat_n[x]	x = 1,2, ... number of nights

Use these standard names for the output of crclean with the comp (line-lamp) images

[lamp]_n[x]	x = 1,2, ... number of nights; use 1a,1b etc. if multiple per night because of dewar re-fill
	lamp = ThAr or CuAr

Object images :

Output of crclean, dohydra and dispcor steps: THESE ARE THE REDUCED DATA PRODUCTS.

*.fits	- Crclean'd or imclean'd image
*.ms.fits	- Output from dohydra (not linearized or sky subtracted)
*.ms_log.fits	- log-rectified (resampled) ms file (dispcor: step 8)
*.ms_lin.fits	- linear-rectified (resampled) ms file (dispcor: step 8)
*.ms_m_lin.fits	- linear-rectified (resampled) ms mask file (dispcor: step 8)
*.ms_m_log.fits	- log-rectified (resampled) ms mask file (dispcor: step 8)
*.sig.fits	- Error image based on first-principles calculation
*.me.fits	- Dohydra extraction of error image
*.me_log.fits	- Log-rectified (resampled) me file
*.me_lin.fits	- linear-rectified (resampled) me file
*.ms_s_log.fits	- Sky-subtracted ms file, log-rectified
*.me_s_log.fits	- Error spectra after sky-subtraction, log-rectified
*.ms_s_lin.fits	- Sky-subtracted ms file, lin-rectified
*.me_s_lin.fits	- Error spectra after sky-subtraction , lin-rectified

where * = [obj]_[group].[band]

obj = U[xxxxx]	for most galaxies -- and all galaxies in the Diskmass survey. U stands for UGC and xxxxx is the 5-digit serial number, left-padded with zeros, i.e., UGC 463 becomes U00463_ etc. We will give equivalent UGC numbers for galaxies identified in the logs or headers by some other catalogue name (e.g., NGC or PGC)
= HR[xxxxxx]	for template or flux-calibration stars.
group = 1,...n	this is the grouping that you assign to a set of observations of one target on a given night for purposes of running through crclean and then do hydra. Put this information in the "Notes" file in each night's directory specifying which individual imags below to each group number. In the few cases where an object is observed on more than one run, we will want to make this group # a running number that is unique for each pointing, across runs.
band = mg,ha,ca	mg: MgI region centered near 510 nm, observed with echelle and VPH configurations.
	ha: Ha reg region centered near 680 nm, observed with echelle and 860 l/mm grating.

ca: CaII region centered near 870 nm, observed
with echelle or VPH configurations.

III. Step by step - inputs for the various tasks

---

1. ccdproc - the "OT" step

The structure in the overscan region varies significantly in consecutive readouts. Therefore, it is neccessary to first overscan correct the raw images before they can be combined into a single Bias or Flat frame for instance. Use a 100th order function to be able to follow the sometimes sharp structures in the overscan region.

(Note for future observing: for a faster-reading device, an increase in the overscan region should enable us to avoid fitting a function but instead subtract the mean overscan values row-by-row; a wider overscan region would give a better S/N in the mean.)

Note that the overscan and trim sections are read from the FITS header of the image, which you can view by typing:

```
cl> imhead [image name] l+ | match TRIMSEC
cl> imhead [image name] l+ | match DATASEC
cl> imhead [image name] l+ | match BIASSEC
```

where you replace '[image name]' with the image name, or a wild card, and,

```
BIASSEC:  overscan portion of frame
TRIMSEC:  region to be extracted
DATASEC:  image portion of frame
```

The following are also useful to note:

```
CCDSEC:   orientation to full frame
ORIGSEC:  original size full frame
```

These tell you the size of the full CCD, and what portion was actually read out.

Use this OT step to move images from the raw/ to the rdx/ directory. Create a list called all_raw.lst simply by listing the contents of the fits images in ../raw from the rdx directory:

```
cl> ls ../raw/*.fits > all_raw.lst
```

Also create all.lst in the rdx/ directory either by editing all_raw.lst and removing the ../raw prefix or doing the same ls command in the raw/ directory and copying the file to rdx.

Switch into the rdx directory, and do all further processing there, including this ccdproc step.

```
PACKAGE = ccdred
   TASK = ccdproc

images  =        @all_raw.lst   List of CCD images to correct
(output =        @all.lst)      List of output CCD images
(ccdtype=                     ) CCD image type to correct
(max_cac=                    0) Maximum image caching memory (in Mbytes)
(noproc =                   no) List processing steps only?

(fixpix =                   no) Fix bad CCD lines and columns?
(oversca=                  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?
   (illumco=                   no) Apply illumination correction?
   (fringec=                   no) Apply fringe correction?
   (readcor=                   no) Convert zero level image to readout correction?
   (scancor=                   no) Convert flat field image to scan correction?

   (readaxi=                 line) Read out axis (column|line)
   (fixfile=                     ) File describing the bad lines and columns
   (biassec=                image) Overscan strip image section
   (trimsec=                image) Trim data section
   (zero   =                     ) Zero level calibration image
   (dark   =                     ) Dark count calibration image
   (flat   =                     ) Flat field images
   (illum  =                     ) Illumination correction images
   (fringe =                     ) Fringe correction images
   (minrepl=                  1.) Minimum flat field value
   (scantyp=            shortscan) Scan type (shortscan|longscan)
   (nscan  =                    1) Number of short scan lines

   (interac=                  yes) Fit overscan interactively?
   (functio=             legendre) Fitting function
   (order  =                  100) 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
```

2. zerocombine - make the average bias (zero) image

Determine the average Bias (or Zero) image after a crreject of cosmics and other outlying pixels. The individual images are first scaled with their modes, i.e. their most common values.

When creating the input list (here the example is Zero_n1.lst), check the individual frames (display and imstat) to remove bad frames. Often the first one or few in a sequence are bad. In the rare cases where they are available, you can combine bias frames at the beginning and end of night to improve noise statistics. A bad bias frame will either have a subtantially different mean than the rest, or show evidence for charge buildup (T2KC in particularl) at the bottom of the chip.

Note that the output name can have a different suffix in case there are multiple Bias frames to be used in a single night. You will rarely need to do this.

Also note that the actual readnoise and gain of the chip may be different from what the FITS header advertises. You can set these values by hand in the parameters, as given in the following table, or specify the image header keywords "RDNOISE" and "GAIN." The headers are usually correctly populated, but you should check. We are adopting the nominal gain values from the header, but have measured the read-noise, adopting these gain values. (A straight-forward recipe for determining the true effective value of the readnoise from imstatistics on the zero images is found elsewhere).

USE THESE VALUES and *not* the keywords in zerocombine.

CCD	gain	rdnoise	
STA1042	TBD	TBD	new device (not yet implemented)

```
          T2KA    2.1     6.7     current device (since 24 Sep 2004 run)
          T2KC    1.7     4.9     old device, unbinned data
          T2KC    1.7     5.3     old device, binned data (2x1)

   You can determine the detector type again by querying the image headers:
          cl> imhead [image name] l+ | match DETECTOR
   and you will get
          DETECTOR= 't2ka              '  /  detector name
   for T2KA.

   Use these standard names for lists and output of zerocombine,
   which should be stored in each night's rdx directory.

   zero_n[x].lst           list of ot-corrected images input to zerocombine
   Zero_n[x].fits          x = 1,2, ... number of nights


   PACKAGE = ccdred
      TASK = zerocombine

   input   =          @zero_n1.lst List of zero level images to combine
   (output =         Zero_n1.fits) Output zero level name
   (combine=              average) Type of combine operation
   (reject =             crreject) Type of rejection
   (ccdtype=                     ) CCD image type to combine
   (process=                   no) Process images before combining?
   (delete =                   no) Delete input images after combining?
   (clobber=                   no) Clobber existing output image?
   (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=                  6.7) ccdclip: CCD readout noise (electrons)
   (gain   =                  2.1) 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



   3. ccdproc - the "Z" step: bias image subtraction
   ==========

   Subtract the average bias image from the other ot-ed non-bias
   images. The name of the "zero" image in the parameter list
   below is nominally the image you created in the previous step.

   Lists:  all_ot.lst              "Z" CCDPROC step input - copy all.lst and
                                   delete bias frames from list.

   We don't specify an output list because we want ccdproc in this
   case to over-write the input list.


   PACKAGE = ccdred
      TASK = ccdproc

   images  =         @all_ot.lst   List of CCD images to correct
   (output =                     ) List of output CCD images
   (ccdtype=                     ) CCD image type to correct
   (max_cac=                   0) Maximum image caching memory (in Mbytes)
```

```
   (noproc =                   no) List processing steps only?

   (fixpix =                   no) Fix bad CCD lines and columns?
   (oversca=                   no) Apply overscan strip correction?
   (trim   =                   no) Trim the image?
   (zerocor=                  yes) Apply zero level correction?
   (darkcor=                   no) Apply dark count correction?
   (flatcor=                   no) Apply flat field correction?
   (illumco=                   no) Apply illumination correction?
   (fringec=                   no) Apply fringe correction?
   (readcor=                   no) Convert zero level image to readout correction?
   (scancor=                   no) Convert flat field image to scan correction?

   (readaxi=                 line) Read out axis (column|line)
   (fixfile=                     ) File describing the bad lines and columns
   (biassec=                image) Overscan strip image section
   (trimsec=                image) Trim data section
   (zero   =                 Bias) Zero level calibration image
   (dark   =                     ) Dark count calibration image
   (flat   =                     ) Flat field images
   (illum  =                     ) Illumination correction images
   (fringe =                     ) Fringe correction images
   (minrepl=                   1.) Minimum flat field value
   (scantyp=            shortscan) Scan type (shortscan|longscan)
   (nscan  =                    1) Number of short scan lines

   (interac=                  yes) Fit overscan interactively?
   (functio=             legendre) Fitting function
   (order  =                  100) 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
```

4. flatcombine - making the flat-field spectrum

Construct an average flat field image. Note that the name of the input list and output image may be different in case there are multiple Flat images to be produced in a night. Also pay attention to the values of rdnoise and gain, as per step 2. Also review the flat images to make sure they all have roughly the same counts.

Use the gain and readnoise values given in Step 2.

Use these standard names for lists and output of flatcombine, which should be stored in each night’s rdx directory.

```
flat_n[x].lst           list of ot-corrected images input to flatcombine
Flat_n[x].fits          x = 1,2, ... number of nights
```

```
   PACKAGE = ccdred
      TASK = flatcombine

   input   =          @flat_n1.lst List of flat field images to combine
   (output =         Flat_n1.fits) Output flat field root name
   (combine=              average) Type of combine operation
   (reject =             crreject) 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?
(clobber=                  no) Clobber existing output image?
(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=                 6.7) ccdclip: CCD readout noise (electrons)
(gain   =                 2.1) 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
```

5. crclean - or imclean; cleaning out cosmic rays from object and line-lamp images

There are several options for this key step of cleaning cosmic rays (CRs). CRs are the dominant source of catastrophic errors in the data, so determining the best option is important.

The best way to clean CRs is to have multiple exposures of the same length, taken under similar conditions, and with the same telescope pointing. Data should be taken with this in mind. In this case, images can be differenced, in pair-wise fashion, with large positive and negative deviations (compared to a clipped standard deviation) identified as CRs. These are flagged with a mask, so that the final combination (average) of images gives these pixels zero weight in the relevant image. The crclean routine carries out this operation -- OPTION a. With only two frames there is some chance that CRs will hit a given pixel in both images. Inspection will reveal the level of the problem. With 3 images, the cleaning is very good. The crclean code works for up to 7 images. In this "best" option, we find it is best to do a limited, tertiary cleaning with imclean on the combined image. The imclean option is described below.

If weather or other conditions do not permit more than one image for a given pointing, depth, or conditions (which you can determine by manually taking the difference of two images and seeing if you see large residuals where the fibers are located), then OPTION b should be selected, which employs imclean only. This routine calls the noao.imred.crutil.cosmirays program, which searches for cosmic rays based on their shape. For well-sampled data, it is easy to distinguish cosmic rays. For the binned SparsePak data, we are usually ok too. In this implementation, it’s also important to throw out pixels neighoring the CR even since there is usually some low-level energy from the CR that can’t be detected by the cosmirays program.

If weather, etc. yields several images that need to be cleaned with imclean, this can be done in an optimal way with multimclean -- OPTION c.

If you are lucky enough to have more than 7 images which can be combined with something like crclean, then use imcombine -- OPTION d.

Imcombine is a standard IRAF routine in images.immatch. Crclean, imclean, and multimclean are proprietary programs in birpkgs.gbupkg. You will need to have this in your loginuser.cl in your IRAF home directory (~/iraf/):

```
reset birpkgs   = "~mab/iraf/scripts/birpkgs/"
task $birpkgs   = "birpkgs$birpkgs.cl"
```

and load birpkgs and gbupkg.

Inputs and outputs:

Comparison lamps -

comp_n[x].lst	list of otz-corrected images input to crclean or imclean
[lamp]_n[x].fits	output image for comps x = 1,2, ... number of nights; use 1a,1b etc. if multiple per night because of dewar re-fill lamp = ThAr or CuAr

Objects -

[obj]_[group].[band].lst	list of otz-corrected images input to crclean or imclean
*.fits	output image, crclean'd or imclean'd where * = [obj]_[group].[band], as defined in Section II above.

OPTION a. crclean + imclean

Remove the CRs from otz-ed images where there are multiple frames for the same object taken with the same conditions and telescope pointing. The image sets are combined into a single output image. This is a two-step process: First the images are combined with crclean, and then the combined image is re-cleaned using imclean to remove some remaining CRs. You should view the output of crclean (which becomes the input of imclean), and compare it to the output of imclean before deleting the intermediate step.

crclean parameters:

In all data CHECK THAT BRIGHT LINES ARE NOT ADVERSELY AFFECTED BY THE CRCLEAN PROCEDURE.

For science frames (object or sky) lower the detection threshold to threshold=6. For line-lamp exposures use threshold=12.

Again, pay attention to the readnoise and gain of the chip; this routine does not take header key-words.

CCD	gain	rdnoise	
STA1042	TBD	TBD	new device (not yet implemented)
T2KA	2.1	6.7	current device (all runs since 24 Sep 2004 run)
T2KC	1.7	4.9	old device, unbinned data
T2KC	1.7	5.3	old device, binned data (2x1)

Weighting and scaling with crclean: The program now allows you to combine images with different exposure lengths to deal with shortened exposures due to telescope-wraps that were otherwise taken in stable conditions and constant telescope pointing (these two conditions need to be verified by image inspection). Setting weight=keyword, scale=keyword, and wkey=EXPTIME and skey=EXPTIME will always do the right operation even if the exposures are all the same length. Use the wmode=RN-limit for the SparsePak data. In the case where the

exposures are the same length, you can also set weight=none and scale=none.

When the list of images contains non-uniform exposures, list the shortest exposures last. THIS IS IMPORTANT.

```
PACKAGE = gbupkg
   TASK = crclean

imtmpl  =         @comp_n1.lst  template of images
output  =     tmp_ThAr_n1.fits  Output Image
(thresho=                  12) Threshold (in sigma) for cr flagging
(fixtype=             replace) Bad pix. fix: interp (lin.) or replace ?
(xinterp=                   5) Size of x interpolation box for fixtype=interp
(yinterp=                   5) Size of y interpolation box for fixtype=interp
(eadu   =                 2.1) Gain = electrons per ADU (DN)
(rn     =                 6.7) Readnoise in electrons

(weight =             keyword) Weight? (template|keyword|none)
(wtempla=                none) Weights-template (must match imtmpl)
(wkeywor=             EXPTIME) Weight header keyword
(wmode  =            RN-limit) Weight transformation (value|RN-limit|photon-limit)

(scale  =             keyword) Scale? (template|keyword|none)
(stempla=                none) Scales-template (must match imtmpl)
(skeywor=             EXPTIME) Scale header keyword
(srefere=                 max) Reference type for scaling (value|min|max|mean)
(srefval=                  1.) Reference scaling value (updated if min,max,mean)

(verbose=                 yes) Print status messages ?
```

*** NOTE: threshold = 12 is for the line-lamp spectra, and should be set to 6 for object spectra.

imclean parameters:

The implementation of imclean here is limited, since the primary goal is to find the few remaining strong, CRs that sometimes remain after crclean, but avoid removing any source or sky flux. The following parameters have window and threshold values set differently than what is used below.

```
PACKAGE = gbupkg
   TASK = imclean

imlist  =     tmp_ThAr_n1.fits  Input image list (template)
cleanlis=         ThAr_n1.fits  Ouput image list (template)
gpm     =                 none  Good pix mask (1=good; 0=bad), or none
(npasses=                    5) Cosmicrays npasses parameter (nominal value 5)
(fluxrat=                  8.5) Cosmicrays fluxrat parameter (nominal value 8.5)
(window =                    5) Cosmicrays window parameter (nominal value 7)
(thresho=                  10.) Cosmicrays threshold parameter in units of stddev (nom
inal value 5)
(stat_ty=            iterstats) Stddev type: iterstats or goodstats
(nrep   =                    3) Repeat cosmic ray finding nrep times
(nneigh =                    1) Repeat adding nearest neighbors to CR map nneigh times
(nx0    =                   11) X dim of initial cleaning box (using input gpm)
(ny0    =                    1) Y dim of initial cleaning box (using input gpm)
(nxi    =                    1) X dim of CR cleaning box
(nyi    =                    3) Y dim of CR cleaning box
(fixall =                  yes) Repeat bad pixel fixing until all are fixed? (CRs only
)
(verbose=                  yes) Print action ?
(display=                   no) Display most results for each iteration?
```

```
   (interac=                   no) Use cosmicrays in interactive mode
   (keepmas=                  yes) Keep masks used to clean cosmicrays?
   (masklis= @imclean_outmask.lst) Template of masks needed if keepmask=yes
```

OPTION b. imclean only

Remove the CRs in object or line-lamp frames where there is only one frame. You will need to load the noao.imred.crutil package. In addition to input and output image lists, also save the masks that are generated to clean the images (see masklist below).

You will likely need to make special input and output lists, but keep file naming conventions for output images.

For the masklist file, use the same root name as the output images, but put a "crmask_" in front of each name.

```
   PACKAGE = gbupkg
      TASK = imclean

   imlist  =   @imclean_input.lst  Input image list (template)
   cleanlis=  @imclean_output.lst  Ouput image list (template)
   gpm     =                 none  Good pix mask (1=good; 0=bad), or none
   (npasses=                    5) Cosmicrays npasses parameter (nominal value 5)
   (fluxrat=                  8.5) Cosmicrays fluxrat parameter (nominal value 8.5)
   (window =                    7) Cosmicrays window parameter (nominal value 7)
   (thresho=                   5.) Cosmicrays threshold parameter in units of stddev (nom
   inal value 5)
   (stat_ty=             iterstats) Stddev type: iterstats or goodstats
   (nrep   =                    3) Repeat cosmic ray finding nrep times
   (nneigh =                    1) Repeat adding nearest neighbors to CR map nneigh times
   (nx0    =                   11) X dim of initial cleaning box (using input gpm)
   (ny0    =                    1) Y dim of initial cleaning box (using input gpm)
   (nxi    =                    1) X dim of CR cleaning box
   (nyi    =                    3) Y dim of CR cleaning box
   (fixall =                  yes) Repeat bad pixel fixing until all are fixed? (CRs only
   )
   (verbose=                  yes) Print action ?
   (display=                   no) Display most results for each iteration?
   (interac=                   no) Use cosmicrays in interactive mode
   (keepmas=                  yes) Keep masks used to clean cosmicrays?
   (masklis= @imclean_outmask.lst) Template of masks needed if keepmask=yes
```

*** Note: The nxi,nyi, and fixall parameters are critical.

OPTION c. multiple imclean: multimclean

Information will be added as needed.

OPTION d. more than 7 frames to combine together: imcombine

The key parameters are combine=average and rejec=crreject. In addition, you must set the rdnoise and gain parameters properly for the right CCD. (The example below is for T2KA).
The example shown here uses an input list caleed image.lst and an output image with name output_image.fits.

If all of the images have the same exposure time, set scale=none and weight=none. Otherwise set these as given below. IN THIS CASE where scale=exposure, you must rescale the output spectrum back to the DN level corresponding to the maximum exposure. Look in the image headers of the input list and then:

```
          imarith output_image.fits * exptime output_image.fits
```

where exptime is the integer or real number you have found
for the longest exposure in the list.

```
PACKAGE = immatch
   TASK = imcombine

input   =          @image.lst   List of images to combine
output  =   output_image.fits   List of output images
(headers=                     ) List of header files (optional)
(bpmasks=                     ) List of bad pixel masks (optional)
(rejmask=                     ) List of rejection masks (optional)
(nrejmas=                     ) List of number rejected masks (optional)
(expmask=                     ) List of exposure masks (optional)
(sigmas =                     ) List of sigma images (optional)
(logfile=               STDOUT) Log file

(combine=              average) Type of combine operation
(reject =             crreject) Type of rejection
(project=                   no) Project highest dimension of input images?
(outtype=                 real) Output image pixel datatype
(outlimi=                     ) Output limits (x1 x2 y1 y2 ...)
(offsets=                 none) Input image offsets
(masktyp=                 none) Mask type
(maskval=                    0) Mask value
(blank  =                   0.) Value if there are no pixels

(scale  =             exposure) Image scaling
(zero   =                 none) Image zero point offset
(weight =             exposure) 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 (neg)
(mclip  =                  yes) Use median in sigma clipping algorithms?
(lsigma =                   3.) Lower sigma clipping factor
(hsigma =                   3.) Upper sigma clipping factor
(rdnoise=                  6.7) ccdclip: CCD readout noise (electrons)
(gain   =                  2.1) ccdclip: CCD gain (electrons/DN)
(snoise =                   0.) ccdclip: Sensitivity noise (fraction)
(sigscal=                  0.1) Tolerance for sigma clipping scaling corrections
(pclip  =                 -0.5) pclip: Percentile clipping parameter
(grow   =                   0.) Radius (pixels) for neighbor rejection
(mode   =                   ql)
```

6. colclean - repairing bad columns (T2KA only)

This program takes a specific format input file that identifies the
location of bad columns (in x) and their extent in y, one column at a
time, as well as two neighboring columns that will be used to fix the
column. The fix requires identifying rows where all three columns are
good. This region is used to define a scaling for the two good
columns over the rows where the bad column is bad.

Identification: This need to be done by hand ONCE per run, and should
be done by visually inspecting the combinned, cleaned dome flat and

zero frames. There are three bad regions that have been identified to date. Their location in y (row) should be the same in every run, but the location in x (column) will change due to how SPK is mounted on the fiber foot mount and what x-portion of the CCD is saved.

(a) two adjacent dark columns (seen most easily in the dome flat) starting around row (y) 1831 and extending to the end (2048) of the detector. The x values are in the range of 700 to 1100 pixels, or the binned equivalent (350 to 550).

(b) two independent hot columns (seen most easily in the zero) starting in row (y) 1118 and extended up to 1350, roughly in column (x) 600-800 or its binned equivalent (300-400); and another starting in row (y) 390 and extending up to 550, roughly in column (x) 1000-1200 or the binned equivalent (500-600).

Choice off good columns: You must look carefully where the bad columns falls w.r.t. the fiber data. (For the hot columns found in the zero frames, blink against the dome flat.) If the bad column is in the middle of a fiber spectrum, the good columns should be immediately adjacent. if the bad column is on the edge of the fiber data (next to the trough), then the two good columns should be the same column on the data-side of the bad column. The example below will help.

Choice of good rows: Take +100 and -100 of the bad-row region.

```
PACKAGE = gbupkg
   TASK = colclean

itmpl   =                       Input template of images
otmpl   =                       Matching template of output (cleaned) images
cfile   =                       Control file (format in program header)
(clobber=                   yes) Delete output image if it exists?
(verbose=                   yes) Print steps
```

Input control-file format:

```
       total_number_of_bad_columns
       bad_column good_column_1 good_column_2
       start_bad_row end_bad_row
       start_good_row1 end_good_row1
       start_good_row2 end_good_row2
```

Here’s an example based on data from PILSP_08May02/n1: The first two are the bright columns seen in the zero frame. The last two are the adjacent dark columns seen in the dome flat. Note how the good columns are chosen. The data is binned 2x1.

```
4                       # total number of bad columns
356 355 357             # set 1: bad_column good_column_1 good_column_2
1117 1350               # set 1: start_bad_row end_bad_row
1017 1450               # set 1: start_good_row1 end_good_row1
1017 1450               # set 1: start_good_row2 end_good_row2
543 542 544             # set 2: bad_column good_column_1 good_column_2
389 525                 # etc...
289 625
289 625
498 497 497             # set 3
1832 2047
1800 1830
1800 1830
499 500 500             # set 4
1832 2048
1732 1831
```

1732 1831

7. dohydra - this is the major spectral processing step
==========

Use dohydra to extract spectra, flatten specta, and determine dispersion correction / wavelength solution (which is written to the header). Our preference is to not use dohydra to apply corrections for scattered light, the SED of the flat-field lamps, or bad pixels. The latter should be taken care of in previous steps. While we have explored numerous methods for sky-subtraction, we recommend starting with a straight average of sky-fibers for sky subtraction, and to perform this step after dohydra extraction.

You will need to edit two parameter files before running dohydra: hydra.params and hydra.dohydra. After giving these, will give some more detailed notes on the process of running dohydra itself.

a. hydra.params - what values to set
b. hydra.doparams - what values to set
c. running dyhydra

a. hydra.params: Below are nominal values for the ‘params’ file for unbinned data using ThAr line-lamp exposures.

Note the two parameters that need to be changed for different binning and different lamps, respectively:

```
ylevel:         0.3     binned 2 x 1
                0.5     unbinned
coordlist:      linelists$thar.dat, or linelists$cuar.dat
```

Also note the parameters that also have changed since earlier versions of this document. These are important changes.

ylevel: (see above)
t_order: 5 -> 15
i_order: 3 -> 1

To determine CCD binning, you can again query the image headers:

```
cl> imhead [image name] l+ | match CCDSUM
```

which will give you:

```
CCDSUM  = ’2 1               ’  /  on chip summation
```

in the case of 2x1 binning.

```
PACKAGE = hydra
   TASK = params

(line   =                INDEF) Default dispersion line
(nsum   =                   10) Number of dispersion lines to sum or median
(order  =           decreasing) Order of apertures
(extras =                   no) Extract sky, sigma, etc.?

                                -- DEFAULT APERTURE LIMITS --
(lower  =                  -5.) Lower aperture limit relative to center
(upper  =                   5.) Upper aperture limit relative to center

                                -- AUTOMATIC APERTURE RESIZING PARAMETERS --
(ylevel =                  0.5) Fraction of peak or intensity for resizing

                                -- TRACE PARAMETERS --
(t_step =                   10) Tracing step
(t_funct=              spline3) Trace fitting function
```

```
   (t_order=                  15) Trace fitting function order
   (t_niter=                   1) Trace rejection iterations
   (t_low  =                  3.) Trace lower rejection sigma
   (t_high =                  3.) Trace upper rejection sigma

                                  -- SCATTERED LIGHT PARAMETERS --
   (buffer =                  0.) Buffer distance from apertures
   (apscat1=                    ) Fitting parameters across the dispersion
   (apscat2=                    ) Fitting parameters along the dispersion

                                  -- APERTURE EXTRACTION PARAMETERS --
   (weights=                none) Extraction weights (none|variance)
   (pfit   =               fit1d) Profile fitting algorithm (fit1d|fit2d)
   (lsigma =                  3.) Lower rejection threshold
   (usigma =                  3.) Upper rejection threshold
   (nsubaps=                   1) Number of subapertures

                                  -- FLAT FIELD FUNCTION FITTING PARAMETERS --
   (f_inter=                 yes) Fit flat field interactively?
   (f_funct=             spline3) Fitting function
   (f_order=                   5) Fitting function order

                                  -- ARC DISPERSION FUNCTION PARAMETERS --
   (thresho=                 10.) Minimum line contrast threshold
   (coordli=             thar.dat) Line list
   (match  =                 -3.) Line list matching limit in Angstroms
   (fwidth =                  4.) Arc line widths in pixels
   (cradius=                 10.) Centering radius in pixels
   (i_funct=             spline3) Coordinate function
   (i_order=                   1) Order of dispersion function
   (i_niter=                   2) Rejection iterations
   (i_low  =                  3.) Lower rejection sigma
   (i_high =                  3.) Upper rejection sigma
   (refit  =                 yes) Refit coordinate function when reidentifying?
   (addfeat=                  no) Add features when reidentifying?

                                  -- AUTOMATIC ARC ASSIGNMENT PARAMETERS --
   (select =               interp) Selection method for reference spectra
   (sort   =                    ) Sort key
   (group  =                    ) Group key
   (time   =                  no) Is sort key a time?
   (timewra=                 17.) Time wrap point for time sorting

                                  -- DISPERSION CORRECTION PARAMETERS --
   (lineari=                  no) Linearize (interpolate) spectra?
   (log    =                  no) Logarithmic wavelength scale?
   (flux   =                 yes) Conserve flux?

                                  -- SKY SUBTRACTION PARAMETERS --
   (combine=             average) Type of combine operation
   (reject =           avsigclip) Sky rejection option
   (scale  =                none) Sky scaling option
   (mode   =                  ql)
```

b. hydra.dohydra. Here are the values for the ‘dohydra’ file:

Note the three parameters that need to be changed for different binning:

	unbinned	bin 2 x 1 (spatial)
width:	6	3
minsep:	5	4
maxsep:	10	6

Note the parameters that also have changed since earlier versions of

this document. These are important changes.

redo: yes -> no

Also note that skyedit and savesky values don't matter if skysubtract=no.

The apidtab can be found at:

http://www.astro.wisc.edu/~mab/research/sparsepak/sparsepak.iraf

and copied into the rdx directory, or specified in the ifupkg directory as:

apidtab = ifupkg$sparsepak.iraf

The latter option requires you have ifupkg defined in your loginuser.cl.

Input list:

pointings.lst - input to dohydra with names from output of crclean

```
PACKAGE = hydra
   TASK = dohydra

objects =       @pointings.lst  List of object spectra
(apref  =         Flat_n1.fits) Aperture reference spectrum
(flat   =         Flat_n1.fits) Flat field spectrum
(through=                     ) Throughput file or image (optional)
(arcs1  =         ThAr_n1.fits) List of arc spectra
(arcs2  =                     ) List of shift arc spectra
(arcrepl=                     ) Special aperture replacements
(arctabl=                     ) Arc assignment table (optional)

(readnoi=                   6.7) Read out noise sigma (photons)
(gain   =                   2.1) Photon gain (photons/data number)
(datamax=                INDEF) Max data value / cosmic ray threshold
(fibers =                    82) Number of fibers
(width  =                    6.) Width of profiles (pixels)
(minsep =                    5.) Minimum separation between fibers (pixels)
(maxsep =                   10.) Maximum separation between fibers (pixels)
(apidtab=       sparsepak.iraf) Aperture identifications
(crval  =                INDEF) Approximate central wavelength
(cdelt  =                INDEF) Approximate dispersion
(objaps =                     ) Object apertures
(skyaps =                     ) Sky apertures
(arcaps =                     ) Arc apertures
(objbeam=                   0,1) Object beam numbers
(skybeam=                     0) Sky beam numbers
(arcbeam=                     ) Arc beam numbers

(scatter=                    no) Subtract scattered light?
(fitflat=                    no) Fit and ratio flat field spectrum?
(clean  =                    no) Detect and replace bad pixels?
(dispcor=                   yes) Dispersion correct spectra?
(savearc=                   yes) Save simultaneous arc apertures?
(skyalig=                    no) Align sky lines?
(skysubt=                    no) Subtract sky?
(skyedit=                   yes) Edit the sky spectra?
(savesky=                   yes) Save sky spectra?
(splot  =                    no) Plot the final spectrum?
(redo   =                    no) Redo operations if previously done?
(update =                   yes) Update spectra if cal data changes?
(batch  =                    no) Extract objects in batch?
(listonl=                    no) List steps but don't process?
```

(params =) Algorithm parameters

c. Running dohydra

Reminders for IF the data is binned: use width=3, minsep=4, maxsep=6 for the profile parameters (dohydra), and ylev=0.3 (params).

Reminders on important flags: In dohydra, set scattered=no, fitflat=no, clean=no, dispcor=yes, savearcs=yes, skyalign=no, skysub=no, splot=no, redo=no, update=yes, batch=no, listonly=no. In the params set of parameters, ensure that coordlist is pointing to the correct line list (ThAr: linelists$thar.dat, etc), and that dohydra will be run without linearization (linearize-).

Basic inputs: Use the combined flat for apref and flat, and the comp lamp for arcs1.

While running dohydra, on the first run, if it asks if you'd like to do something, always say yes, with one exception concerning fitting the wavelength solution individually for all fibers (see below).

You will first be put into a graphical interface which shows you the apertures found. Check that the aperature identification yields all 82 fibers (ordered right to left).

Next, dohydra will fit (trace) the apref images (dome flat). Use an order 15 spline3 function, and force dohydra to use the same fitting function for all the fibers. Note the amplitude of the residuals is tiny (1-3% of a pixel).

The third major step is the hardest: determining the wavelength solution. Note that T2KA has wavelength (lambda) increases with pixel number, while T2KC has lambda decreases with pixel number. Your objective is to identify 3-4 lines spanning the pixel range with values found in the reference library. These form the initial solution, from which additional lines are found, after which a final solution is created.

Therefore, when establishing the dispersion correction you first need a reference atlas for line comparisons. A nice facility exists at:

http://www.noao.edu/kpno/specatlas/

This allows you to create spectra with the bright lines marked and also a table of these lines wavelengths for your choice of wavelength range. (You will need a guess at your central wave and dispersions.) Unfortunately, the relative line-strengths in these spectra are often very different than in the SparsePak spectra, making the identification difficult. We have therefore created a set of plots and reference (wavelength calibrated) spectra for SparsePak setups we often use. These can be found at:

http://www.astro.wisc.edu/~mab/projects/diskmass/analysis/sparsepak/

To mark the initial 3-4 lines, center the cursor in x on the by-hand/eye idenified reference line (zoom in with shift-x, shift-y so you can give a good initial guess for the center in x), and hit 'm'. This marks and centers the line. As per above, mark these 3 or 4 fiducial lines across the FULL range of the spectrum.

Next, type 'f' to fit these 3-4 points. IF AT ALL POSSIBLE FIT WITH AN ORDER 1 SPLINE3 FUNCTION. Only go to higher orders if absolutely necessary. If the lines are marked correctly, you should have a very

low (10^-12 +/- an order of magnitude) rms value. If you don't, hit 'q' to quit the fit, erase all the lines you marked and then redo the marks, and refit with 'f'.

Once you've got a good initial fit, return again to the spectrum-plotting marking screen with'q', and type 'l' to read other lines from the line list and mark lines near peaks in the spectrum. THIS STEP IS CRUCIAL BECAUSE IT READS IN THE HIGH-PRECISION LIST OF LINES.

Hit 'f' to fit again. This time when fitting check to see which lines are outliers by flipping between 'f' and 'q', delete outliers with 'd' either in the spectrum plot or the fitted residuals plot, and delete any "lines" that have been automatically marked that look like they have low S/N or might be on blended lines.

Iterate until you're satisfied with the fit: the RMS should be a tenth to a couple tenths of a pixel.

After fitting the central fiber dohydra will ask you if you want to fit all the fibers individually. HERE IS WHERE YOU SAY NO. You can look at the registered spectrum after the fact to see how well the sky lines line up; if they don't, then you may have to go back and do this step again.

During a final setp, with some of the more recent data, dohydra complains that it doesn't understand the observatory parameter (it's set to WIYN in some of the later data), so set it to kpno on the command line as the prompt arises.

At the end, you should have an *.ms.fits file where, for example, row one is the spectrum extracted for fiber 1, and * = [obj]_[group].[band], as defined in Section II above, and as you have defined in the pointings.lst file.

8. dispcor - apply dispersion solution, rectify and resample data

In this step you will apply disperion solutions, rectify and resample the data with dispcor; and then create mask files with specreg_mask.

a. dispcor

You will want to run the hydra.dispcor propram *TWICE* on all ms-file output from dohydra, once to create rectified and linear-resampled spectra, and also for rectified log-resampled spectra.

The basic dispcor parameters:

```
PACKAGE = hydra
   TASK = dispcor

        input = ""              List of input spectra
       output = ""              List of output spectra
   (linearize = yes)            Linearize (interpolate) spectra?
    (database = "database")     Dispersion solution database
       (table = "")             Wavelength table for apertures
          (w1 = 4990.5)         Starting wavelength
          (w2 = INDEF)          Ending wavelength
          (dw = 0.129)          Wavelength interval per pixel
          (nw = 2002)           Number of output pixels
         (log = no)             Logarithmic wavelength scale?
        (flux = yes)            Conserve flux?
       (blank = 0.)             Output value of points not in input
```

```
      (samedisp = yes)           Same dispersion in all apertures?
        (global = no)            Apply global defaults?
     (ignoreaps = no)            Ignore apertures?
       (confirm = no)            Confirm dispersion coordinates?
      (listonly = no)            List the dispersion coordinates only?
       (verbose = yes)           Print linear dispersion assignments?
       (logfile = "")            Log file
```

The main things to change are

(i) w1,w2,dw,nw

depending your setup. Values for standard Diskmass and related setups:

Echelle:

```
        Mg o11:      w1=4975. w2=INDEF dw=0.129 nw=2250
        Ca o6 :      w1=8365. w2=INDEF dw=0.281 nw=2090
        Ca o7 :      w1=8435. w2=INDEF dw=0.144 nw=2125
        Ha o8 :      w1=6460. w2=INDEF dw=0.201 nw=2200
```

VPH 740

```
        a = 22.5     w1=4575. w2=INDEF dw=0.528 nw=2225
```

(ii) log, samedisp

depending on your sampling. To rectify the data, which you want to do in both cases, samedisp=yes. The linearize+ parameter also is always set this way.

For log-resampling, log+, not, log-. Then the wavelength parameters set the wavelength solution.

Input files:

```
        *.ms.fits - Output from dohydra (not linearized or sky subtracted)
```

Output files:

```
        *.ms_log.fits - log-rectified (resampled) ms file
        *.ms_lin.fits - linear-rectified (resampled) ms file
```

where * = [obj]_[group].[band], as defined in Section II above, and as you have defined in the pointings.lst file.

b. specreg_mask

The specreg_mask program creates a mask where data is not defined for all pixels (beyond ends of spectrum). The program is in the ifupkg and is a C++ program that can be called within IRAF.

```
        specreg_mask -h

        Command line arguments (spaces must be as shown):
                 -R[file]: Raw, non-linearized ms image (use '@' for batch)
                 -L[file]: Linearized ms image: log or lin (use '@' for batch)
                 -M[file]: Output mask image (use '@' for batch)
                 -log:     Input linearized ms image is log-linear in wavelength
                 -h:       Print this listing
```

The call will look like this:

```
        specreg_mask -R@dispcor_input.lst -L@dispcor_lin.lst -M@dispcor_m_lin.lst
```

for the linearized data and

```
        specreg_mask -R@dispcor_input.lst -L@dispcor_log.lst -M@dispcor_m_log.lst -log
```

for the log-linearized data.

NOTE THAT WHEN USING T2KC or STA1 DATA w/ the dispersion flipped,
you'll need to add the flip-flag:

```
        specreg_mask -R@dispcor_input.lst -L@dispcor_lin.lst -M@dispcor_m_lin.lst -fli
p
```

etc.

The list files dispcor_input.lst, dispcor_lin.lst, dispcor_log.lst
should exist from the previous step with dispcor. The
dispcor_m_lin.lst and dispcor_m_log.lst files should be generated
using the convention *.ms_m_lin.fits and *.ms_m_log.fits defined
above.

9. error calculation

NOTE: Performing the error calculation is not necessary for performing the
sky-subtraction; however, calculation of the error in the sky-subtracted
spectra requires having first produced the error on the unsubtracted data.

You will need to used proprietary programs. Define the following
in your loginuser.cl in your IRAF home directory (~/iraf/):

```
        reset ifupkg    = "~mab/iraf/scripts/ifupkg/"
        task $ifupkg    = "ifupkg$ifupkg.cl"
```

and load the ifupkg.

a. Create the raw (unextracted) error (sigma) images using ifupkg.rawimerr.

The possible rawimerr command line arguments are (spaces must be as shown):

```
        -I[file]:  Input image
        -o[num]:   Number of pixels used in overscan correction (def: 30)
        -b[num]:   Number of bias frames used for bias level (def: 10)
  *     -E[file]:  Empirical relation between (1) DN value and (2) error
        -N[file]:  Image with number of images combined for each pixel
        -nim[num]: Number of images used to create input image (def: 1)
  *     -sum:      Images produced from sums of others (def: mean)
        -g[name]:  Header keyword name for the gain (def: GAIN)
        -rn[name]: Header keyword name for the read noise (def: RDNOISE)
        -num:      Read provided gain and readnoise as numbers (def: keywords)
        -h:        Print options
```

You will need to verify the number of overscan pixels used (look at
the image header with imhead), the number of bias frames that went
into your zero image, and count the number of object images that were
combined in the CR cleaning stage. These values, along with the
explicit gain and readnoise should be given.

A typical command will be:

```
        cl> rawimerr -IU01771_1.mg.fits -o30 -b10 -nim1 -g2.1 -rn6.7 -num
```

Most often you will be running the program on a single image at a time.

In the above example, rawimerr calculates the error on the image
U01771_1.mg.fits using an overscan region of 30 pixels, 10 bias

frames, one images were combined to make the image, uses a gain of 2.1, and a readnoise of 6.7. The resulting image is U01771_1.mg.sig.fits.

You will very likely never have to use the two starred (*) options, but they allow the user to provide their own empirical relation between the pixel value and the associated error (-E[file]) and to change the error to be calculated for an image created from a sum of a set of images instead of the average (-sum).

Other Examples:

1. To get a list of the command line options, type:

```
> rawimerr -h
```

2. If you want to change the number of overscan pixels, the number of bias frames, or the number of images used in the combination, an example call would be:

```
> rawimerr -IU01771_1.mg.fits -o35 -b9 -nim2 -g2.1 -rn6.7 -num
```

3. If you want the program to read the readnoise and/or gain from the header, an example would be:

```
> rawimerr -IU01771_1.mg.fits -gGAIN_12 -rnNOISE_12
```

where the header keywords with the gain and read noise are GAIN_12 and NOISE_12, respectively. Note the default is to read the keywords GAIN and RDNOISE from the header.

4. If you have a mask that gives the number pixels averaged together to create the image, an example would be:

```
> rawimerr -IU01771_1.mg.fits -Nmask_U01771_1.mg.fits
```

5. If you have a list of images THAT WERE CREATED FROM THE SAME NUMBER OF AVERAGED IMAGES, you can process them all simultaneously using a list file. An example would be:

```
> rawimerr -I@image.lst -N@image_mask.lst -nim3
```

b. Create the extracted error images using the IRAF task ifupkg.mkmes:

The parameters for mkmes are:

```
        PACKAGE = ifupkg
           TASK = mkmes

        imlst = "@pointings.lst" Input image list
       apflat = "Flat_n1.fits"  Input aperture reference and flat image
      (linlst = lin_n1.lst      Linearized object spectra
     (displog = no)             Logarithmic wavelength scale?
   (keepsteps = no)             Keep intermediate calculations?
        (redo = no)             Recalculate *.me.fits file if already exists?
```

The input list of images (a template file) is the same that you entered into dohydra (NOT ms files). You can either input them all as above, or input them individually. The apflat image is also the input flat-field image for dohydra and that was used to trace, define the extraction, and flatten the spectra (in the above example, the input flat-field and apref images is Flat_n1.fits). The linlst is the template file of the linearized or log-linearized ms files produced in step-8 above with dispcor. The dispersion parameters for the output linearized or log-linearized error spectra are read from the header of

the images provided. In the above example, the linear-rectified images are used.

In the example above, the program searches for and must find the *.ms.fits and *.sig.fits counterparts to the images in the pointings.lst file. For example, if the pointings.lst file contains U01771_1.mg.fits, the U01771_1.mg.ms.fits and U01771_1.mg.sig.fits files must be found in the same directory, or the script will fail.

You'll need to run mkmes twice, once with displog- and once with displog+. Use displog=no for linear-rectified images (counterparts to *.ms_lin.fits images) and displog=yes for log-rectified images (counterparts to *.ms_log.fits images). The first time will create the *.me.fits and one of either the *.me_lin.fits or *.me_log.fits files. If redo=no, the second time you run mkmes it will not recreate *.me.fits, but simply use the existing one to create the other set of rectified images.

**NOTE: THE HYDRA PACKAGE MUST BE LOADED, AND IT IS IMPORTANT THAT THE DOHYDRA AND PARAMS PARAMETERS BE THE SAME AS THEY WERE WHEN EXTRACTING THE OBSERVED SPECTRA! This is why they need to be written to the Notes.txt file.

10. sky-subtraction

You will again need to used proprietary programs. Define the following in your loginuser.cl in your IRAF home directory (~/iraf/):

```
reset ifupkg    = "~mab/iraf/scripts/ifupkg/"
task $ifupkg    = "ifupkg$ifupkg.cl"
```

and load the ifupkg if you haven't already.

a. Edit the IRAF function skysub, in the hydra package, and set these parameters:

```
PACKAGE = hydra
   TASK = skysub

        input = ""              Input spectra to sky subtract
     (output = "")              Output sky subtracted spectra
     (objaps = "")              Object apertures
     (skyaps = "")              Sky apertures
   (objbeams = "0,1")           Object beam numbers
   (skybeams = "0")             Sky beam numbers
    (skyedit = yes)             Edit the sky spectra?
    (combine = "average")       Combining option
     (reject = "avsigclip")     Sky rejection option
      (scale = "none")          Sky scaling option
   (saveskys = yes)             Save sky spectra?
    (logfile = "logfile")       Logfile
```

b. Perform both the sky-subtraction and the error calculation of the sky-subtracted spectra using the IRAF task subsky_err (which will call skysub). The parameters for skysub_err are:

```
PACKAGE = ifupkg
   TASK = skysub_err

        mslst = "@skysub.lst"   Input list of ms images
       omslst = "@skysub_out.lst" Output list of sky-subtracted images
       (melst = "@skysub_me.lst") Input list of me images
       (sslog = "logfile")      Logfile for sky-subtraction
```

where skysub.lst is a file listing both *.ms_lin.fits and *.ms_log.fits files; skysub_out.lst contains that same file names but altered according to our naming convention, i.e., *.ms_s_lin.fits and *.ms_s_log.fits; and the skysub_me.lst file contains the list of associated *.me_lin.fits and *.me_log.fits files. If no melst file is entered, the script performs the sky-subtraction without the error calculation. The error calculation produces files of the form *.me_s_lin.fits and *.me_s_slog.fits. The program complains if you try to run the subsky_err twice with the melst defined, since it will fail to be able to overwrite these files.

NOTE: The entered files must have been resampled such that the wavelength solution is identical for all fibers.

When you execute the program, an IRAF window will display all the sky spectra for you to edit. Look at the group of sky spectra to see if any have excess flux, which indicates the fiber was actually on the target galaxy or some random other object. If you find a fiber with excess flux, use the ’d’ key to delete it. Immediately after you delete a spectrum, redraw the list of spectra with the ’r’ key. If you’ve mistakenly deleted the wrong spectrum, you can regain ONLY THE MOST PREVIOUSLY DELETED SPECTUM with the ’e’ key.

Once all the spectra have nearly identical flux, hit the ’q’ key, and then the program will ask you about the type of rejection. Continue with the ’avsigclip’ algorithm by just hitting the carriage return.

You’ll follow the same procedure for each spectrum in your input list.

NOTE: Because you’ll be sky-subtracting the *.ms_lin.fits and *.ms_log.fits separately, be sure to combine the same set of sky spectra for both (i.e., delete the same sky spectra from both). You can check this by looking at the logfile; search for calls to SCOMBINE and look at the list of fibers used to make the sky spectrum. Each image should now have a sky spectrum, which is saved as sky*.ms_s_log.fits, etc.

You are done!

EOF