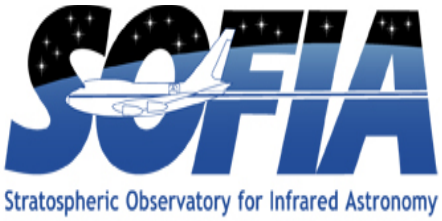

FORCAST Redux Users Manual

SCI-US-HBK-OP10-2003

Date: December 30, 2013
Revision: -

DFRC
Dryden Flight Research Center
Edwards, CA 93523

ARC
Ames Research Center
Moffett Field, CA 94035

German Space Agency, DLR
Deutsches Zentrum für Luft und Raumfahrt

FORCAST Redux Users Manual

SCI-US-HBK-OP10-2003

AUTHOR:

William Vacca, USRA, SOFIA Senior Scientist	Date
Melanie Clarke, USRA, SOFIA Pipeline Engineer	Date

CONCURRENCE:

R.Y. Shuping, USRA, SOFIA DPS Lead	Date
William Vacca, USRA, SOFIA Senior Scientist	Date

APPROVAL:

William Reach, USRA, SOFIA Associate Director for Science	Date
Helen J. Hall, USRA, SOFIA Associate Director for Program Management	Date

REVISION HISTORY

REV	DATE	DESCRIPTION
-	12/30/13	Initial Release

FORCAST Redux Users Manual
SCI-US-HBK-OP10-2003

Table of Contents

FORCAST Redux Users Manual
SCI-US-HBK-OP10-2003

1. INTRODUCTION

The SI Pipeline Users Manual (OP10) is intended for use by both SOFIA Science Center staff during routine data processing and analysis, and also as a reference for General Investigators (GIs) and archive users to understand how the data in which they are interested was processed. This manual is intended to provide all the needed information to execute the SI Level 2 Pipeline, flux calibrate the results, and assess the data quality of the resulting products. It will also provide a description of the algorithms used by the pipeline and both the final and intermediate data products.

A description of the current pipeline capabilities, testing results, known issues, and installation procedures are documented in the SI Pipeline Software Version Description Document (SVDD, SW06, DOCREF). The overall Verification and Validation (V&V) approach can be found in the Data Processing System V&V Plan (SV01-2232). Both documents can be obtained from the SOFIA document library in Windchill at location: / Software Management Development or Verification / Pipelines (DPS).

2. SI OBSERVING MODES SUPPORTED

2.1. FORCAST observing techniques

Because the sky is so bright in the mid-infrared (MIR) relative to astronomical sources, the way in which observations are made in the MIR is considerably different from the more familiar way they are made in the optical. Any raw image of a region in the MIR is overwhelmed by this sky "background" emission. The situation is similar to trying to observe in the optical during the day. The bright daylight sky swamps the detector and makes it impossible to see astronomical sources in the raw images.

In order to remove the background from the MIR image and detect the faint astronomical sources, observations of another region (free of sources) are made and the two images are subtracted. However, the MIR is highly variable, both spatially and – more importantly – temporally. It would take far too long (on the order of seconds) to reposition a large telescope to observe this "sky background" region: by the time the telescope had moved and settled at the new location, the sky background level would have changed so much that the subtraction of the two images would be useless. In order to avoid this problem, the secondary mirror (which is considerably smaller than the primary mirror) of the telescope is tilted, rather than moving the entire telescope. This allows observers to look at two different sky positions very quickly (on the order of a few to ten times per second), because tilting the secondary by an angle θ moves the center of the field imaged by the detector by θ on the sky. Tilting the secondary between two positions is known as "chopping". FORCAST observations are typically made with a chopping

frequency of 4 Hz. That is, every 0.25 sec, the secondary is moved between the two observing positions.

Chopping can be done either symmetrically or asymmetrically. Symmetric chopping means that the secondary mirror is tilted symmetrically about the telescope optical axis (also known as the boresight) in the two chop positions. The distance between the two chop positions is known as the chop throw. The distance between the boresight and either chop position is known as the chop amplitude and is equal to half the chop throw (see Figure 1).

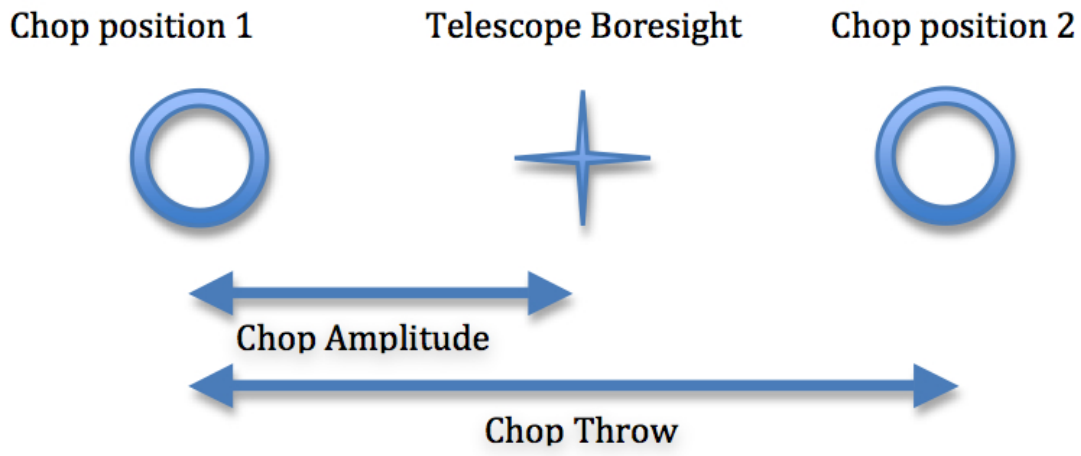

Figure 1: Symmetric Chop

Asymmetric chopping means that the secondary is aligned with the telescope boresight in one position, but is tilted away from the boresight in the chop position. The chop amplitude is equal to the chop throw in this case (see Figure 2).

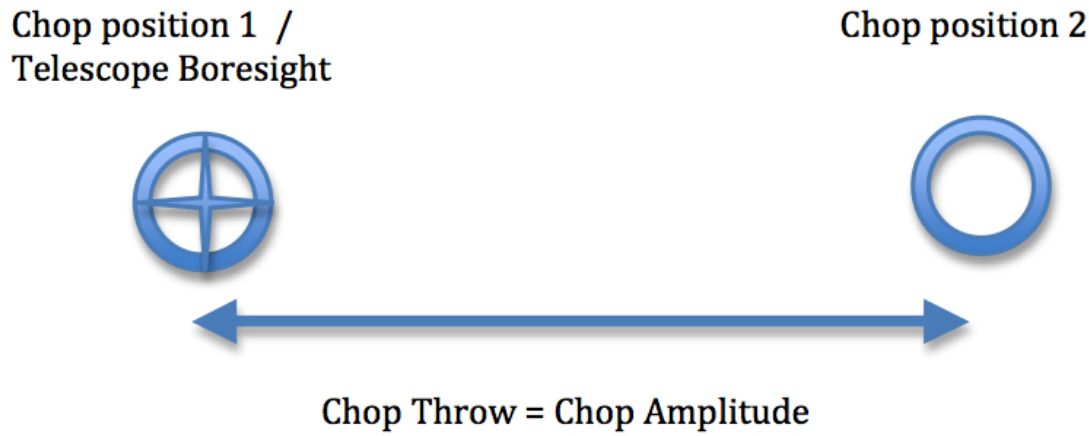

Figure 2: Asymmetric Chop

Unfortunately, moving the secondary mirror causes the telescope to be slightly misaligned, which introduces optical distortions (notably the optical aberration known as coma) and additional background emission from the telescope (considerably smaller than the sky emission

but present nonetheless) in the images. The optical distortions can be minimized by tilting the secondary only tiny fractions of a degree. The additional telescopic background can be removed by moving the entire telescope to a new position and then chopping the secondary again between two positions. Subtracting the two chop images at this new telescope position will remove the sky emission but leave the additional telescopic background due to the misalignment; subtracting the result from the chop-subtracted image at the first telescope position will then remove the background. Since the process of moving to a new position is needed to remove the additional background from the telescope, not the sky, it can be done on a much longer timescale. The variation in the telescopic backgrounds occurs on timescales on the order of tens of seconds to minutes, much slower than the variation in the sky emission.

This movement of the entire telescope, on a much longer timescale than chopping, is known as nodding. The two nod positions are usually referred to as nod A and nod B. The distance between the two nod positions is known as the nod throw or the nod amplitude. For FORCAST observations, nods are done every 5 to 30 seconds. The chop-subtracted images at nod position B are then subtracted from the chop-- subtracted images at nod position A. The result will be an image of the region, without the sky background emission or the additional emission resulting from tilting the secondary during the chopping process. The sequence of chopping in one telescope position, nodding, and chopping again in a second position is known as a chop/nod cycle.

Again, because the MIR sky is so bright, deep images of a region cannot be obtained (as they are in the optical) by simply observing the region for a long time with the detector collecting photons continuously. As stated above, the observations require chopping and nodding at fairly frequent intervals. Hence, deep observations are made by “stacking” a series of chop/nod images. Furthermore, MIR detectors are not perfect, and often have bad pixels or flaws. In order to avoid these defects on the arrays, and prevent them from marring the final images, observers employ a technique known as “dithering”. Dithering entails moving the position of the telescope slightly with respect to the center of the region observed each time a new chop/nod cycle is begun, or after several chop/nod cycles. When the images are processed, the observed region will appear in a slightly different place on the detector. This means that the bad pixels do not appear in the same place relative to the observed region. The individual images can then be registered and averaged or median-combined, a process that will eliminate (in theory) the bad pixels from the final image.

2.2. Available chopping modes

2.2.1. Symmetric chopping modes: C2N and C2ND

FORCAST acquires astronomical observations in two symmetric chopping modes: two-position chopping with no nodding (C2) and two-position chopping with nodding (C2N). Dithering can be implemented for either mode; two-position chopping with nodding and dithering is referred to as C2ND. The most common observing methods used are C2N and C2ND. C2ND is conceptually very similar the C2N mode: the only difference is a slight movement of the telescope position after each chop/nod cycle.

FORCAST can make two types of C2N observations: Nod Match Chop (NMC) and Nod Perp Chop (NPC). The positions of the telescope boresight, the two chop positions, and the two nod positions for these observing types are shown below (Figures 3 and 4).

2.2.2. C2N: Nod Match Chop (NMC)

In this case, the telescope is pointed at a position half of the chop throw distance away from the object to be observed, and the secondary chops between two positions, one of which is centered on the object. The nod throw has the same magnitude as the chop throw, and is in a direction exactly 180 degrees from that of the chop direction. The final image generated by subtracting the images obtained for the two chop positions at nod A and those at nod B, and then subtracting the results. This will produce three images of the star, one positive and two negative, with the positive being twice as bright as the negatives.

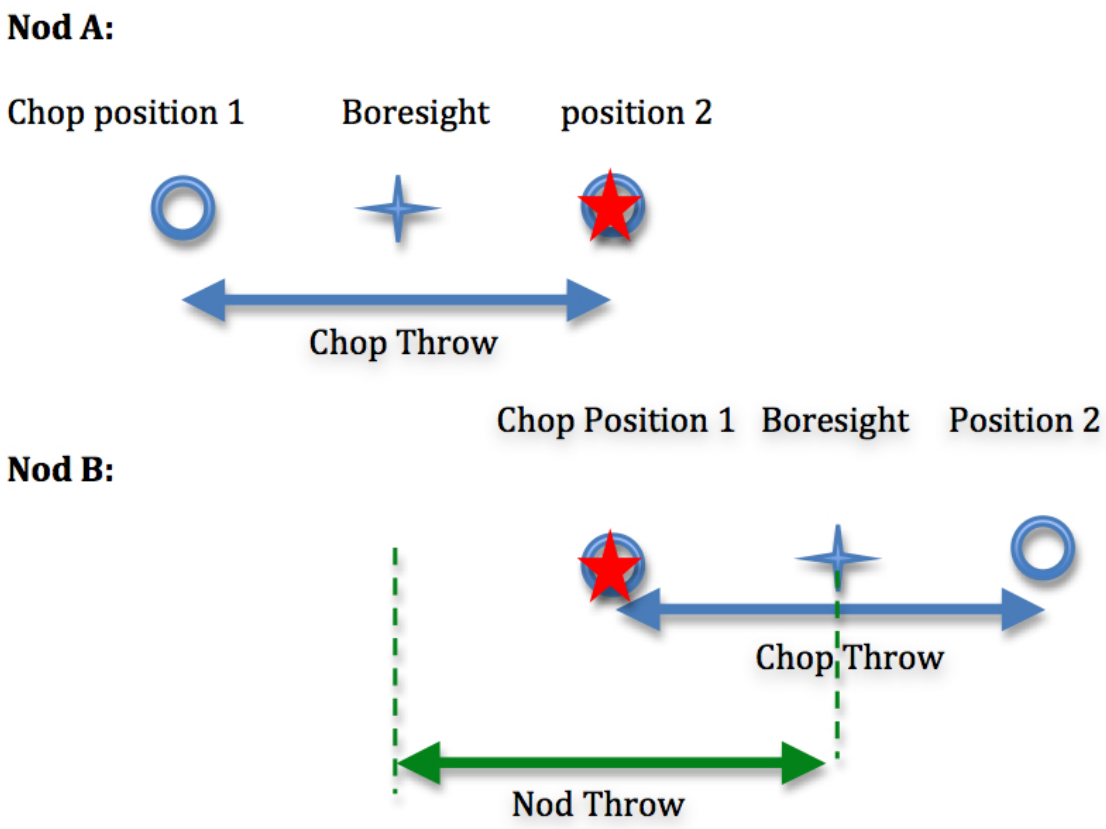

Figure 3: Nod Match Chop mode

2.2.3. C2N: Nod Perp Chop (NPC)

In this case, the telescope is offset by half the nod throw from the target in a direction perpendicular to the chop direction, and the secondary chops between two positions. The nod throw usually (but not necessarily) has the same magnitude as the chop, but it is in a direction perpendicular to the chop direction. The final image is generated by subtracting the images obtained for the two chop positions at nod A and those at nod B, and then subtracting the results.

This will produce four images of the star in a rectangular pattern, with the image values alternating positive and negative.

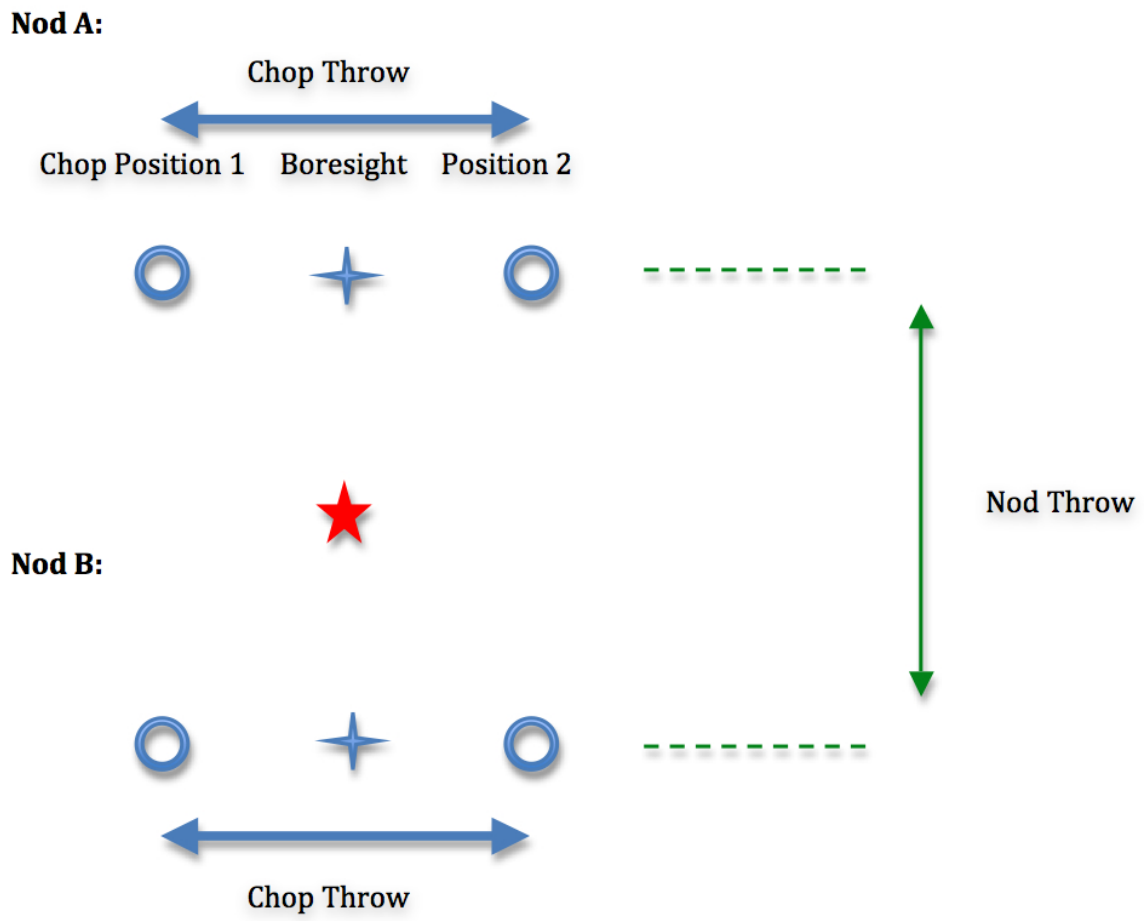

Figure 4: Nod Perp Chop mode

2.2.4. Asymmetrical chopping mode: C2NC2

FORCAST also has an asymmetrical chop mode, known as C2NC2. In this mode, the telescope is first pointed at the target (position A). In this first position, the secondary is aligned with the boresight for one observation and then is tilted some amount (often 180-480 arcseconds) for the second (asymmetrically chopped) observation. This is an asymmetric C2 mode observation. The telescope is then slewed some distance from the target, to some sky region without sources (position B), and the asymmetric chop pattern is repeated. C2NC2 observations are taken as a series of 8 (C2) files in the sequence A B A A B A A B. The time between slews is typically 30 seconds.

2.2.5. Nod not related to Chop, Asymmetric Chop: NXCAC (Grism only)

This replaces C2NC2 mode when the GI wants to use C2NC2 mode with grisms **only**. This is ABBA, like C2N mode (not ABA, like C2NC2). Again, nods are packaged together, so data from this mode will reduce just like C2N mode. The reason for adding this mode stems from the need to define our large chops and nods in ERF, and dither in SIRF along the slit.

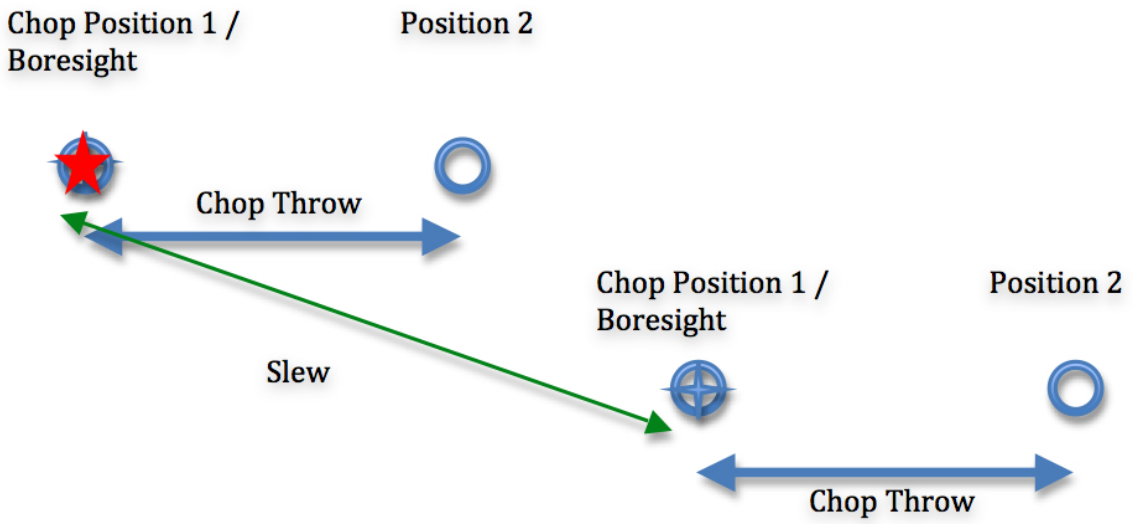

Figure 5: C2NC2 mode

2.3. Calibration files

Calibration files for FORCAST may include dark frames and flatfield frames. FORCAST flat field images are either images of blank sky or of the on-board calibration source, which is a hot plate imaged onto the camera pupil. Dark frames may be used to correct the dark current in a flat field, but are not necessary for science frames, since the dark current will be removed when chop/nod subtraction is performed. It is not expected that flats and darks will change significantly over short time scales, so the FORCAST SI team will provide standard flats and darks to be used in every reduction. These will be distributed along with the reduction software code.

3. ALGORITHM DESCRIPTION

3.1. Overview of data reduction steps

Redux applies a number of corrections to each input file, regardless of the chop/nod mode used to take the data. The steps used for imaging and grism modes are nearly identical; points where the results or the procedure differ for either mode are noted below. After preprocessing, individual images or spectra of a source must be combined to produce the final data product. This procedure depends strongly on the chop/nod mode.

All raw files are first processed as images, with algorithms developed for the DRIP reduction package. Spectroscopy files then undergo spectral extraction and combination of the resulting one-dimensional spectra, using algorithms from the FSpextool reduction package.

See Figure 6 for a flowchart of all processing steps used by the pipeline.

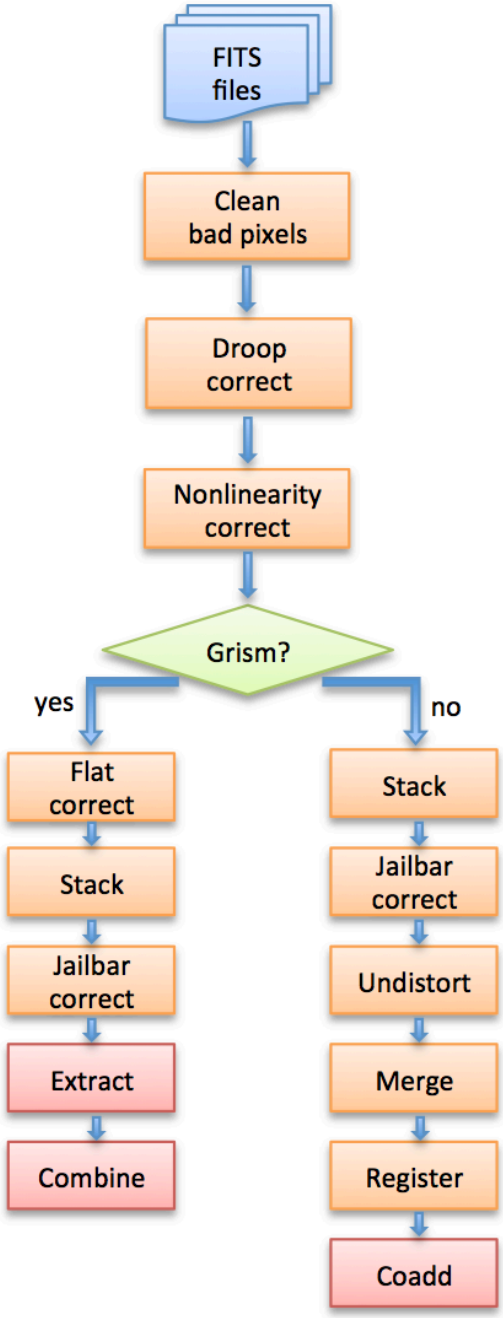

Figure 6: Processing steps for imaging and grism data. Boxes shown in red are algorithms that come from the FSpextool package; all others come from the DRIP package.

3.2. Reduction algorithms

The following subsections detail each of the data reduction pipeline steps:

- Cleaning of bad pixels
- Droop effect correction
- Nonlinearity correction
- Flatfield correction
- Background subtraction (chop/nod stacking)
- Jailbar removal (crosstalk correction)
- Optical distortion correction
- Image shift and rotation (imaging only)
- Spectral extraction (grism only)
- Image registration (imaging only)
- Coadding multiple observations

3.2.1. Clean

Bad pixels in the FORCAST arrays take the form of hot pixels (with extreme dark current) or pixels with very different response (usually much lower) than the surrounding pixels. The pipeline minimizes the effects of bad pixels by using a bad pixel mask to identify their locations and then replacing the bad pixels with values derived from the surrounding operational pixels. The DRIP clean function (`drip_clean.pro`) is built around the IDL procedure `MASKINTERP`, written by J. Harrington, which fits a 2-dimentional surface to an aperture in the image centered on the bad pixel(s) while ignoring the bad pixel(s) identified in the mask. `MASKINTERP` then replaces the bad pixels with the corresponding values of the surface fit. `MASKINTERP` is set to use a planar surface with an aperture radius of 6 pixels. These parameters can be changed quickly by altering the options in the `MASKINTERP` function call located in the clean function.

The bad pixel map for both FORCAST channels (`swc_badpix.fits` and `lwc_badpix.fits`) is currently produced manually, independent of the pipeline. The mask is a 256x256 image with pixel value = 0 for bad pixels and pixel value = 1 otherwise.

3.2.2. Droop correction

The FORCAST arrays and readout electronics exhibit a linear response offset caused by the presence of a signal on the array. This effect is called ‘droop’ since the result is a reduced signal. Droop results in each pixel having a reduced signal that is proportional to the total signal in the 15 other pixels in the row read from the multiplexer simultaneously with that pixel. The effect, illustrated in Figure 7, is images with periodic spurious sources spread across the array rows. The droop correction removes the droop offset by multiplying each pixel by a value derived from the sum of every 16th pixel in the same row all multiplied by an empirically determined offset fraction: droopfrac = 0.0035. The pipeline gets droopfrac from the dripconf.txt configuration file; the droop correction algorithm can be found in the `drip_droop.pro` function.

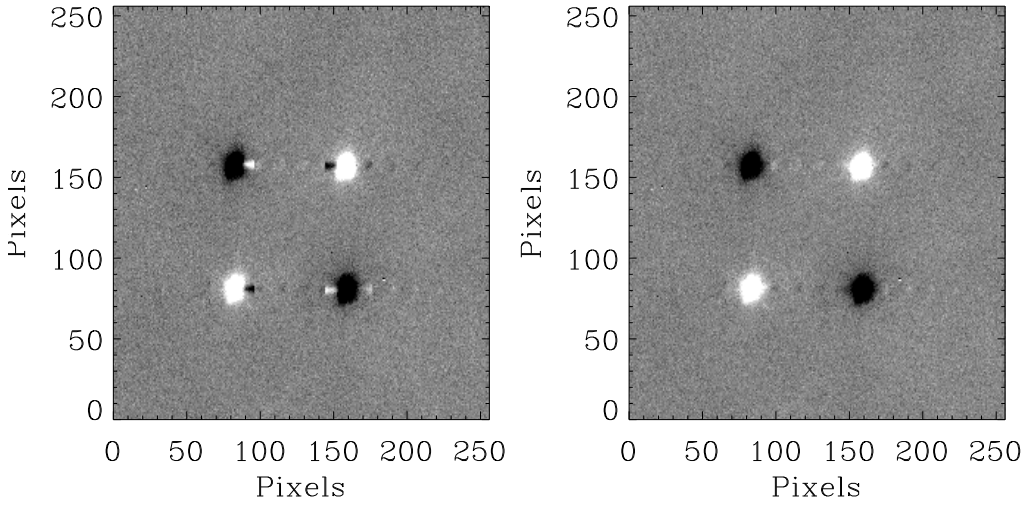

Figure 7: Background-subtracted FORCAST images of a star with droop (left) and with the droop correction applied (right).

3.2.3. Nonlinearity correction

In principle, the response of each of the pixels in our detector arrays should be linear with incident flux. In practice, the degree to which detector linearity depends on the level of charge in the wells relative to the saturation level. Empirical tests optimizing signal-to-noise indicate that signal levels in the neighborhood of 60% of full well for a given detector capacitance in the FORCAST arrays have minimal departures from linear response and optimal signal-to-noise. For a given background level we can keep signal levels near optimal by adjusting the detector readout frame rate and detector capacitance. Since keeping signals near 60% of saturation level is not always possible or practical, we have measured response curves (response in analog-to-digital units (ADU) as a function of well depth for varying background levels) that yield linearity correction factors. These multiplicative correction factors linearize the response for a much larger range of well depths (~15% – 90% of saturation). The linearity correction is applied globally to FORCAST images prior to background subtraction. The pipeline first calculates the background level for a sub-image defined in the `dripconf.txt` configuration file (`drip_background.pro`). This level is then used to look up the linearity correction factor (also located in `dripconf.txt`) and the pipeline applies the correction factor to the entire image. The code for this correction is in `drip_imgnonlin.pro`.

This global-level correction should remove non-linearity effects to first order, but it is also possible that non-linearity effects may vary from one pixel to another. The DRIP package includes an algorithm for applying a non-linearity correction individually to each pixel (`drip_nonlin.pro`). However, these corrections have not yet been determined for the FORCAST array, so the pixel-to-pixel non-linearity correction is not currently applied.

3.2.4. Flatfield correction

The flatfield correction should, in principle, remove pixel-to-pixel variations in gain, dark current, and responsivity and also variations in illumination across the detector array due to uneven illumination or image distortion. The pipeline (`drip_flatsum.pro`) produces a

master flat image from raw FORCAST flatfields, a single image plane containing linear offsets. The offsets are calculated by dividing the image (pixel by pixel) by the median pixel value of the image. Thus a perfectly flat image would have a master flat containing values of 1.0. The flatfield correction (`drip_flat.pro`) divides raw image frames by the master flat image.

The FORCAST data reduction pipeline produces a master flat image from raw flat files differently depending on the number of frames in the raw flatfield data cube. A cube of four image planes is assumed to be composed of a hot blackbody source (two frames) and a cool blackbody source (two frames) which are differenced when generating the master flatfield image. This is the default format produced by invoking the 'FLAT' button/function in the FORCAST data acquisition software. Any other number of frames (including a single frame) are assumed to be composed of identical blackbody sources and are averaged when generating the master flatfield image.

In the mid-infrared, it is often difficult to produce a flat field frame that improves photometric precision, rather than worsens it. The imaging flats presently available for FORCAST do not improve image quality, so flat field correction is currently disabled for the imaging mode of the pipeline.

For spectroscopic modes, the flatfield may be normalized using a 300K blackbody continuum and is applied via the `drip_flat` function.

3.2.5. Background subtraction (chop/nod stacking)

Background subtraction is accomplished by subtracting chopped image pairs and then subtracting nodded image pairs. This happens in the `drip_stack.pro` function. For C2N/NPC imaging mode with chop/nod on-chip (ie. chop throws smaller than the FORCAST field of view), the four chop/nod images in the raw data file are reduced to a single stacked image frame with a pattern of four background-subtracted images of the source, two of them negative. For chop/nod larger than the FORCAST field of view the raw files are reduced to a single frame with one background-subtracted image of the source. For the C2N/NPC spectroscopic mode, either the chop or the nod is always off the slit, so there will be two traces in the subtracted image: one positive and one negative. If the chop or nod throw is larger than the field of view, there will be a single trace in the image.

In the case of the C2N/NMC mode for either imaging or spectroscopy, the nod direction is the same as the chop direction with the same throw so that the subtracted image frame contains three background-subtracted images of the source. The central image or trace is positive and the two outlying images are negative. If the chop/nod throw is larger than the FORCAST field of view, there will be a single image or trace in the image.

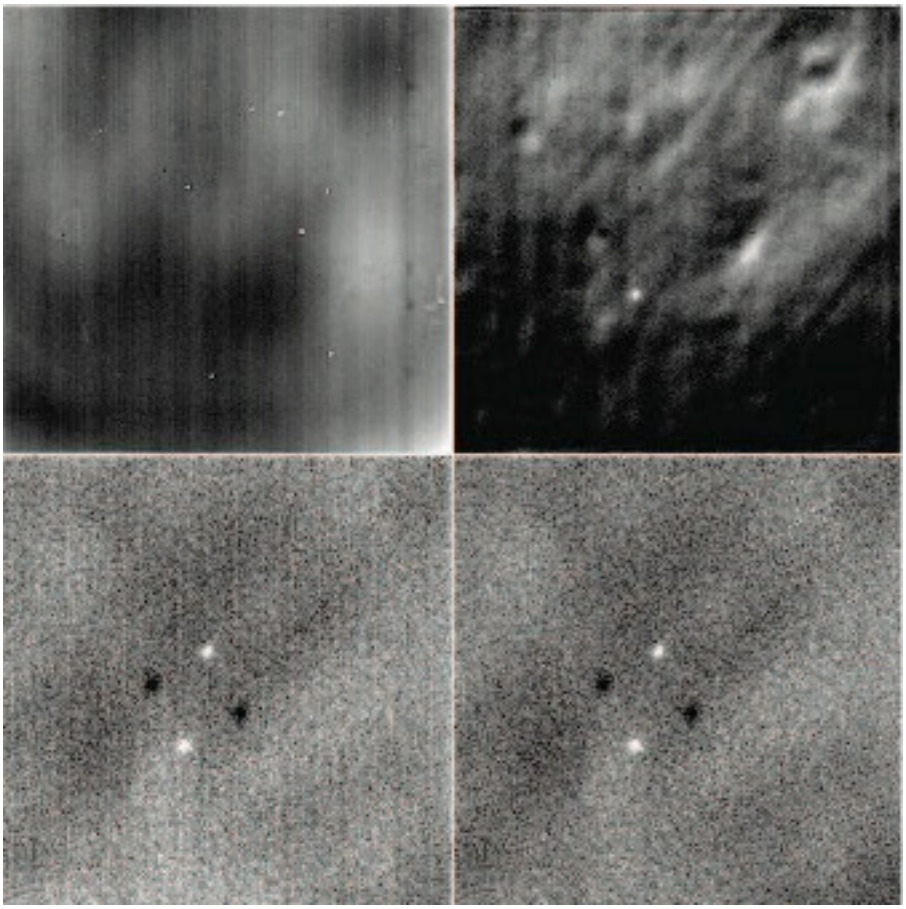

Figure 8. Images at four stages of background subtraction: raw frames (upper left), chop-subtracted (upper right), chop/nod-subtracted for a single nod sequence (nod positions AB, lower left), and chop/nod-subtracted for a full nod cycle (nod positions ABBA, lower right).

C2NC2 raw data sets for imaging or spectroscopy consist of a set of 5 FITS files, each with 4 image planes containing the chop pairs for both the on-source position (position A) and the blank sky position (position B). The four planes can be reduced in the same manner as any C2N image, by first subtracting chopped image pairs for both and then subtracting nodded image pairs. The nod sequence for C2NC2 is $A_1B_1A_2A_3B_2A_4A_5B_3$, where the off-source B nods are **shared** between some of the files (shared B beams shown in bold):

File 1 = $A_1\mathbf{B_1}$
File 2 = $\mathbf{B_1}A_2$
File 3 = $A_3\mathbf{B_2}$
File 4 = $\mathbf{B_2}A_4$
File 5 = A_5B_3

At this point, the background in the chop/nod-subtracted stack should be zero, but if there is a slight mismatch between the background level in the individual frames, there may still remain some small residual background level. After stacking, the pipeline estimates this residual

background by taking the mode of the image data in a central section of the image, then subtracts this level from the stacked image.

The last step in the imaging stack pipeline step is to convert pixel data from analog-to-digital units (ADU) per frame to mega-electrons per second (Me^-/s) using the gain and frame rate used for the observation.

For grism data, this conversion is applied as well. Then, individual frames taken at the same dither position may be combined together to increase the signal-to-noise in the two-dimensional spectral image. This aids in the extraction of faint sources, but is not necessary if the source is bright. This combination will generally be done for science targets, but not for telluric standards.

3.2.6. Jailbar removal (Crosstalk correction)

The FORCAST array readout circuitry has a residual, or latent, signal that persists when pixels have high contrast relative to the surrounding pixels. This can occur for bad pixels or for bright point sources. This residual is present not only in the affected pixels, but is correlated between all pixels read by the same one of sixteen multiplexer channels. This results in a linear pattern of bars (every 16 pixels the pattern repeats) known as "jailbars" in the background-subtracted (stacked) images. Jailbars can interfere with subsequent efforts to register multiple images since the pattern can dominate the cross-correlation algorithm used in image registration. The jailbars can also interfere with photometry in images and with spectral flux in spectroscopy frames. Particularly troublesome is the fact that the jailbar intensity appears to depend on the intensity of signal on the array so that "flatfielding" it out will not work.

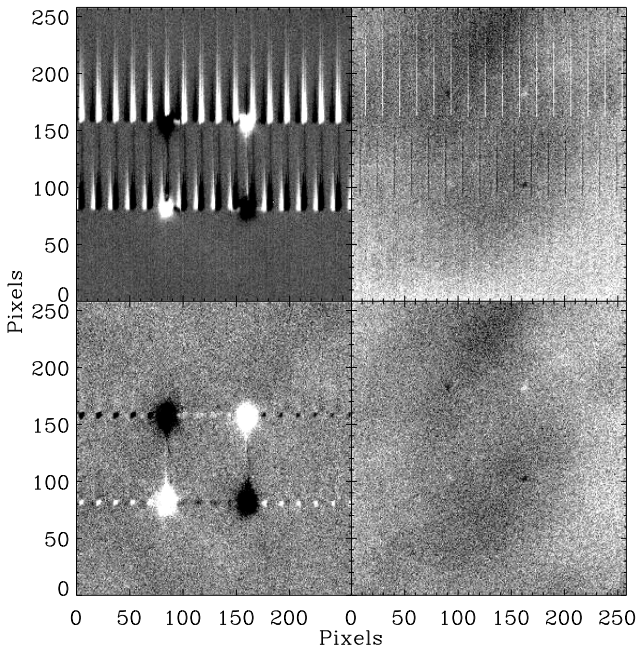

Figure 9. Cross talk correction for a bright point source on left, and faint source on right. Images on the top are before correction; images on the bottom are after correction.

The pipeline attempts to remove jailbar patterns from the background-subtracted images by replacing pixel values by the median value of pixels in that row that are read by the same multiplexer channel (i.e. every 16th pixel in that row starting with the pixel being corrected). The jailbar pattern is located by subtracting a 1-dimensional (along rows) median filtered image from the raw image.

The jailbar correction (`drip_jbclean.pro`) can be toggled on or off by setting the 'JBCLEAN' keyword in the `dripconf.txt` file prior to running the pipeline: `jbclean = 'MEDIAN'` switches the correction ON, `jbclean = 'N'` switches the correction OFF. There is currently no way to automatically determine, prior to data reduction, whether the jailbar correction will be necessary. If the jailbar correction is desired, it will be applied at the end of the stack step, to the chop/nod-subtracted image.

3.2.7. Optical distortion correction

The FORCAST optical system introduces anamorphic magnification and barrel distortion in the images. The distortion correction uses pixel coordinate offsets for a grid of pinholes imaged in the lab and a 2-d polynomial warping function (IDL's `polywarp.pro`, and `poly_2d.pro`) to resample the 256x256 pixels to an undistorted grid. The resulting image is 262x247 pixels with image scale 0.768"/pixel for a corrected field of view of 3.4x3.2 arc minutes. The distortion-corrected image is centered in a 512x512 pixel array to accommodate the distortion correction and to make room for subsequent shifting and adding of chop/nod images and for image rotation prior to the final coaddition step of the reduction process.

There is no distortion correction for the grism mode since the extracted spectra have a wavelength calibration applied to the array rows.

3.2.8. Image shift and rotation (merging)

The stack step of the pipeline in imaging mode produces images with multiple positive and negative source images depending on the chop/nod mode used for data acquisition. These positive and negative sources must be merged (`drip_merge.pro`) by copying, shifting, and re-combining the image. The final image must then be rotated to the nominal sky angle (NORTH = UP, EAST = LEFT in the displayed image).

The merge pipeline step (implemented in `drip_merge.pro`) makes a number of copies of the stacked image, shifts them by the chop and nod throws used in data acquisition, and adds or subtracts them (depending on whether the image is a positive or negative background-subtracted image). DRIP can use three different methods for registration in the merge process (selected in the `dripconf.txt` configuration file with the keyword `CORMERGE`):

- centroid of the brightest point sources in the stacked images (`CORMERGE='CENT'`)
- cross-correlation, usually best for extended or nebulous sources (`CORMERGE='COR'`)
- chop/nod data from the FITS header (`CORMERGE='N'`)

Then, the merged image is rotated using the `SKYANGLE` FITS keyword value. Thus, the final merged image consists of a positive image of the source surrounded by a number of positive and

negative residual source images left over from the merge shift and add process. The central image is the source to use for science.

Note that the most common reason for *failure* of the merge step to properly register stacked images is the centroid or the cross-correlation function triggering on a source other than the target. Spurious sources can be regions of poor background subtraction in the image, incomplete jailbar correction, or other array pattern noise.

For the NPC imaging mode with chop/nod amplitude smaller than the field of view, the stack step produces a single stacked image frame with a pattern of four background-subtracted images of the source, two of them negative. The merge step makes four copies of the stacked frame, then shifts each using the selected algorithm. It adds or subtracts each copy, depending on whether the source is positive or negative.

For the NMC imaging mode with chop/nod amplitude smaller than the field of view, the stacked image contains three background-subtracted sources, two negative, and one positive. The positive source has double the flux of the negative ones, since the source falls in the same place on the detector for two of the chop/nod positions. The merge step for this mode makes three copies of the image, shifts the two negative sources on top of the positive one, then subtracts them.

While performing the merge, the locations of overlap for the shifted images are recorded. For NPC mode, the final merged image is normalized by dividing by the number of overlapping images at each pixel. For NMC mode, because the source is doubled in the stacking step, the final merged image is divided by the number of overlapping images, plus one. In the nominal case, if all positive and negative sources were found and coadded, the signal in the central source, in either mode, should now be the average of four observations of the source. If the chop or nod was relatively wide, however, and one or more of the extra sources were not found on the array, then the central source may be an average of fewer observations.

For either NPC or NMC imaging modes, with chop/nod amplitude greater than half of the array, there is no merging to be done, as the extra sources are off the detector. However, for NMC mode, the data is still divided by 2 to account for the doubled central source. For C2NC2 mode, the chops and telescope moves-to-sky are always larger than the FORCAST field of view, so merging is never required. Merging is also not required for spectroscopy observations, as the spectra in the stacked image are extracted separately, and then coadded directly.

3.2.9. Spectral extraction

The FSpextool spectral extraction algorithms used by Redux offer two different extraction methods depending on the nature of the target source, as defined by the `SRCTYPE` FITS keyword. For point sources, Redux uses an optimal extraction algorithm, described at length in the Spextool paper (see the Other Resources section, below, for a reference). For extended sources, Redux uses a standard summing extraction, which simply sums the flux over an aperture, which can be specified directly by the user or determined automatically from the spatial distribution of the flux over the slit (the spatial profile).

For the NPC grism mode, with chop/nod amplitude less than the field of view, there will be a positive and a negative spectral trace in the stacked image. Redux extracts both, multiplying the negative spectrum by -1 to make it positive. It then merges the spectra by coadding them and dividing by 2.

For the NMC grism mode, with chop/nod amplitude less than the field of view, and chopping or nodding along the slit, there will be a positive and two negative spectral traces in the stacked image. In this case, Redux extracts all three spectra, multiplying the negative ones by -1. It then merges the spectra by coadding them and dividing by four (to account for the doubled central source). If the chop/nod amplitude for the NMC mode is larger than the field of view or it is chopping off the slit, there will be only a single spectral trace. In this case, Redux extracts this spectrum, then simply divides it by two to account for the doubled source.

All other grism modes produce a single positive spectral trace in the stacked image, which Redux extracts directly.

3.2.1. Image registration

In order to combine multiple imaging observations of the same source, each image must be registered to a reference image, so that the pixels from each image correspond to the same location on the sky. This step is performed by `drip_register.pro`. This algorithm uses the same three options for registration of images as the merge step (centroid, cross-correlation, or FITS header data), which are set in the `dripconf.txt` configuration file with the keyword `CORCOADD`. The first image in the set is treated as the reference image; the algorithm uses header data to shift this image to account for any initial dither offset. For all subsequent images, the specified algorithm is used to find the shift required to register it to the first image. The interpolation order of the shift may be set in the configuration file, using the keyword SHIFTORD. SHIFTORD=0 indicates that integer pixel shifts should be used, SHIFTORD=1 uses bilinear interpolation to do sub-pixel shifts, and SHIFTORD=3 (the default) uses a cubic interpolation to do sub-pixel shifts.

3.2.2. Coadding multiple observations

The final pipeline step is coaddition of multiple observations of the same source with the same instrument configuration and observation mode.

For imaging, the image combination is performed by an FSpextool algorithm that allows rejection of outlying values. The default statistic is a robust mean.

For spectroscopy, the FSpextool algorithm that combines the individual spectra by default scales them to a median value before combining them with a robust weighted mean statistic. It may also optionally correct each spectrum's spectral shape to the first spectrum in the set before combination.

3.3. Uncertainties

Redux calculates the expected uncertainties for raw FORCAST data as a variance image associated with the input data. It then propagates this variance image along with the data

through each processing step. This variance image is written to disk as an extra plane in all FITS images produced at intermediate steps. For the 1D extracted spectra written to disk, the uncertainty is saved as a standard deviation (the square root of the propagated variance) in an extra dimension in the image file.

FORCAST raw data is recorded in units of ADU per coadded frame. The variance associated with the *i,j*th pixel in this raw data is calculated as:

$$V_{ij} = \frac{N_{ij} \cdot \beta_g}{FR \cdot t \cdot g} + \frac{RN^2}{FR \cdot t \cdot g^2}$$

where N is the raw ADU per frame in each pixel, β_g is the excess noise factor, FR is the frame rate, t is the integration time, g is the gain, and RN is the read noise in electrons. The first term corresponds to the Poisson noise, and the second to the read noise. Since FORCAST data are expected to be background-limited, the Poisson noise term should dominate the read noise term.

The variance for the standard extraction is a simple sum of the variances in each pixel within the aperture. For the optimal extraction algorithm, the variance on the *i*th pixel in the extracted spectrum is calculated as:

$$V_i = \sum_j \frac{1}{{P_{ij}}^2 \cdot V_{ij}}$$

where P_{ij} is the spatial profile, V_{ij} is the variance at each pixel, and the sum is over all pixels j in the extraction aperture. This equation comes from the Spextool paper, describing optimal extraction.

3.4. Other Resources

For more information on how to run the FSpextool interactive tools (*xspextool, ximgtool*, *xvspec*, *xwavecal2d, xcombspec*, *xtellcor*, and *xcleanspec*), see the help files distributed with the FSpextool code, under *fspextool/Helpfiles*.

For more information about the Redux, DRIP, and FSpexool software architecture, see the Redux Developer's Manual, located in *redux/helpfiles*.

For more information on the reduction algorithms used in FSpextool, see the Spextool papers:
Spextool: A Spectral Extraction Package for SpeX, a 0.8-5.5 micron Cross-Dispersed Spectrograph Michael C. Cushing, William D. Vacca and John T. Rayner (2004, PASP 116, 362).
A Method of Correcting Near-Infrared Spectra for Telluric Absorption William D. Vacca, Michael C. Cushing and John T. Rayner (2003, PASP 115, 389).
Nonlinearity Corrections and Statistical Uncertainties Asociated with Near-Infrared Arrays William D. Vacca, Michael C. Cushing and John T. Rayner (2004, PASP 116, 352).

4. GROUPING LEVEL_1 DATA FOR PROCESSING

In order for a group of imaging data to be reduced together usefully, all images must have the same target object and be taken in the same chop/nod mode. They must also have the same detector, filter, and dichroic setting. In order to be calibrated together, they must also be taken on the same mission, with similar altitudes and zenith angles. Optionally, it may also be useful to separate out data files taken from different observations, or on different flight legs, or line-of-sight rewinds.

For spectroscopy, all the same rules hold, with the replacement of grism element for filter, and with the additional requirement that the same slit be used for all data files.

These requirements translate into a set of FITS header keywords that must match in order for a set of data to be grouped together. These keyword requirements are summarized below, in Table 1.

Keyword	Required for imaging	Required for spectroscopy
OBSTYPE	yes	yes
OBJECT	yes	yes
INSTCFG	yes	yes
DETCHAN	yes	yes
SPECTEL1 / SPECTEL2*	yes	yes
DICHROIC	yes	yes
MISSN-ID	yes	yes
ALTI_STA	yes	yes
ALTI_END	yes	yes
ZA_START	yes	yes
ZA_END	yes	yes
AOR-ID	optional	optional
FLIGHTLG	optional	optional
LASTREW	optional	optional
SLIT	no	yes

*SPECTEL1 is used if the detector is the SWC (DETCHAN=0),
SPECTEL2 is used for LWC (DETCHAN=1)

Table 1: Keyword requirements for FORCAST imaging and spectroscopy grouping

5. CONFIGURATION AND EXECUTION

5.1. Installation

Redux is a software package written in IDL that is designed to be a framework for executing any number or combination of data reduction algorithms. For FORCAST, it has been developed to support seamlessly running image processing algorithms from the DRIP package and spectral extraction algorithms from the FSpextool package. Redux can run in an automatic batch mode, integrated with the SOFIA Data Pipeline System (DPS), or it can run with a graphical front end

as a quick-look data viewer in flight or during manual data reduction and analysis. Redux with DRIP and FSpextool was developed under Linux and MacOS X operating systems, running IDL 8.1. Other operating systems and versions of IDL may also work, but have not been tested.

Running Redux requires IDL 8.1 or later, as well as the latest version of the IDL Astronomy User's Library, the Coyote graphics library, the DRIP package, the FSpextool package, and the Redux code. DRIP, FSpextool, and Redux are under SOFIA DPS revision control and can be obtained directly from git repositories there. The IDL Astronomy User's Library (astrolib) is publicly available, and can be downloaded from the website at the following URL:

http://idlastro.gsfc.nasa.gov/homepage.html.

The Coyote graphics library (coyote) is also publicly available and can be downloaded from:

http://www.idlcoyote.com/documents/programs.php.

When these packages have been installed, their locations should be added to the IDL_PATH environment variable, so that their procedures are accessible to Redux.

When you have the gzipped tar file of the Redux, DRIP, and FSpextool codes, unpack them, as, for example:

```
tar xvzf redux.tar.gz
tar xvzf drip.tar.gz
tar xvzf fspextool.tar.gz
```

This will create directories called `redux`, `drip`, and `fspextool`, which will contain a number of subdirectories. Each of these package directories should be added to the IDL_PATH as well.

5.2. Configuration

For DRIP algorithms, default options are specified in the `dripconf.txt` configuration file, located in the drip package directory, which is read when the pipeline is initiated. This file contains a list of keywords and their values that are used in the data reduction process. Some of the keywords are duplicates of FORCAST FITS header keywords and some are unique to the data reduction configuration. If a keyword is a duplicate FORCAST FITS header keyword the pipeline reads the keyword value in the dripconf file first, replacing the value listed in the raw FITS data file being reduced. We call this 'FITS keyword replacement' and it can be very useful when FITS files are non-standard for some reason (e.g. missing keywords or bad keyword values) or when experimenting with or testing DRIP algorithms. If a FORCAST keyword duplicate in the `dripconf` file is set to its own name (e.g. `instcfg = instcfg`) then the pipeline uses the keyword value in the raw FITS header of the data being reduced. See Appendix B for a sample of this configuration file.

For FSpextool algorithms, default options are specified in a FORCAST.dat configuration file, located in the fspextool package directory. This file also contains keyword-value pairs, in the format `parameter=value`. The parameters must all be present and in the correct order, but can have any number of spaces or comments between them. Comment lines begin with the % or # character. See Appendix B for a sample of this configuration file as well.

In automatic pipeline mode, parameters set in the configuration files are the values actually used by the pipeline. In interactive mode, the configuration files set the default values, but the parameter values used can be modified at run-time.

5.3. Input data

Redux takes as input raw FORCAST FITS data files, which are image cubes composed of 256x256 pixel image arrays. The number of frames per raw data cube depends on the chop/nod mode used to acquire the data (Table 2). FITS headers contain data acquisition and observation parameters and combined with the pipeline configuration file, `dripconf.txt`, comprise the information necessary to complete all steps of the data reduction process.

Chop/Nod Mode	Number of frames	Comments
C2N, NMC	4	Two-Position Chop with Nod Matched in throw and parallel to the chop direction 2 chop positions in each of 2 nod positions
C2N, NPC	4	Two-Position Chop with Nod perpendicular to the chop direction 2 chop positions in each of 2 nod positions
C2NC2	4	Extreme asymmetric chop and telescope move to blank sky: two chop positions per sky position. Typically 5 input files corresponding to ABAABAAB pattern; see Section 3.2.5 for details.
N	2	Two-position Nod only, may be used for grism spectroscopy
SLITSCAN	4 or C2NC2	Spectral map of an extended source, most likely using C2NC2 but could use C2N

Table 2: Contents of FORCAST raw data files by observing mode

The pipeline reads a raw FORCAST data file, identifies the observing mode used and associates the file with a set of ancillary and calibration data files (Table 3).

Ancillary data file	Data type	Comments
`dripconf.txt`	ASCII	Contains initial configuration of pipeline, path to calibration data, and if necessary, keyword replacement
`swc_badpix.fits`	FITS	Single 2-d image containing locations of bad pixels in short wave camera

SWC_linearity_coeff.fits	FITS	Image cube (256x256x10) containing images used to calculate pixel-to-pixel linearity correction for short wave camera
lwc_badpix.fits	FITS	Single 2-d image containing locations of bad pixels in long wave camera
LWC_linearity_coeff.fits	FITS	Image cube (256x256x10) containing images used to calculate pixel-to-pixel linearity correction for long wave camera
swc_darkfile.fits	FITS	Dark frame for short wave camera
lwc_darkfile.fits	FITS	Dark frame for long wave camera
swc_flatfile.fits	FITS	Flatfield frame for short wave camera
lwc_flatfile.fits	FITS	Flatfield frame for long wave camera
pinhole_locs.txt	ASCII	Pinhole locations file for distortion correction

Table 3: Auxiliary files

NOTE that although the DRIP has a fully functioning flatfield correction algorithm (`drip_flat.pro`), dark and flatfield files are not currently used since we are still working on understanding how best to gather flatfield data. Dark frames are in principle not necessary since the dark signal is removed during the background subtraction step of the pipeline (`drip_stack.pro`). However, dark frames are sometimes used to dark-subtract flatfield frames. The generic frames are stored in a calibration data directory distributed along with the code; they can be overridden at runtime in the interactive mode.

For each file in the list of raw data files to be reduced, ancillary and calibration data files are located and opened (`drip_getcal.pro`). If the necessary data files cannot be found the pipeline will try to complete data reduction, ignoring steps for which data are missing, or the pipeline will halt and complain that the data cannot be found or are not valid. Invalid data files may be missing or not located in the specified directory, be of the wrong format (e.g. 2D when they should be 3D), have incomplete or invalid FITS header information, or be corrupted. FITS data corruption most commonly happens during file transfer (e.g. `ftp`) and can often be fixed by repeating the transfer.

5.4. Required input keywords

In order for Redux to complete processing successfully, there are a few keywords that must be correct in the input FITS headers. These are described in Appendix A.

5.5. Automatic mode execution

Redux is an object-oriented program whose basic unit is a reduction object (`forcast_imaging_reduction__define.pro` or `forcast_grism_reduction__define.pro`). To run the pipeline from the IDL command line as a DCS black box pipeline, we run the pipeline wrapper (`forcast_pipe.pro`). This wrapper takes as input the path to an input manifest file. This

text file should contain a line specifying the number of input files, then the relative path to each input file, one per line. The script then reads these input files, instantiates the appropriate reduction object according to the mode specified in the input FITS headers, then calls the object's reduce method. This method calls each processing step in order, as appropriate for the given mode. Finally, the wrapper script will write an output manifest called `outfiles.txt` containing the names of the produced data files.

This wrapper can be invoked directly from the IDL prompt, as

```
IDL> forcast_pipe, 'infiles.txt'
```

or directly from a terminal as

```
localhost$ echo "forcast_pipe, 'infiles.txt'" | idl
```

The wrapper accepts a single input parameter on the command line, which, if set, will save all intermediate output. This option is specified as, for example:

```
IDL> forcast_pipe, 'infiles.txt', /save_intermediate
```

5.6. Manual mode execution

It is also possible to run the pipeline interactively, using a graphical user interface. The IDL command `redux`, called without arguments, will launch the Redux GUI.

5.6.1. Basic workflow

To start an interactive reduction, open a set of FORCAST files, using the File menu (**File->Open New Reduction**). This will bring up a file dialog window (Figure 10). All files selected will be reduced together as a single reduction set.

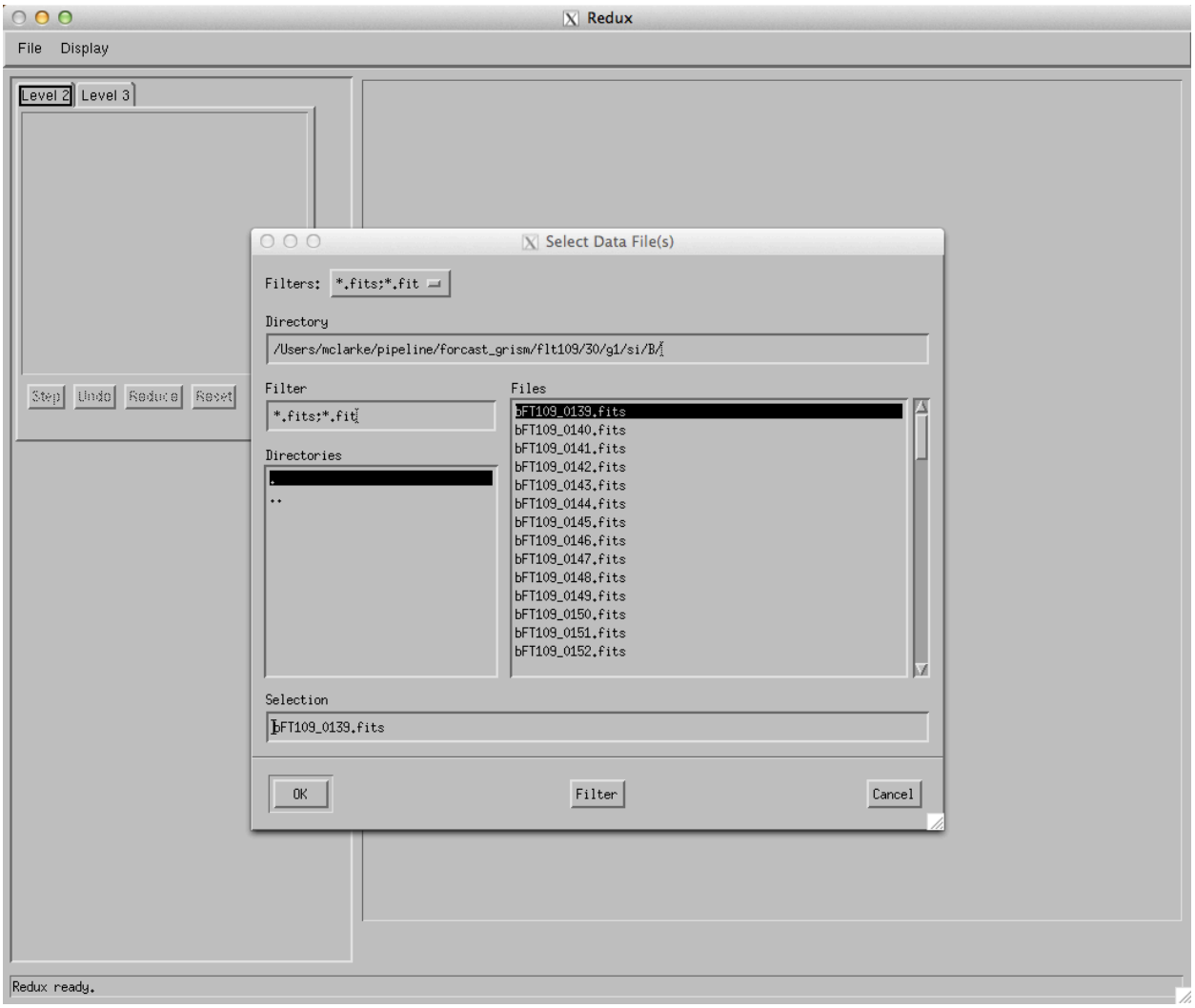

Figure 10: Open New Reduction

Redux will decide the appropriate reduction steps from the input files, and load them into the GUI. The steps for imaging inputs will differ from the steps for grism inputs (Figures 11 and 12), following the flowchart in Figure 6.

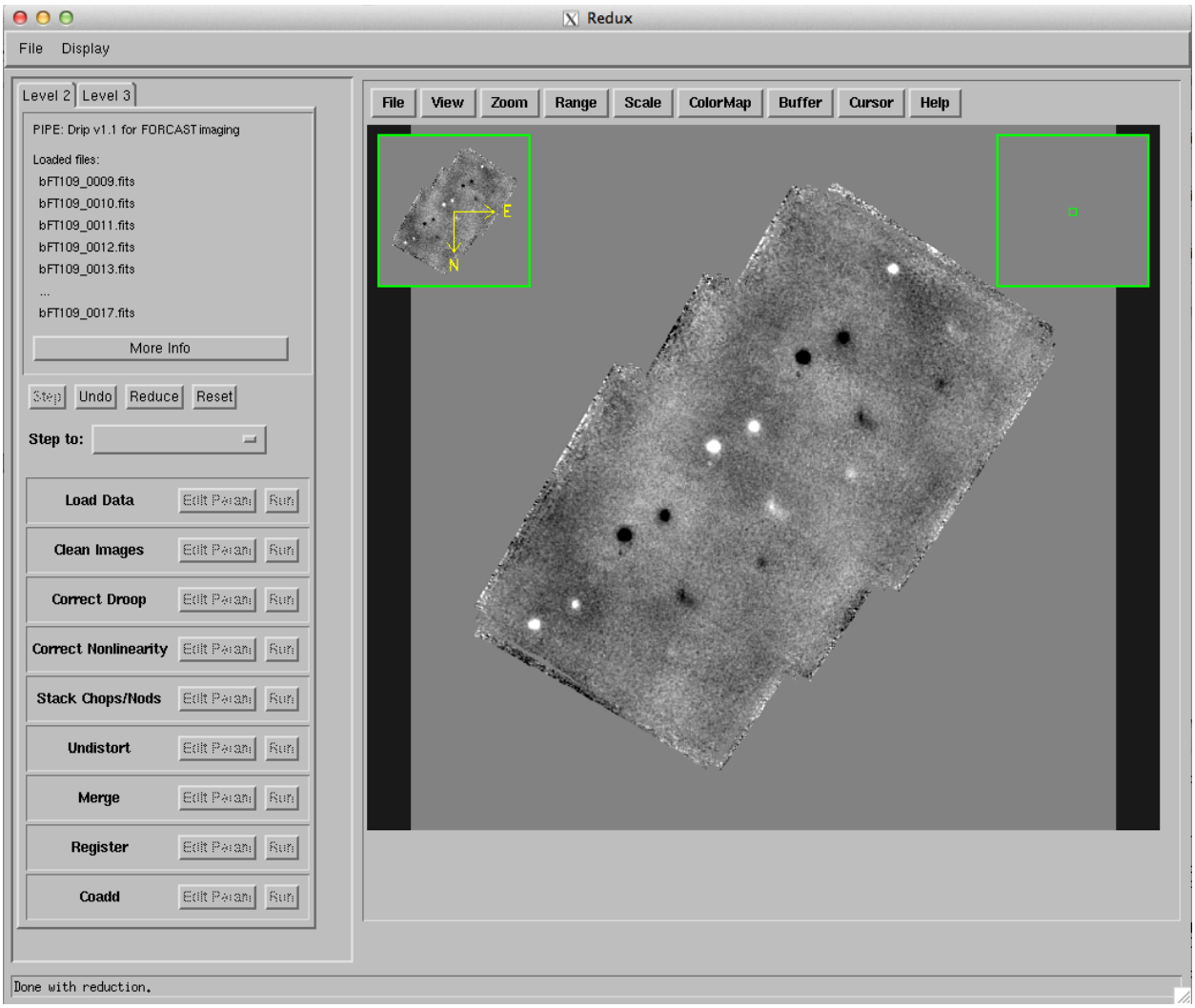

Figure 11: Imaging reduction

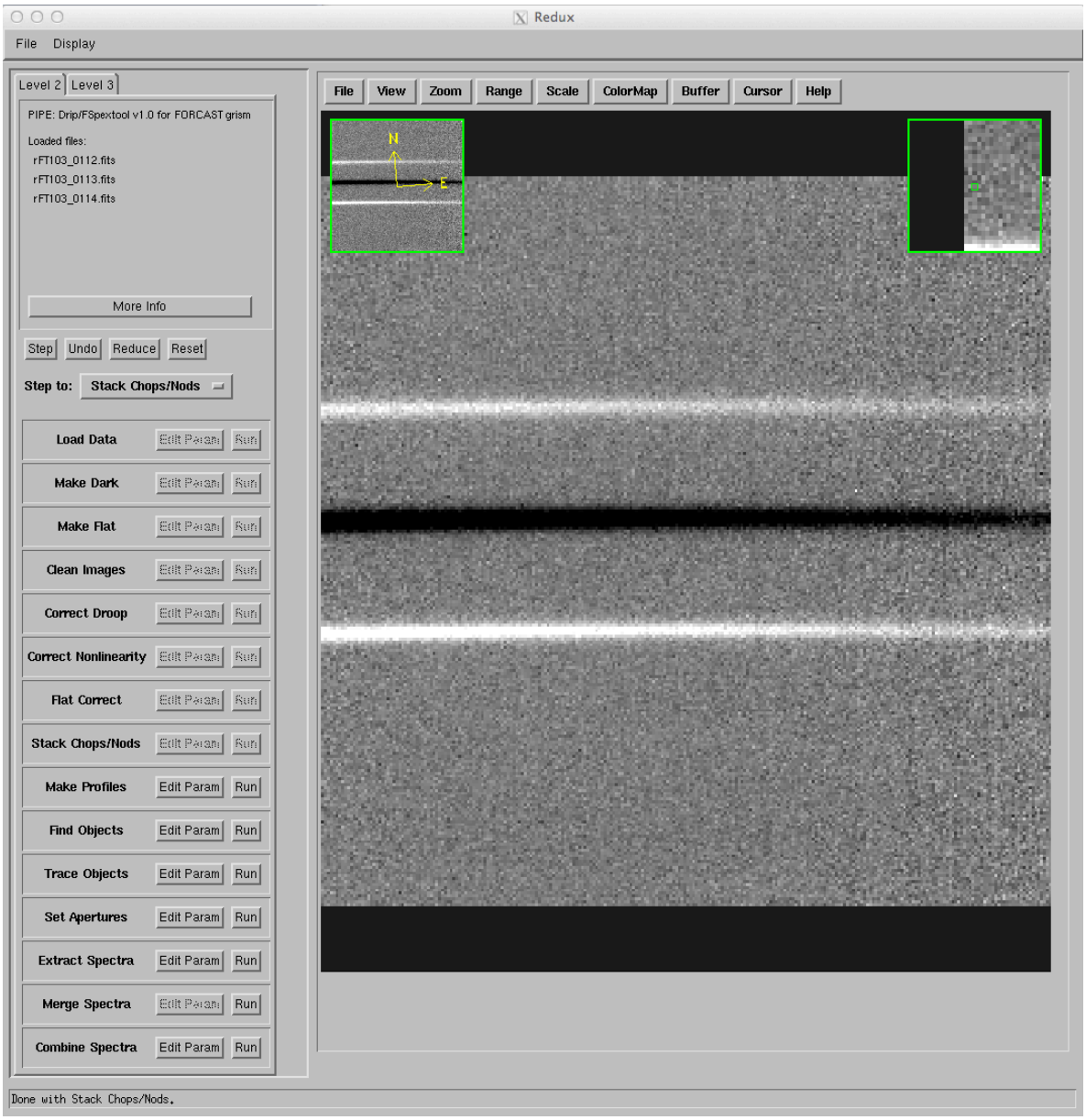

Figure 12: Grism reduction

In either mode, each reduction step has a number of parameters that can be edited before running the step. To examine or edit these parameters, click the **Edit Param** button next to the step name to bring up the parameter editor for that step (Figure 13). DRIP parameters are read from the `dripconf.txt` configuration file the first time the **Load Data** step is run, so parameters for later steps should not be edited prior to running this step. Within the parameter editor, all values may be edited, but will not be used unless **Save** or **Done** is selected. Clicking **Save** will leave the parameter editor window open; clicking **Done** will save values and close the window. Clicking **Reset** will restore any edited values to their defaults; clicking **Cancel** will discard all changes to the parameters.

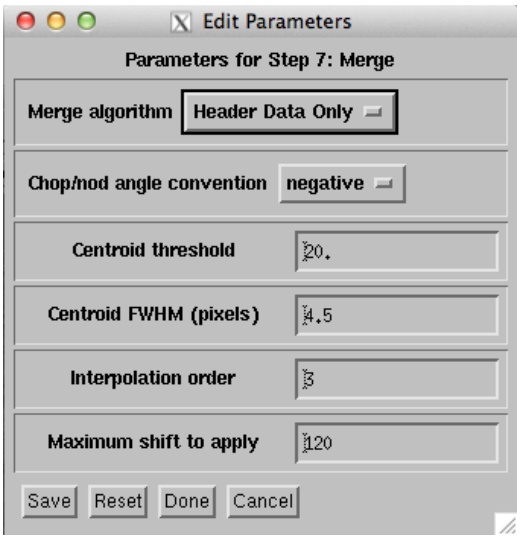

Figure 13: Sample parameter editor (for imaging Merge step)

After all parameters for a step have been examined and set to the user's satisfaction, a processing step can be run on all loaded files either by clicking **Step**, or the **Run** button next to the step name. Each processing step must be run in order, but if a processing step is selected in the **Step to:** widget, then clicking **Step** will treat all steps up through the selected step as a single step. When a step has been completed, its buttons will be grayed out and inaccessible. It is possible to undo one previous step by clicking **Undo**. All remaining steps can be run at once by clicking **Reduce**. After each step, the results of the processing will be displayed in the display window. Clicking **Reset** will restore the reduction to the initial state.

It is possible to reduce a single file at a time by opening a new reduction on a single file, reducing it, then adding in a new file (**File->Add Files**). Files can also be removed from the reduction set (**File->Remove Files**), but this will reset the reduction for all loaded files. Selecting **Display->Display File Information**, or the **More Info** button, will pull up a table of information about the currently loaded files (Figure 14). The table rows displayed can be filtered by entering a search string into the **Filter** text box.

All parameters can be reset at once by selecting **File->Reset All Parameters**. The currently displayed data can be saved to disk by selecting **File->Save Current Data**.

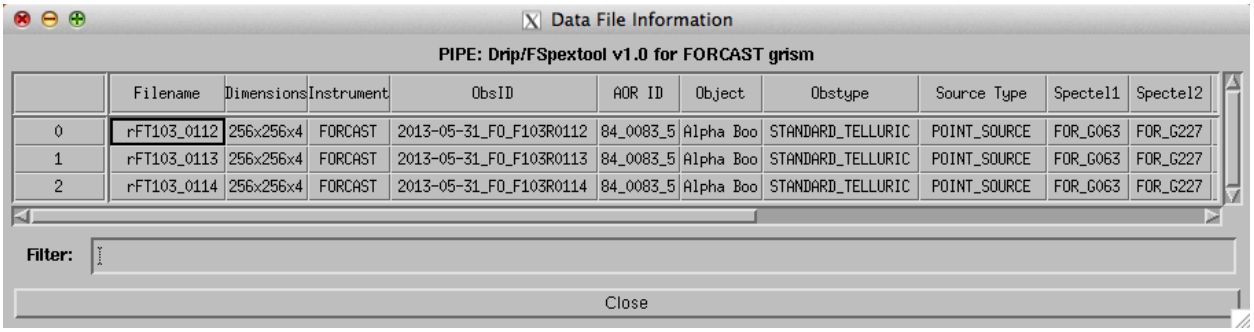

Figure 14: File information table

5.6.2. Display features

Redux displays images using ximgtool, a full-featured display tool distributed with FSpextool. For more information, see the ximgtool help file, available from Redux via the **Help** button just above the display. See Table 4 for a quick listing of the most useful ximgtool features.

Feature	Menu button	Keyboard shortcut
Load new file	**File->Load FITS**	--
Load file into new frame	**File->New Frame**	--
View FITS header	**File->View Header**	--
Zoom	**Zoom->Zoom In**, **Zoom Out**, **Zoom To Fit**	Press **z** to enter zoom mode, then **i** to zoom in, **o** to zoom out, or **t** to zoom to fit
Color stretch	**Cursor->Stretch**	Press **s** to enter stretch mode, click and drag to change brightness and contrast
Set display range	**Cursor->Range**	Press **r** to enter range mode, click and drag to select the box that sets the display range
Display distance	--	Press **d** to enter distance mode, then click and drag to identify start and end points
Line cut	--	Press **l** to enter line cut mode, then click and drag to identify start and end points
Display image statistics	--	Press **m** to enter moments mode, then click and drag to identify box for which the statistics should be calculated
Photometry	--	Press **a** over a star to do basic photometry.
Clear current mode	--	Press **c**
Buffer select	**Buffer->Buffer 1, Buffer 2…**	Press **f** to move to the next buffer, **b** to move to the previous buffer.
Buffer math	**Buffer->Buffer Math**, then select buffers and arithmetic operation to perform	--
Blink buffers	**Buffer->Blink Buffers**	--

Table 4: Useful ximgtool features

Ximgtool has five buffers available for simultaneous display of images. If there are more than five frames loaded into Redux, they can only be viewed by selecting **Display->Quick Look** from the Redux menu. This will cycle through the data in its current processing state, allowing interaction and analysis with each image in turn. To move between images, click the **Next File** or **Previous File** buttons, below the image. Click **Cancel** to quit the quick look display.

5.6.3. Imaging Reduction

FORCAST imaging reduction with Redux is straightforward. Each processing step in the GUI corresponds to a single step in the flowchart of Figure 6 (with the exception of the Jailbar Correct step: this is applied as part of the Stack Chop/Nods processing step in the Redux GUI).

Some useful options to note for imaging mode:

- Intermediate files can be saved by selecting the **Save all intermediate files** button in the Load Data parameters. The output path for reduced data can also be set in the Load Data parameters.
- The default for imaging is to subtract any residual background after stacking chop/nod frames. This can be turned off by deselecting **Subtract residual background** in the Stack Chop/Nods parameters.
- The default method for both merging and registration steps is to use the header data to determine the shifts. This can be overridden to select either the **Centroid** or **Cross Correlation** methods in the Merge or Coadd parameters.
- By default, the Coadd step examines the noise characteristics of each input file and attempts to exclude files that are determined to be unusually noisy. This can be turned off by deselecting **Exclude noisy files** in the Coadd parameters.
- The most common point of failure for the imaging mode is at the Register step. Therefore, it is sometimes useful to load in previously merged images and simply run the register and coadd steps. This can be done via Redux, using the **File->Open New Reduction** menu button, and selecting the merged products to coadd.

5.7. Grism Reduction

FORCAST grism reduction with Redux is slightly more complicated than for imaging. The GUI has additional steps to make processed dark and flat files, and breaks down the spectral extraction into six separate steps to give more control over the extraction process. These steps are:

- Make Profiles: Generate a smoothed model of the relative distribution of the flux across the slit (the spatial profile)
- Find Objects: Use the spatial profile to identify spectra to extract. By default, Redux attempts to automatically identify sources, but they can also be manually identified by entering a guess position to fit near, or a fixed position, in the Find Objects parameters. These manual positions should be comma-separated values, listed in arcseconds (refer to the spatial profile).
- Trace Objects: Identify the location of the spectrum across the array, by either fitting the continuum or fixing the location to the aperture center.
- Set Apertures: Identify the data to extract from the spatial profile (Figure 15). This is done automatically by default, but all aperture parameters can be overridden manually in the parameters for this step.
- Extract Spectra: Extract the one-dimensional spectrum from the identified apertures. By default, Redux will attempt optimal extraction for files with keyword SRCTYPE=POINT_SOURCE, and standard extraction for any other value. The method can be overridden in the parameters for this step.
- Merge Spectra: All apertures are merged into a single spectrum, and normalized according to the Chop/Nod mode, as detailed in section 3.2.9

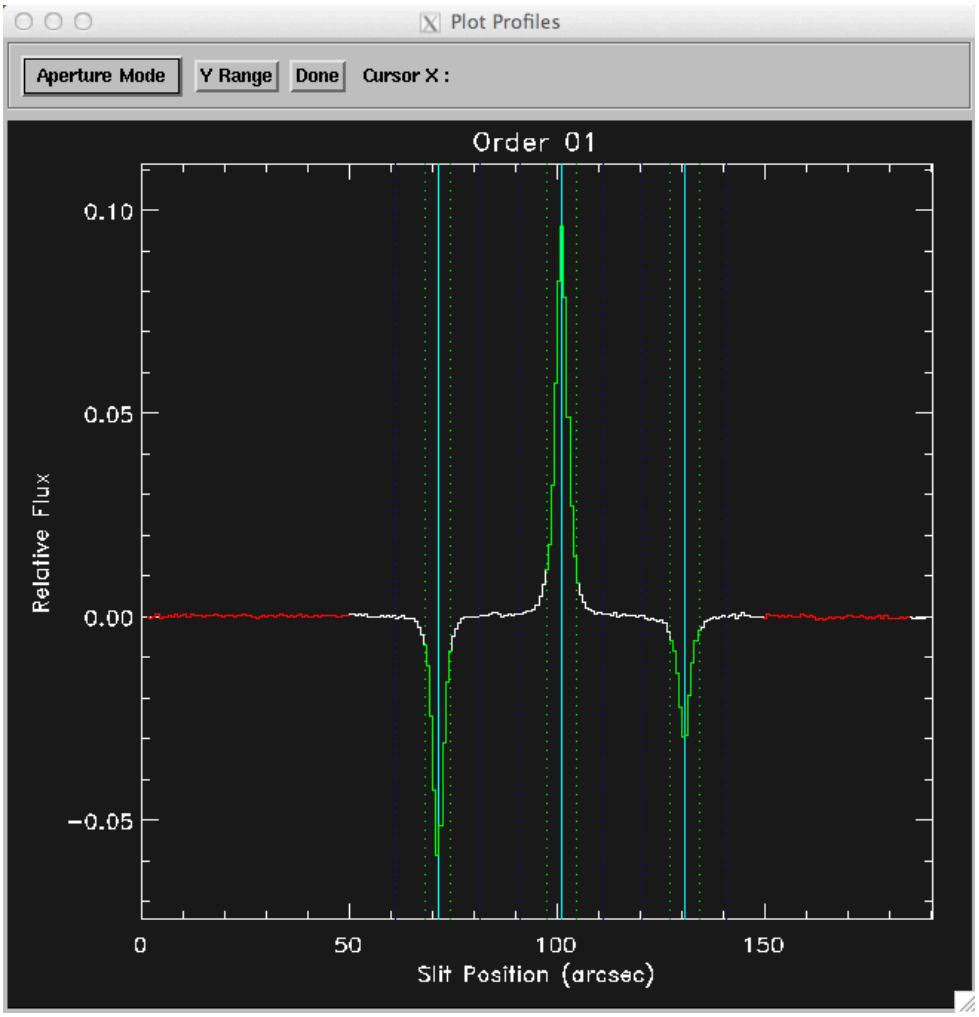

Figure 15: Aperture locations automatically identified and overplotted on the spatial profile. The three positions identified (light blue lines) correspond to the positive and negative spectra in Figure 12. Green lines indicated the extraction aperture, dark blue lines indicate the PSF radius (the point at which the flux goes to zero), and red lines indicate background regions.

Other important parameters to note:

- As for imaging reductions, intermediate files can be saved by selecting the **Save all intermediate files** button in the Load Data parameters. Output paths are also set in these parameters.
- Default dark, flat, and wavelength calibration files are determined after the Load Data step is run. They can be overridden in the parameters for the Make Dark, Make Flat, and Make Profiles steps.
- By default for all grism reductions except telluric standards, files that are taken at the same dither position are combined after chop/nod stacking, before extraction. This generally increases the signal to noise in the final spectra, as it results in a higher-quality spatial profile and spectral trace. This option can be turned off in the Stack Chops/Nods step by deselecting **Combine common dither positions**.

- Previously generated stacked images can be loaded directly into Redux for extraction using the **File->Open New Reduction** menu button, and selecting the stacked products. Previously generated merged spectra can be loaded similarly, for direct access to the combination step.

Extracted spectra are displayed using xvspec (Figure 16), a display tool packaged with FSpextool. Xvspec can only display one spectrum at a time, but each loaded spectrum can be examined by using the **Quick Look** feature, or else additional spectra may be directly loaded for display by clicking **Load Spextool FITS**. For more information on xvspec features, use the Help button in the xvspec window.

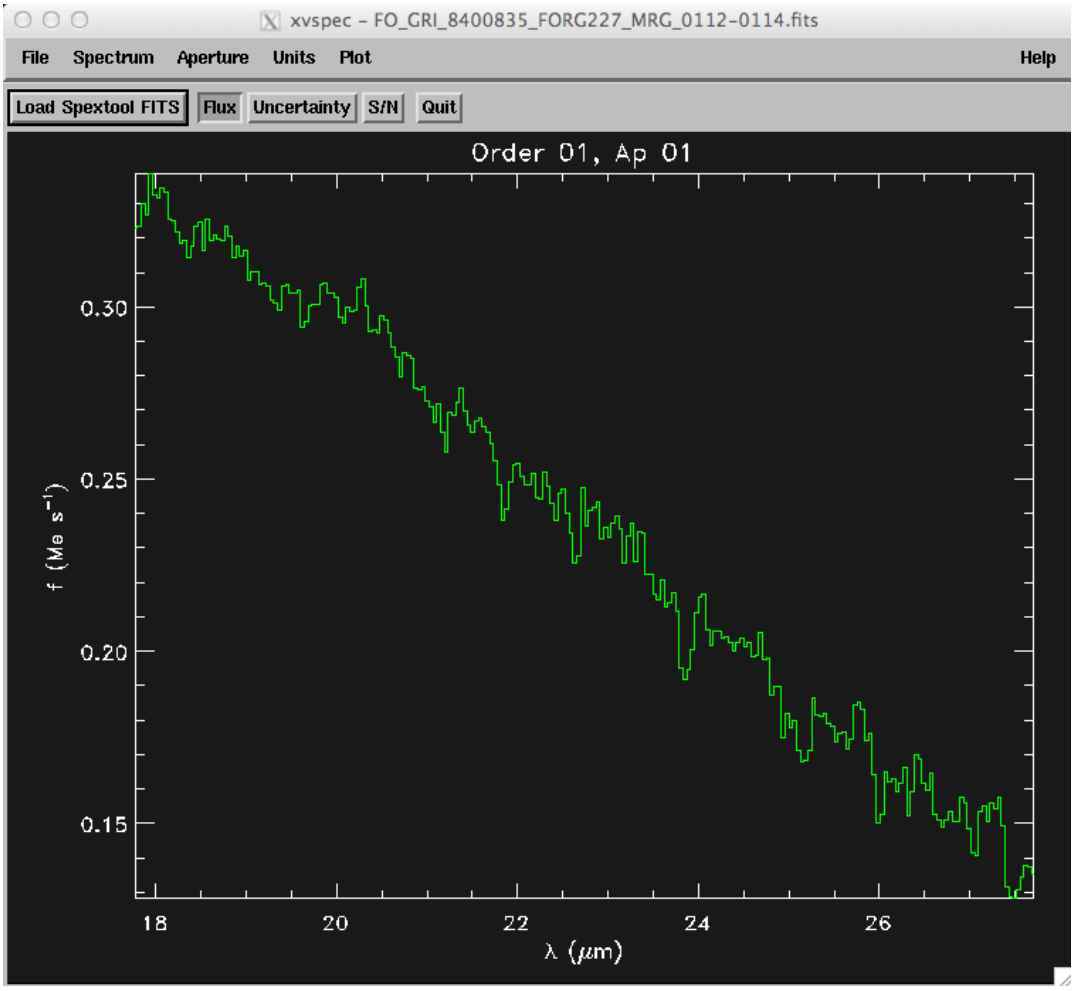

Figure 16: Final extracted spectrum, displayed in xvspec

6. FLUX CALIBRATION

6.1. Imaging Flux Calibration

The pipeline can apply a flux calibration. The calibration factors (e.g. Me-/sec per Jy) must be derived for each FORCAST filter configuration from observations of calibrator stars. The forcast pipeline includes a module to measure the photometry of the standard stars and apply the

calibration factor to the targets based on the theoretical flux of the standard star and the observed photometry. The module, namely pipecal, allows users via a GUI to interactively start the photometry and the calibration processes as well as visualize the results of each step in the form of tables.

6.1.1. Reduction Steps

The reduction is carried in four steps that in practice are performed iteratively. The first step consists of measuring the photometry of all the standard stars for a specific mission. Since the flux and colors of these stars are known, the absorption of the atmosphere to the observation can be derived. Standards are observed in a specific instrument configuration (filter and dichroic) as well as in an observation configuration (altitude and zenith). In principle, for each instrument configuration of an observed target there should be an observation of a standard star using the same configuration. Otherwise, the calibration cannot be performed for that target. However, there could be observation of standards at different observation configurations for the same instrument configuration, which don't necessarily match the observation configuration of the target. The second step of the process is to select the optimal standards for a specific target. There are various ways to select the optimal standard depending on how the calibration factor is calculated in the third step of the process. Currently, the pipecal module associates targets and standards by looking at the closest altitudes and zeniths for each target. This association will result on a list of standards for a specific target. In the third step, pipecal calculates the calibration factor and its error derived for each target and its associated standards. The final calibration for a specific target will be average of the calibration factors of the associated standards weighted by their error. The final calibration factor, its error and the reference wavelength value are written in the headers of the target as CALFCTR, ERRCALF and LAMREF. After all calibration factors were derived for a mission, the final step requires studying the calibration values. The calibration factor for each instrument configuration should be consistent within a mission and even between consecutive missions. Values that are not consistent may come from bad observations of a standard star. The operator should remove bad standard stars and start again the reduction process from the first step for each mission that contains inconsistent values.

6.1.2. Software Execution

To start the flux calibration software run one of the following methods:

- Go to the pipecal directory under the forcast pipeline tree, enter idl and call @start
- Set the following environment variables, start idl and run calplan_gui
    - IDL_PATH should contain all directories under pipecal
    - FLUXCAL_RESOURCES should point to pipecal/resources

Before the GUI opens, the code will open a message dialog window asking if the current directory contains the data. NOTICE that the search will be recursive, so any directory under the selected one will be inspected. To save your time, select the directory containing the data that you want to calibrate by clicking "No" to the dialog message and browsing to the data location. Then press "OK". The code will inspect all the data and separate the standards from the objects.

This process may take a few minutes. After it finishes, the results will show in two tables of the main window of the GUI. The top table will contain the information of the standards and the bottom the summary of the targets. Users can configure the information that is displayed in any of these two tables by clicking in the "Configure" button next to it. A pop-up window will show allowing checking the columns to be displayed. It is also possible to select displaying only some specific data by using the filter widgets on the right of the tables. Filters can also be configured by pressing "=>". To start the processing, you can either do each step of the processing separately by pressing first the "Run Standards" button and then "Run Objects". It is also possible to run both consecutively by pressing the "Run All" button. To perform the calibration of data in another location, change the directory by pressing "Browse" and browsing the location in the pop-up window. When done calibrating, select File->Quit to quit the program.

Note: Some functionalities are not implemented yet and buttons don't always have any application.

6.2. Spectrophotometric Flux Calibration

The objective of our spectral calibration efforts is to provide observers using the FORCAST Grism Mode on SOFIA with the data and tools necessary to produce wavelength-calibrated and flux-calibrated spectra. The common approach to characterizing atmospheric transmission for ground-based infrared spectroscopy is to obtain, for every science target, similar observations of a spectroscopic standard source with as close a match as possible in both airmass and time. Such an approach is not practical for airborne observations, as it imposes too heavy a burden on flight planning and lowers efficiency of science observations. Thus we will have to make use of a calibration plan incorporating a few observations of calibration sources per flight with a dynamic atmospheric model–i.e. one that we can adjust for varying atmospheric and observational conditions–and measurements (or estimates) of precipitable water vapor in parallel with calibration and science target observations.

6.2.1. Astronomical Calibrators

We will begin with a library of 8-10 stars for which we have SWS "truth" spectra (e.g. Engelke et al. [2010]) and systematically expand this list over time as we observe additional FORCAST Grism calibration sources with FORCAST/SOFIA.

The truth spectra for the calibration sources will have 1–2% precision in both photometry and spectrophotometry, but since the atmospheric spectrum is variable depending airmass, PWV, and as a function of wavelength, we expect large uncertainties to translate into low accuracy in relatively opaque regions of the FORCAST spectral range (e.g. we expect ~30% spectrophotometric accuracy in the ozone band, but in generally transparent regions we expect that 5–10% spectroscopic accuracy is possible).

6.2.2. Wavelength Calibration

We use a laboratory wavelength calibration derived from identifications of water vapor lines and a polynomial fit to centroids of those features in pixel space for each row (i.e. along the dispersion direction). We will use this method in flight with telluric lines, and with other spectral feature in observations of spectral standard sources, to improve this calibration. Specification of

a wavelength calibration is an interactive process, but application of the derived wavelength calibration is automatic and part of the data reduction pipeline.

6.2.3. Spectral transmission correction (including atmospheric correction)

Since flight scheduling will generally not enable frequent returns (or perhaps ANY returns) to a calibration source during a single flight (in order to match calibrator airmass and water vapor conditions to those during science target observations) we will use an atmospheric spectral correction derived from ATRAN models, measured (or estimated) precipitable water vapor (PWV), and the airmass of the science target observations. *The PWV will come either from the SOFIA in-flight water vapor monitor or from iterative fits of the water lines in the uncorrected spectrum to a library of ATRAN models scaled to the airmass of the individual observations.* We plan to use a flux calibration method similar to that used by the HIFOGS team on the KAO (Wooden et al.). This method starts with an atmospheric transmission model derived using ATRAN to generate a set of transmission corrections as a function of airmass, typically at 2–3 values for precipitable water vapor (PWV). We will try to observe a standard star at least three times during a flight to produce a spectral correction and use atmospheric transmissions for the most suitable PWV to modify the correction for differences in airmass. This method takes as input:

- a model of atmospheric transmission, such as ATRAN;
- the precipitable water vapor (PWV, either measured or estimated from fits of ATRAN models to uncorrected calibration source spectra);
- spectral observations of a standard star, along with its known, or "truth" spectrum; and
- housekeeping data for the observations, including the airmass of the target, and other parameters that might influence the atmospheric transmission (e.g., the aircraft's altitude, the time of the observation, etc.).

Standard star observations will be needed for each FORCAST grism configuration used. This method relies most heavily on the actual observations to determine a spectral correction, which has built into it a correction for atmospheric transmission. Because the atmospheric conditions will have changed between the observation of a standard source and a science target, transmission spectra from atmospheric models will be used, reproducing the PWV and airmass of both the science target and standard, and the differences in the models will be applied to the spectral correction.

There are some uncertainties in the above approach that will require some experimenting during commissioning of the FORCAST grism modes. One major concern is the temporal variation of the spectral correction. Eventually monitoring PWV will solve one major problem, but CO_2 and O_3 are also major components of the atmospheric absorption. Past experience on the KAO suggests that these scale, to first order, with PWV, but ground-based experience suggests that second-order effects can be significant. For example the O_3 absorption band at 9.6 μm is a particular concern, as it is variable and unpredictable.

Another issue we will address is the question of the spectral response function (SRF). The SRF is analogous to the point spread function (PSF), except that it applies in the dispersion direction. Effectively, it gives the profile of an unresolved spectral line. When shifting output from atmospheric models from the model resolution to the resolution of a FORCAST grism

configuration, deriving an accurate SRF is essential to prevent significant residuals between the model transmission and the actual transmission as recorded by the instrument.

7. DATA PRODUCTS

7.1.1. Filenames

Output files from Redux are named according to the convention:

FILENAME = FO_IMA|GRI_AOR-ID_SPECTEL1|SPECTEL2_Type_FN1[-FN2],

where FO is the instrument identifier, IMA or GRI specifies that it is an imaging or grism file, AOR-ID is the 8 digit AOR identifier for the observation, SPECTEL1|SPECTEL2 is the keyword specifying the filter or grism used, Type is three letters identifying the product type (listed in Tables 5 and 6, below), FN1 is the file number corresponding to the input file. FN1-FN2 is used if there are multiple input files for a single output file, where FN1 is the file number of the first input file and FN2 is the file number of the last input file.

7.1.2. Pipeline Products

Table 5 and 6 lists all intermediate products generated by Redux for imaging and grism modes, in the order in which they are produced. By default, for imaging, the undistorted, merged, registered, and coadded products are saved; for grism, the stacked, mrgspec, and combined products are saved. All products can be saved by specifying the appropriate option in either the automatic or interactive modes.

Step	Description	PRODTYPE	Identifier	Saved by default?
Clean	Bad pixels cleaned	cleaned	CLN	N
Droop Correct	Corrected for droop effect	drooped	DRP	N
Nonlinearity Correct	Corrected for detector nonlinearity	linearized	LNZ	N
Stack	Chop/Nod stacked	stacked	STK	N
Stack	Common dither positions combined (not usually used for imaging)	stackeddithers	SKD	N
Undistort	Corrected for optical distortion	undistorted	UND	Y
Merge	Chop/nod images merged into single source	merged	MRG	Y
Merge	Image overlap map used to determine normalization	normmap	NMP	N
Register	Multiple observations registered to a reference image	registered	REG	Y
Coadd	Multiple observations averaged	coadded	COA	Y

Table 5: Intermediate data products for imaging reduction

Step	Description	PRODTYPE	Identifier	Saved by default?
Make Dark	Processed dark frame	masterdark	DRK	N
Make Flat	Processed flat frame	masterflat	FLT	N
Clean	Bad pixels cleaned	cleaned	CLN	N
Droop Correct	Corrected for droop effect	drooped	DRP	N
Nonlinearity Correct	Corrected for detector nonlinearity	linearized	LNZ	N
Flat Correct	Divided by flat field	flatted	FTD	N
Stack	Chop/Nod stacked	stacked	STK	Y
Stack	Common dither positions combined	stackeddithers	SKD	Y
Extract	Bad pixels identified during extraction	mask	MSK	N
Extract	Raw extracted spectra	spec	SPC	N
Extract	Merged spectra	mrgspec	MRG	Y
Combine	Combined spectra	combspec	CMB	Y

Table 6: Intermediate data products for grism reduction

8. DATA QUALITY ASSESSMENT

8.1.1. Imaging

The first step in the quality analysis is to check the information from the pipeline. This information is available in the output log to the terminal and in the headers of the output files. Since the final product header contains all the information of the intermediate steps, a quick look at the header could tell us about any potential failure in the process. It is also important to look at the HISTORY of the headers of the final image and calibration files to identify messages from each step. Redux writes error messages in the header history when one of the steps is not performed correctly.

The second step of the quality analysis requires visualizing the images that were produced by the pipeline. First, check the final image for major problems. Some problems can be quickly identified by checking if the final image contains the expected pattern according to the observation mode. Then, check each of the intermediate products and calibration files to identify the following potential problems (see Table 7).

Product Type to Check	Features to look for	Problem
cleaned (or any subsequent image)	Bad pixels at the positions indicated by the mask.	Bad pixel removal was not performed correctly.
drooped	Stellar profiles dip below background level at their edges	Droop correction was not performed correctly.
merged	Elongated stars	Indicates that: - the shift of the merging was not calculated correctly * N option: chop and nod keywords are wrong * CENT and COR options: algorithm did not work - or the distortion correction is wrong. Check if it was performed and which input file was used. Check also related keyword values.
stacked or merged	Jailbar structures	Jailbar correction was not performed correctly at drip_stack step for the data.
merged	Pattern of stars does not match the expected one for observation mode.	The shift when merging was wrong. Check the shift values and compare them with the distance between stars in the stacked image. If CENT algorithm was used, there may be a detection of a cosmic ray as a star.
coadded	Central star shows twice or there is an obvious mismatch of two coadded images	The shift was calculated wrong or the headers are wrong. Check the dither keyword values for N option or the shift values for CENT or COR. NOTE: A bad coadd may not be obvious in the final image when many fits are combined because the average may remove the mismatched pattern. It is sometimes better to check the coadded image after each dither position.

Table 7: Potential problems with imaging reductions

8.1.2. Spectroscopy

Because all the products except for the extracted spectra are created in the same way as for imaging, the QA of spectroscopy data will be performed in a similar way as for imaging data. The difference will come from the fact that the features in the visualized data will be different (see Table 7).

Product Type to Check	Features to look for	Problem
cleaned (or any subsequent image)	Bad pixels at the positions indicated by the mask.	Bad pixel removal was not performed correctly
drooped	Spatial profile dips below background level near bright spectral trace	Droop correction was not performed correctly.
stacked	Jail bar structures	Jail bar correction was not performed correctly at drip_stack step for the data, or flat fielding was

		performed with a flat containing jailbar structures.
stacked	Position of spectra along the slit does not match the expected one for observation mode.	This may indicate a problem with the raw data, if a chop or nod failed.
stackeddithers	Multiple spectra appear where there should be only one	This may indicated that the source appeared in a different place in the slit for one or more input files
mrgspec, combspec	Features are duplicated or repeat a number of times.	The wavelength was calculated wrong or the headers are wrong.

Table 7: Potential problems with grism reductions

APPENDIX A: REQUIRED INPUT KEYWORDS

This table describes the type and expected value for all FITS keywords used by the DRIP/FSpextool pipelines.

Table 8: Required input keywords

Keyword	Type	Expected value
Image keywords		
ALTI_STA	float	0-60000.
ALTI_END	float	0-60000.
AOR_ID	string	
DATASRC	string	ASTRO, CALIBRATION, LAB, TEST, OTHER, FIRSTPOINT
DATE-OBS	string	yyyy-mm-ddThh:mm:ss[.sss]
DETCHAN	int	0, 1
DETECTOR	string	As-010, Sb-085
DETITIME	float	> 0
EPERADU	float	> 0
FRMRATE	float	> 0
ILOWCAP	bool	
INSTCFG	string	IMAGING_SWC, IMAGING_LWC, IMAGING_DUAL, GRISM_XD, GRISM_SWC, GRISM_LWC, GRISM_DUAL, GRISM_XD-LSV, GRISM-SSV, GRISM-LSV
INSTMODE	string	C2, C2N, C2NC2, N, SLITSCAN
INSTRUME	string	FORCAST
INTTIME	float	
MISSN_ID	string	
NAXIS1	int	256
NAXIS2	int	256
OBJECT	string	
OBS_ID	string	
OBSTYPE	string	OBJECT, STANDARD_FLUX, STANDARD_TELLURIC, LAMP, FLAT, DARK, BIAS, SKY
OTMODE	string	AD
OTSTACKS	int	>0
OTNBUFS	int	>0
SPECTEL1	string	FOR_F064, FOR_F066, FOR_F077, FOR_F111, FOR_F197, FOR_F242, FOR_G063, FOR_G111, FOR_XG063, FOR_XG111, NONE
SPECTEL2	string	FOR_F315, FOR_F336, FOR_F348, FOR_F371, FOR_G227, FOR_G329, NONE

TELESCOP	string	SOFIA
TIME-OBS	string	
WAVELNTH	float	0-40.
ZA_START	float	0-90.
ZA_END	float	0-90.
Shifting keywords		
DITHER	bool	
DITHERCS	int	0, 1, 2
NDITHERS	int	> 0
DITHERP	int	> 0
DITHERX	float	
DITHERY	float	
CHOPPING	bool	
CHPCOORD	int	0, 1, 2
CHPAMP1	float	≥ 0
CHPANGLR	float	
CHPANGLE	float	
CHPNPOS	int	> 0
NODDING	bool	
NODCOORD	int	0, 1, 2
NODAMP	float	≥ 0
NODANGLR	float	
NODANGLE	float	
NODBEAM	string	A, B
SKY_ANGL	float	
Instrument Mode keywords		
C2NC2	bool	
SKYMODE	string	C2NC2, NMC, NPC, NPCNAS, NPCCAS, SLITSCAN, NOD
Additional keywords for FG		
SRCTYPE	string	POINT_SOURCE, EXTENDED_SOURCE, OTHER, UNKNOWN
SLIT	string	FOR_SS24, FOR_LS24, FOR_LS47, NONE
Required dripconf keywords		
RESIZE	float	
BORDER	int	
CORMERGE	string	COR, CENT, N
MTHRESH	float	
MFWHM	float	
JBCLEAN	string	MEDIAN, FFT, N
XYSHIFT	float	
CORCOADD	string	COR, CENT, N
CTHRESH	float	
FRACDROOP	float	

MINDROOP	float	
MAXDROOP	float	
NRODROOP	int	
BADFILESWC, BADFILELWC	string	
DARKFILESWC, DARKFILELWC	string	
FLATFILESWC, FLATFILELWC	string	
MASKFLT	string	Y, N
FMASKFILE	string	
PIN_NPTS	int array	should have 2 elements
PIN_SPX	int array	should have 8 elements
PIN_SPY	int array	should have 8 elements
PINHOLE_FILE	string	
FIND_CAL	string	Y, N
CALDATA	string	
FKEYSEL	string array	
NLINREFS	float	
NLINSCAL	float	
NLCSWCLO	float array	
NLCSWCHI	float array	
NLCLWCLO	float array	
NLCLWCHI	float array	
LIMSWCLO	float array	
LIMSWCHI	float array	
LIMLWCLO	float array	
LIMLWCHI	float array	
ORDER	int	
NLINSECTION	int array	should have 4 elements

APPENDIX B: SAMPLE CONFIGURATION FILES

This is a sample of the `dripconf.txt` configuration file, used to set reduction parameters for DRIP algorithms and override FITS keyword values as necessary.

```
; Calibration files for swc and lwc cameras
badfileswc='swc_badpix.fits'
badfilelwc='lwc_badpix.fits'
flatfileswc='b039_1202.fits'
flatfilelwc='r039_1202.fits'
darkfileswc='b039_1203.fits'
darkfilelwc='r039_1203.fits'

; Parameters for variance calculation
rn_high=2400.
rn_low=244.8
beta_g=1.0

; Parameters for resizing imaging
resize=1.0
border=128

; Parameters for stacking
; if bgsub=1, residual background will be subtracted after stacking
bgsub=1

; Parameters for merging and coadding
anglconv='negative'
xyshift=15.
shiftord=0
; cormerge='COR' triggers drip_merge to use cross correlation
; cormerge='CENT' triggers drip_merge to use centroid
; cormerge='N' triggers drip_merge to use nominal chop/nod positions
CORMERGE = 'N'
CORCOADD = 'N'
; jbclean = 'FFT' triggers cleaning jailbar pattern with fft
; jbclean='MEDIAN' trigger cleaning jailbar pattern with median filter
; jbclean='N' no jailbar cleaning
jbclean = 'MEDIAN'
; Find peaks
cthresh=15.
mthresh=15.
;mfwhm=20.

; distortion correction
order=3
pinhole_file='pinhole_locs.txt'
pin_npts=[12,12]
pin_spx=[3,3,3,2,5,5,6,6]
pin_spy=[1,2,3,3,5,6,5,6]

; droop correction
mindroop    = 0.0
maxdroop    = 65535.0
nrodroop = 16
fracdroop = 0.0035

; Global image correction
nlinsection=[128,128,190,190]
nlinrefs = 7000.
nlinscal = 7000.
nlcswclo = [1.0000000, 0.39280,-0.16786,-0.11277]
nlcswchi = [1.0000000, 0.39280,-0.16786,-0.11277]
nlclwclo = [0.99867,0.34156,-0.14747,-0.07846]
nlclwchi = [0.99867,0.34156,-0.14747,-0.07846]
limswclo  = [1900.0, 12100.0]
limswchi  = [1900.0, 12100.0]
limlwclo  = [2500.0, 11000.0]
limlwchi  = [2500.0, 11000.0]
```

This is a sample of the FSpextool configuration file for FORCAST, called FORCAST.dat, condensed into a single image. Page 1 lists parameters used by the pipeline. Pages 2 and 3 list all keywords that will be propagated from the input files to the output files.

Page 1

```
# This is the calibration file for the FORCAST spectrograph on SOFIA.
# Note the values must be in the correct order, but can have any number of
# spaces/comments between them.
#
#------------------------------------------------------------------------
#
 INSTRUMENT=FORCAST
 NCOLS=256
 NROWS=256
 STDIMAGE=256
 PLOTWINSIZE=700 512
 FILENAME=FILENAME
 EXPTIME=EXPTIME
 TIME=TIME_OBS
 POSANGLE=None
 HA=None
 AIRMASS=None
 NINT=4
 BADPIXMASK=None
%
%CAL BASE
%
 CALMODULE=mc_forcastcals1d
%
%FILE READ MODE
%
 FILEREADMODE=Filename
 IPREFIX=F
 OPREFIX=r
 SUFFIX=*.fits*
 FITSREADPROGRAM=mc_readforcastfits
 HEADCOMBPROGRAM=mc_forcastdcshdr
 YUNITS=Me/s
 YTITLE=f (!5Me s!u-1!n)
%
% Reduction Mode
%
 REDUCTIONMODE=A
%
% Combine Base Information
%
 COMBMODE=A
 COMBSTAT=Median (Median Error)
 COMBTHRESH=8.0
 COMBODIR=proc/
%
%  Sky Base Information
%
 SKYSTAT=Robust Weighted Mean
 SKYTHRESH=8.0
%
% Profile Parameters
%
 YBUFFER=2
 OVERSAMP=1
 ATMOSTHRESH=0.7
%
% Point Source Base
%
PSNAPS=1
PSPSFRAD=22.0
PSAPRAD=7.0
PSBGSUB=1
PSBGSTART=24.0
PSBGWIDTH=30
PSBGDEG=0
PSBGMULT=2.0
%
% Extended Source Base
%
XSBGSUB=1
XSBG=0-9,24-36,54-60
XSBGDEG=0
%
% Additional processing base
%
 ADDLMODULE=mc_forcastaddlproc
%
%Other Base Parameters
%
 TRACEDEG=2
 TRACESTEP=7
 TRACESUMAP=7
 TRACESIGTHRESH=1
 TRACEWINTHRESH=5
 BADPIXELTHRESH=7
 PLOTSATURATEDPIXELS=0
 SATURATION=3000
 CHECKSEEING=0
 SEEINGTHRESH=3
 LINCORRECT=0
 ERRORPROPAGATION=1
 FLATFIELD=0
 FIXBADPIXELS=1
 OPTIMALEXTRACTION=1
```

Page 2

```
%
% FITS Header Keywords to Grab
% ----------------------------
%
% Keywords required for processing, may also be required
% by SOFIA DCS
%
KEYWORD=ALTI_STA
KEYWORD=ALTI_END
KEYWORD=AOR_ID
KEYWORD=CZNCZ
KEYWORD=CHOPPING
KEYWORD=CHPAMP1
KEYWORD=CHPANGLE
KEYWORD=CHPANGLR
KEYWORD=CHPCOORD
KEYWORD=CHPNPOS
KEYWORD=DATASRC
KEYWORD=DATE-OBS
KEYWORD=DETCHAN
KEYWORD=DETECTOR
KEYWORD=DETITIME
KEYWORD=DITHER
KEYWORD=DITHERCS
KEYWORD=DITHERX
KEYWORD=DITHERY
KEYWORD=DTHINDEX
KEYWORD=DTHNPOS
KEYWORD=EPERADU
KEYWORD=FILENUM
KEYWORD=FRMRATE
KEYWORD=ILOWCAP
KEYWORD=INSTCFG
KEYWORD=INSTMODE
KEYWORD=INSTRUME
KEYWORD=INTTIME
KEYWORD=MISSN-ID
KEYWORD=NODAMP
KEYWORD=NODANGLE
KEYWORD=NODANGLR
KEYWORD=NODBEAM
KEYWORD=NODCOORD
KEYWORD=NODDING
KEYWORD=OBJECT
KEYWORD=OBS_ID
KEYWORD=OBSTYPE
KEYWORD=OTMODE
KEYWORD=OTNBUFS
KEYWORD=OTSTACKS
KEYWORD=SKY_ANGL
KEYWORD=SKYMODE
KEYWORD=SLIT
KEYWORD=SPECTEL1
KEYWORD=SPECTEL2
KEYWORD=SRCTYPE
KEYWORD=TELESCOP
KEYWORD=UTCSTART
KEYWORD=WAVELNTH
KEYWORD=ZA_START
KEYWORD=ZA_END
%
% Keywords required by SOFIA DCS keyword dictionary
%
KEYWORD=KWDICT
KEYWORD=IMAGEID
KEYWORD=AOT_ID
KEYWORD=PROCSTAT
KEYWORD=HEADSTAT
KEYWORD=FILEREV
KEYWORD=PLANID
KEYWORD=DEPLOY
KEYWORD=FLIGHTLG
KEYWORD=ORIGIN
KEYWORD=OBSERVER
KEYWORD=CREATOR
KEYWORD=OPERATOR
KEYWORD=FILENAME
KEYWORD=DATE
KEYWORD=UTCEND
KEYWORD=WVZ_STA
KEYWORD=WVZ_END
KEYWORD=TEMP_OUT
KEYWORD=TEMPPRI1
KEYWORD=TEMPPRI2
KEYWORD=TEMPPRI3
KEYWORD=TEMPSEC1
```

Page 3

```
KEYWORD=AIRSPEED
KEYWORD=GRDSPEED
KEYWORD=LAT_STA
KEYWORD=LON_STA
KEYWORD=LAT_END
KEYWORD=LON_END
KEYWORD=HEADING
KEYWORD=TRACKANG
KEYWORD=TELCONF
KEYWORD=TELRA
KEYWORD=TELDEC
KEYWORD=TELVPA
KEYWORD=TELEQUI
KEYWORD=LASTREW
KEYWORD=FOCUS_ST
KEYWORD=FOCUS_EN
KEYWORD=TELEL
KEYWORD=TELXEL
KEYWORD=TELLOS
KEYWORD=TSC-STAT
KEYWORD=FBC-STAT
KEYWORD=OBSRA
KEYWORD=OBSDEC
KEYWORD=EQUINOX
KEYWORD=TRACMODE
KEYWORD=TRACERR
KEYWORD=MAPPING
KEYWORD=SCANNING
KEYWORD=DATATYPE
KEYWORD=MCCSMODE
KEYWORD=EXPTIME
KEYWORD=RESOLUN
KEYWORD=DETSIZE
KEYWORD=PIXSCAL
KEYWORD=SIBS_X
KEYWORD=SIBS_Y
KEYWORD=CHPFREQ
KEYWORD=CHPPROF
KEYWORD=CHPSYM
KEYWORD=CHPAMP2
KEYWORD=CHPCRSYS
KEYWORD=CHPTIP
KEYWORD=CHPTILT
KEYWORD=CHPPHASE
KEYWORD=NODTIME
KEYWORD=NODN
KEYWORD=NODSETL
KEYWORD=NODPATT
KEYWORD=NODSTYLE
KEYWORD=NODCRSYS
KEYWORD=DTHPATT
KEYWORD=DTHOFFS
KEYWORD=MAPCRSYS
KEYWORD=MAPNXPOS
KEYWORD=MAPNYPOS
KEYWORD=MAPINTX
KEYWORD=MAPINTY
KEYWORD=SCNRA0
KEYWORD=SCNDEC0
KEYWORD=SCNRAF
KEYWORD=SCNDECF
KEYWORD=SCNRATE
KEYWORD=SCNDIR
%
% Pipeline keywords
%
KEYWORD=PIPELINE
KEYWORD=PIPEVERS
KEYWORD=PRODTYPE
KEYWORD=BAD_OID
KEYWORD=LINC_OID
KEYWORD=DARK_OID
KEYWORD=FLAT_OID
KEYWORD=HISTORY
```