The GeRT User's Guide

The Germanium Reprocessing Tool

Spitzer Heritage Archive Documentation

Post-BCD Tools Team and Science User Support Team

Version 070904, February 2011

Contents

Chapter 1. Introduction

The Ge Reprocessing Tools (GeRT) allow users to carry out custom offline data reduction of Spitzer MIPS-Ge (70 µm and 160 µm) data. The GeRT is contributed software from the MIPS Instrument Support Team made available to the public to help users maximize their science returns with MIPS-Ge data. Users should be careful when processing their data offline. If used properly, the GeRT can yield enhanced data products, but users can also corrupt the quality and calibration of their data if used improperly.

The GeRT uses an offline version of the SSC pipeline to produce the basic calibrated data products (BCDs), following the algorithms derived by the MIPS Instrument Team and the MIPS Instrument Support Team (Gordon et al. 2005, PASP, 117, 503). With the GeRT, users can reprocess their data completely from scratch starting with the "raw" data from the archive or can cleanup (re-filter) the BCD products from the archive. The GeRT scripts are comprised of Perl and tcsh wrappers that use copies of SSC downlink binaries and libraries. The GeRT package includes all of the calibration files (e.g., DARK, IC, PMASK) and "namelist" files (namelists are ASCII parameter files that control the way the processing is done. They can be found in the *cdf/* directory of your GeRT distribution).

When new calibration files, namelist files, and/or new software techniques are available, users can use the GeRT to reprocess their data offline instead of waiting for the online re-processing of their data. In addition, expert users may find it useful to optimize the namelist files for their specific science application. The SSC online pipelines have been optimized primarily for faint point sources in low background regions. The online pipelines are robust for many science applications, but observers of bright, extended regions may find that offline processing with the GeRT is helpful. The MIPS-IST encourages users of the GeRT to provide feedback (contact the helpdesk at http://irsa.ipac.caltech.edu/data/SPITZER/docs/spitzerhelpdesk/), so that we can improve the processing of MIPS-Ge data.

The most common reasons for using the GeRT are:

- Re-filtering the data to optimize it for extended sources;
- Correcting poor stim flash calibration in the BCD frames;
- Re-filtering the data to correct for poor calibration due to bright sources;
- Recovering flux densities from saturated sources.

Important Documentation

The following documents and webpages are recommended reading, and are referred to throughout this User's Guide:

- MIPS Instrument Handbook:
  http://irsa.ipac.caltech.edu/data/SPITZER/docs/mips/mipsinstrumenthandbook/
- Spitzer Data Analysis Cookbook:
  http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/cookbook/
- GeRT main page:
  http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/tools/gert
- GeRT download page:
  http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/tools/gert/gert/

1.1 Getting Help

GeRT announcements are made online on the GeRT main page. The Spitzer Data Analysis Cookbook contains worked examples of common uses of the GeRT. If you have any questions about the GeRT, you can get help by submitting a ticket to the Spitzer Helpdesk. Please let us know which operating platform you are running (MacOSX/Linux/Solaris), as much information as possible about the problem, the AOR ID of your data, plus any error messages you saw. We endeavor to reply to all queries within 48 hours. Please note that our system does not autoreply. If you receive a message back within a few minutes, please check it for further information.

1.2 GeRT Updates

Version 070904 is the most recent release for the Mac version of the GeRT, and contains updated binaries and calibration files that are/will be implemented for the S16/S17 version of the online MIPS pipeline. S16 changes are restricted to changes of some cal files (160 µm flux conversion factor and various SED mode calibration files). For S17, there are some changes to the Interp module to better handle erroneous stimflash minus background calculations and stimflash extrapolations.

The 060415 version of the GeRT is the most recent version for all other platforms, and uses the latest SSC S14.0.0 software version and the current online S13.2 calibration files (see MIPS data handbook for calibration details). This version of the GeRT was designed to process all types of input MIPS-Ge data, e.g., PHT, SCAN, FINE, TPM, SED, DARK, and IC data. The standard science modes (PHT, SCAN) have been tested extensively, but the other modes have yet to be fully validated. In addition, the GeRT is now independent of the MOPEX modules.

The GeRT can now use Perl THREADs to carry out multiple processing jobs in parallel, significantly increasing the speed of the processing. This is particularly useful if running the GeRT on multi-CPU machines. Set the GERT_THREAD parameter to a value of approximately number_of_CPUs + 1 ---> 2 x number_of_CPUs for optimal performance. To use THREADs your system Perl must include Thread.pm; if you run into "Can't locate Thread.pm in @INC" errors, then your system does not support Perl threads, and you will need to set GERT_THREAD = 0.

The old version of the GeRT only worked for Solaris with the FORTE compilers and was tied to a specific version of Perl used by our database system. The new version of the GeRT uses the Perl version on your system. Not all Solaris systems support the FORTE compile. GNU-compiled versions of the GeRT for Solaris, Linux, and Mac have also been made recently. The GNU-software versions make the GeRT more portable, but some system specific issues still exist given the use of shared system libraries.

1.3 GeRT Download and Setup Instructions

GeRT can be downloaded from the IRSA website, at
http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/tools/gert/gert/

The webpage provides full installation and setup instructions for all of the supported operating systems (Solaris, Linux, Mac). .

The GeRT package includes the following directories and files:

cal/	Calibration files used for MIPS-70 and MIPS-160
cdf/	Namelist directory
cdf/2	Namelists for MIPS-70 bcd pipeline processing
cdf/3	Namelists for MIPS-160 bcd pipeline processing
cdf/*.nl	Namelists for filtering scripts and postbcd processing
scripts/	Scripts for MIPS-Ge processing
bin/	MIPS-Ge binaries
lib/	MIPS-Ge shared libraries (if applicable)
gert_setup.csh	Environment setup file for running GeRT software
gert_doc	Documentation for this package (ASCII)

1.4 BCD Processing

The majority of users will find the online BCD products sufficient for their purposes (before or after re-filtering; see §2.1). However, since it is simple to reprocess data, users may want to check if the GeRT offline processing yields better products than those currently available in the SSC archive. In particular, the GeRT has proven useful for the following cases:

1. Data where the online stimflash calibration was not done optimally (e.g., extrapolated stimflash calibration or stimflash calibration corrupted by bright emission regions).

2. New updated calibration files and/or pipeline corrections which are not available online yet.

3. Optimization of the pipeline namelist files for non-standard data and/or specific science applications (e.g., ultra-deep data with high redundancy vs low redundancy data).

4. Data brighter than the advertised saturation limits (see §2.4).

1.5 How to Run the GeRT

1. Source the GeRT setup file (see §1.3 to setup the GeRT on your system).

   ```
   unix% source gert_setup.csh
   ```

2. Make an input list of RAW data files.

   ```
   unix% ls *raw.fits > IN70.lis
   ```

3. The list of input files should match the order in which the data are taken. An example raw data file from the SSC archive has a name like the following:

   ```
   SPITZER_M2_12345678_0001_0004_2_raw.fits
   ```

   where

M2	= 70μm (and M3 = 160μm)
12345678	= AORKEY
0001	= EXPID exposure sequence number
0004	= DCENUM within the EXPID exposure
2	= archived processing number

Since names are given with leading zeros, "ls" will list the files in the order in which the data were taken. Users should take special care and only include the data they want to reduce in the list, e.g., only PRIMEARR data. For photometry mode observations, one can get EXPIDs associated with 24µm, 70µm, and/or 160µm data for each AOR. If taking photometry data in all three bands, the first set of EXPIDs will be 24µm, followed by 70µm, and then followed by 160µm observations. Read the fits headers to figure out which EXPIDs are associated with which array (keyword FOVNAME). e.g.:

```
unix% imheader rawdata/*00*_0000_raw.fits | grep FOVNAME
FOVNAME = 'MIPS_70um_default_small_FOV1' / Field of View Name
FOVNAME = 'MIPS_70um_default_small_FOV1' / Field of View Name
```

4. Run the GeRT, providing the input list and an output directory.

   e.g.,

```
unix% $WRAPDIR/gert.pl IN70.lis OUTdir

GeRT S14.0 v1.1 with 0 threads
0001 <: rawdata/MIPS.2.0012997376.0000.0000.raw.fits
0002  * rawdata/MIPS.2.0012997376.0001.0000.raw.fits
0003  : rawdata/MIPS.2.0012997376.0001.0001.raw.fits
0004  : rawdata/MIPS.2.0012997376.0001.0002.raw.fits
0005  : rawdata/MIPS.2.0012997376.0001.0003.raw.fits
0006  : rawdata/MIPS.2.0012997376.0001.0004.raw.fits
0007  : rawdata/MIPS.2.0012997376.0001.0005.raw.fits
0008  * rawdata/MIPS.2.0012997376.0001.0006.raw.fits
SLOPER Processing done. 8 dce's in 22 seconds.
CALER Processing done, BCDs in out/bcd processed in 3 seconds.
 Total processing of 8 dce's processed in 25 seconds
```

   IN70.lis is an input list of RAW fits files and *OUTdir* is output directory of the GeRT products. SLOPER and CALER are the two main steps of processing described in Section 4.2. The symbols "*" indicate stimflash frames while ":" is shown for non-stimflash data.

For default processing, the GeRT reads the header of the first file to determine the data type and picks the appropriate pipeline. The important keywords that control the processing are CHNLNUM, EXPTYPE, and FOVID:

CHNLNUM = 2 for 70µm
CHNLNUM = 3 for 160µm
EXPTYPE = scn ==> Scan data, normal science pipeline is run.
EXPTYPE = pht ==> Photometry data, normal science pipeline is run.

		If EXPTYPE=pht and FOVID>117, then the data are 70µm. FINE-scale observations and the appropriate calibration files are chosen.
EXPTYPE = sed	==>	70µm SED data, SED pipeline is run.
EXPTYPE = tpm	==>	Total power mode data, TPM pipeline is run.
EXPTYPE = d2a	==>	DARK data, DARK pipeline is run.
EXPTYPE = sfl	==>	Illumination Correction (IC) data from scanning, IC pipeline is run.
EXPTYPE = pfl	==>	IC data from photometry, IC pipeline is run.
EXPTYPE = ffl	==>	IC data from FINE-scale photometry, IC pipeline is run.
EXPTYPE = dfl	==>	SED IC data, IC pipeline is run.
EXPTYPE = tfl	==>	Total power IC data, IC pipeline is run.

The above listing provides the standard MIPS-Ge data types. See the MIPS Instrument Handbook for a full list of file definitions and for information about the Illumination Correction.

Users can force the GeRT to run a specific pipeline by providing the EXPTYPE on the command line (3rd input argument). For example, if you would like to make your own Illumination Correction (IC) calibration file from your science data you could try the following, e.g.:

```
unix% $WRAPDIR/gert.pl IN70.lis OUTdir sfl
```

where "sfl" causes the GeRT to run the IC pipeline.

1.6 My edited namelist files don't seem to work

When editing namelist files, you should include a space before and after the "=" for numerical parameters, and you must end all lines in the namelist with a "," (even comment lines need to be followed by ",". The parameter lines must be within the namelist block (e.g., between &BLOCKNAME and &END).

example of GOOD input:

```
StimHi = 4,
```

example of BAD input (don't do):

```
StimHi = 4
StimHi =4,
StimHi= 4,
StimHi=4
```

The GeRT namelist files are stripped down versions of the online namelists. Blocks only used online are not included in the offline GeRT namelists. The online input/output (I/O) names of the modules is controlled via namelists, but offline, the GeRT perl script defines the I/O names, which takes precedent over what is listed in the namelist.

1.7 Pipeline Summary

BCD processing has two main steps: (1) calculation of the slope of the data ramp (SLOPER) and (2) calibration of the slope image (CALER). MIPS-Ge raw data are comprised of data ramps of 24, 32, or 80 non-destructive reads (for 3, 4, 10 MIPS-sec data collection events (DCEs) respectively).

The namelist file controlling "SLOPER" processing is *cdf/{2/3}/MIPS{70/160}_SLOPE_0.nl*. The first step in the SLOPER processing of the data is CVTI2R4g which converts the input integer values into real values and makes the initial dmask and bmask files (output "*dce.fits*"). The SATURATION module sets bits in the dmask for saturated samples. The ELECNL module corrects for electronic nonlinearity (output "*enlcorrd.fits*"). RESET checks for extra resets in the DCE and sets the dmask appropriately. RADHIT checks for radhits in the ramps and sets the dmask at the location of radhit(s). The SSC pipeline identifies discontinuities using a maximum likelihood technique (Hesselroth et al. 2000, Spaceborne Infrared Remote Sensing VIII, SPIE Conference Proceedings, 4131, 26). SLOPE calculates the linear slopes of segments with at least 4 consecutive good reads as indicated by the dmask (output "currents.fits"). FUSION calculates a noise-weighted average slope from the segments based on the empirical errors estimated from the scatter of the data within the ramp segments (output "current.fits"). The slope image (current.fits) is the final product of the SLOPER part of the pipeline. The last step in sloper is MASKWRITE which copies information from the dmask and pmask to the bmask file for use in the 2nd part of the pipeline.

The 2nd part of processing [CALER] is controlled via the namelist file *cdf/{2/3}/MIPS{70/160}_FLUXCAL_0.nl*. Calibration and filtering are done on data cubes. Input cubes are made by stacking the SLOPER products with the STACKLAYER module. The calibration of MIPS-Ge data is based on frequent measurements of internal stimulator flashes (stims), which are used to track the response of the detectors as a function of time. The stim flash signal is measured by subtracting the previous ``background'' DCE from the stim frame, which is taken at the same position on the sky. For each AOR, a stim response function (SR[t]) is calculated from interpolating between the stim minus background measurements, which is done via the module INTERP (output "*interpstim.fits*"). After the determination of the stim response as a function of time (SR[t]), the BCD data are calibrated in SLOPECAL module using the following equation:

$$BCD(t) = \frac{FC\left[\frac{U(t)}{SR(t)} - DARK\right]}{IC}$$

Equation 1.1

where U(t) is the uncalibrated slope image, DARK is the dark calibration file, and IC is the illumination correction calibration file which corrects for the combined illumination pattern from the telescope and the stim flash signal. The DARK and IC calibration files are stable and are generated by combining data from several different campaigns to improve the signal-to-noise. The flux conversion factor (FC) converts the instrument units into physical surface brightness units of MJy/sr and is derived from observations of standard calibrators.

The SLOPECAL module of CALER, which applies the calibration, also does optional filtering. The online filtering process is optimized for point sources. Filtering removes data artifacts, but also removes extended source flux. Examples and discussion of MIPS-Ge artifacts are presented in the MIPS Instrument Handbook. The two main artifacts impacting Ge data are the stim flash latents and the variations of the slow response as a function of time. The 160µm data are affected by these issues to a lesser degree due to the faster time constants of the 160µm stressed-Germanium detectors. For point sources in low backgrounds (e.g., *Frayer et al. 2006, AJ, 131, 250*), the stim latent and slow response residuals are additive effects. The Ge filtering removes a running median per pixel as a function of time by subtracting the median value of the surrounding DCEs closest in time (ignoring the current DCE, stim DCEs, and bad data). This "high-pass" time filter does not remove all of the affects of stim latents. To remove the stim latent at 160µm, one can throw out the first DCE after the stim flash since the stim latents typically decay away within one DCE. At 70µm stim latents remain for many DCEs and are correlated by column. Since the scan map direction and photometry scan dithers are nearly along the columns of the array, the column artifacts are amplified. Column residuals at 70µm are removed by subtracting the median of the values along each column for every DCE. The combination of the high-pass time median filter per pixel and the column median filter removes the bulk of the data artifacts at 70µm.

1.8 GeRT Products

The GeRT makes many intermediate products in addition to the BCD products one can retrieve from the online archive. These intermediate products provide important diagnostic information about the data as well as enabling the testing and optimization of the pipeline software. The products from SLOPER (the first part of the pipeline) are stored under *OUTdir/0XXX*, where XXX = increasing number corresponding to the files in the raw input list.

1.8.1 SLOPER Products:

bmask.fits	bmask file used to identify bad pixels (32x32)
current.fits	Final slope image from SLOPER
current_unc.fits	Uncertainty of slope image
currents.fits	Slopes of line segments within the ramps
currents_unc.fits	Uncertainties on the line segment slopes
dce.fits	Raw data ramp converted into real
dmask.fits	Mask used to identify bad reads in ramps (32x32xN)
enlcorrd.fits	Data ramp corrected for the electronic nonlinearity
enlcorrd_unc.fits	Uncertainty on enlcorrd.fits
sloper.log	Log file for the sloper processing

The intermediate products of the CALER part of the pipeline are stored in *OUTdir*, while the BCD products are stored in the *OUTdir/bcd* directory.

1.8.2 CALER Intermediate Products:

SCLK_OBS.fits	Vector giving DCE timing information used for stim interpolation (SCLK_OBS keyword)
*.txt	Output listing of file names for GETLAYER
*list	Input listing of file names for STACKLAYER
bmaskcube.fits	Final bmask cube from SLOPECAL module associated with nofiltcube
cal@	Symbolic link to cal files ($SOS_GeRT/cal)
calcube.fits	Final filtered data cube from SLOPECAL
calcube_bmask.fits	Mask file associated with calcube
calcube_unc.fits	Uncertainty file associated with calcube
caler.log	Log file for the caler processing
caler_stack*.log	Stacklayer log (error code: 0 = ok)
curcube.fits	Cube of uncalibrated slopes
curcube_unc.fits	Uncertainty associated with curcube
darksubt.fits	Intermediate file used for making the IC in the calibration pipelines [U(t)/SR(t) - DARK]
darksubt_bmask.fits	Mask file associated with darksubt
darksubt_unc.fits	Uncertainty file associated with darksubt
fluxcal.nl	Copy of MIPS*_FLUXCAL_0.nl namelist file used by CALER part of the pipeline
interp_bmaskcube.fits	Bmask file output from INTERP module
interpstim.fits	Stim Response cube from INTERP
interpstim_unc.fits	Uncertainty file associated with interpstim
nofiltcube.fits	Unfiltered BCD cube
nofiltcube_unc.fits	Uncertainty file for nofiltcube

slopecal.nl	Copy of MIPS*_SLOPE_0.nl namelist file used by SLOPER part of the pipeline

1.8.3 BCD Products (OUTdir/bcd directory)

bcd.fits	Unfiltered BCD products made from *nofiltcube.fits*
bunc.fits	Uncertainty file made from *nofiltcube_unc.fits*
bmask.fits	bmask file made from *bmaskcube.fits*
fbcd.fits	Filtered BCD product made from *calcube.fits*

The bunc and bmask files are applicable for both the BCD and fBCD products. These suffixes match the names of the SSC archived product names. The full name of the GeRT BCD products are given by *OUTdir.XXXX.sufix.fits*, where *OUTdir* is the user given output directory, *XXXX* is an increasing number associated with the DCEs in the input list (e.g., 0001 is associated with the first DCE in the list, 0002 associated with the 2nd DCE...), and *sufix* = (<=5)-character suffix (bcd, bunc, bmask, fbcd).

Please note if combining data with MOPEX, all input files names must be unique (recommend using different *OUTdir* names for different AORKeys when working with large data sets).

1.8.4 Calibration Pipeline Products

For the DARK pipeline (EXPTYPE=d2a), the products are (*OUTdir* directory):

darkfinal.fits	Output DARK calibration file corresponding to online calibration file *MIPS*_DARK.fits*.
darkfinal_mask.fits	Mask file for DARK corresponding to online file *MIPS*_DARK_C.fits*.
darkfinal_unc.fits	Uncertainty file for the DARK corresponding to online file *MIPS*_DARK_U.fits*.
*MIPS*_DARK_M.fits*	Coverage map of DARK file (representing number of "good" DCES used per pixel)

For the IC ("flat") pipeline (EXPTYPE=sfl,pfl,dfl,tfl,ffl), the products are (OUTdir directory):

ilcorr.fits:	Output IC calibration file corresponding to online calibration file *MIPS*_ILCORR.fits*.
ilcorr_cmask.fits:	Mask file for the IC file corresponding to online file *MIPS*_ILCORR_C.fits*.
ilcorr_unc.fits	Uncertainty file for the IC file corresponding to online file *MIPS*_ILCORR_U.fits*.
*MIPS*_ILCORR_M.fits*	Coverage map of IC file (representing number of "good" DCES used per pixel)

Chapter 2. Use Cases for the GeRT

The following sections give examples of common use cases for the GeRT. More examples can be found in the Spitzer Data Analysis Cookbook.

2.1 Second-Pass Filtering

The online filtering of MIPS-Ge products can be affected by bright sources and can yield negative "side-lobes" around bright emission regions (e.g., see examples in MIPS Data Handbook). If you see negative side-lobes around the source of interest in the **mfilt.fits* archive mosaic product, you should use the unfiltered **msaic.fits* image or re-filter your data offline using one of the procedures below. Filtering is particularly useful at 70 µm to remove streaking. Users need to first identify the location of bright emission regions to be masked from an initial first pass data reduction (and/or the online data products). In the second pass filtering, bright emission regions are masked to avoid biasing the filtering corrections by sources.

2.1.1 Bright Point Source Filtering at 70 µm

Performs column filtering followed by a high-pass time median filter of the original BCDs, ignoring pixels near the location of a bright point source(s).

- Make input lists of BCDs, uncertainties, and bmasks., e.g.:

```
unix% ls *_bcd.fits > bcd70.lis
unix% ls *_bunc.fits > unc70.lis
unix% ls *_bmask.fits > mask70.lis
```

- Create a file, *source.tbl,* to identify the positions of bright emission to be masked. The input source table needs to be in IPAC table format (use the output of APEX or make by hand or via other software). The values within the IPAC table must be located between the vertical ``|'' to be read properly. e.g., *source.tbl:*

```
|srcid          |RA             |Dec            |
|i              |d              |d              |
 1               XXX.xxxxx       XX.xxxxx
```

  RA,Dec is source position in decimal degrees.

- Copy *source.tbl* into your working directory.

- Edit *mask_pointsource.nl* to point to the proper source list name, e.g.,
  PointSourceList = 'source.tbl',
  (NOTE: the comma at the end of namelist lines is required)

- Run *cleanup70.tcsh*. Performs column filtering followed by a high-pass time median filter of the original BCDs, ignoring pixels near the location of a bright point source(s).

  ```
  $WRAPDIR/cleanup70.tcsh bcd70.lis mask70.lis unc70.lis OUTID
  ```

  where *bcd70.lis* is the list of original BCDs, *mask70.lis* is the bmask list, and *unc70.lis* is the list of uncertainties. The updated filtered BCDs are saved in the output *cc$OUTID* directory. The filtering is controlled by *caler_cleanup70.nl*. The median_count parameter is the high-pass filtering width in DCEs (16 for online processing). "*colfilt = 1*" means apply column filter and "*colfiltfirst = 1*" means apply column filter before high-pass filter. Users may want to test the quality of the filtering corrections on their own data by switching the order of the filters (*colfiltfirst = 1/0*) and modifying the high-pass filter width (*median_count = 10 – 50* DCEs).

- Coadd the updated BCDs using MOPEX or the software of your choice.

2.1.2 Extended Source Filtering at 70 µm (with no column filtering)

Performs high-pass time median filter of the original BCDs. Does not do column filtering since the sizes of extended sources represent a significant fraction of the array.

- Make input lists of BCDs, uncertainties, and bmasks (see §2.1.1).

- Make *source.tbl* to identify the positions of bright emission to be masked. The input source table needs to be in IPAC table format (see §2.1.1).

- Copy *source.tbl* into your working directory.

- Edit *mask_pointsource_extended.nl* to point to the proper source list name and set the size of the mask regions to cover all of the extended area to be masked. e.g., PointSourceList = 'source.tbl',

  ```
  #Mask_Radius takes precedence over MaskBox_X(Y)Size,
  #  Mask_Radius = 3,
  MaskBox_Xsize = 20,
  MaskBox_Ysize = 20,
  ```

Sizes in original pixels, 5x5 box works for point sources, but a larger *MaskBox_*size* should be used to cover extended sources as needed.

- Run *cleanup70_extended.tcsh*. This does a time median filter to remove streaking. Depending on the size-scale of your region of interest, you should modify the median filter window size in *caler_cleanup70_extended.nl*. e.g.,

  ```
  median_count = 30,
  ```

  The online processing median_count = 16 (at 70 um). For large sources/extended regions you may want to use a larger window size.

  ```
  unix%   $WRAPDIR/cleanup70_extended.tcsh   bcd70.lis   mask70.lis
         unc70.lis OUTID
  ```

  where *bcd70.lis* is the list of original BCDs, *mask70.lis* is the bmask list, and *unc70.lis* is the list of uncertainties. The updated filtered BCDs are saved in the output *cc$OUTID* directory.

- Coadd the updated BCDs using MOPEX or software of your choice.

2.2 Bright Source Filtering at 160µm

Performs high-pass time median filter of the original BCDs, ignoring source pixels. Filtering generally works best at 160um for scan data, given that photometry (small-field) does not have enough off-source data to derive good corrections.

- Make input lists of BCDs, uncertainties, and bmasks (see §2.1.1).

- Make *source.tbl* to identify the positions of bright emission to be masked (see §2.1.1).

- Copy *source.tbl* into your working directory.

- Edit *mask_pointsource160.nl* to point to the proper source list name and set the size of the mask regions appropriately, e.g.:

  ```
  PointSourceList = 'source.tbl',
  #Mask_Radius takes precedence over MaskBox_X(Y)Size,
  #  Mask_Radius = 3,
  MaskBox_Xsize = 5,
  MaskBox_Ysize = 5,
  ```

(sizes in original pixels)

- Run *cleanup160.tcsh*. This does a time median filter to remove any residual pixel response variations as a function time. Depending on the size-scale of your region of interest, you should modify the median filter window size in *caler_cleanup160.nl*. e.g.,

  ```
  median_count = 20,
  ```

  A small median_count (=16) is ok for point sources, while larger windows (30-40) can be used for large sources/extended regions. The default online median_count = 20 for 160µm. How the data is taken (e.g., fast scan vs photometry) may affect the proper choice of the 160µm window filter size (median_count).

  ```
  unix% cp $SOS_GeRT/scripts/cleanup160.tcsh .
  unix% cleanup160.tcsh bcd160.lis mask160.lis unc160.lis OUTID
  ```

  where *bcd160.lis* is the list of original BCDs, *mask160.lis* is the bmask list and *unc160.lis* is the list of uncertainties. The updated "two-pass" cleaned-up BCDs are saved in the output *c2c$OUTID* directory.

- Coadd the updated BCDs with MOPEX or the software of your choice.

2.3 Correcting Bad Stim Solutions for Bright Regions

In cases of bright regions, the stim-minus-background solutions may be corrupted, yielding data jumps near the sources (see the MIPS Instrument Handbook for examples). If only one (or a few stims frames) is affected, you can simply delete the "bad" stim frames from the input list and re-run the GeRT to interpolate over this region. You may want to zero-out the affected pixels in the raw stim and/or background ramp (to save most of the stim frame). Online we currently do a simple cubic spline, which smoothly connects all stim measurements. Users may want to play around with the different methods of stim interpolation to see what works best for their data (see the INTERP module). Expert users could also clean-up artifacts in the *interpstim.fits* file offline (remove large jumps and/or smooth the stim solutions) and then re-run the SLOPECAL module.

2.4 Recovering Saturated Data

2.4.1 Saturated Stim (Stim+Sky) Data

Several projects push the saturation limits of MIPS-Ge. In some cases, you may find that your non-stim data are not saturated, but you get NaN's in the BCDs since the stim+sky data did not have enough reads to derive a stim solution. One can get more information out of such data sets with offline reprocessing. The online system ignores the first few reads [4] in the stim ramp due to the slight stim warm-up nonlinearity. Offline, the user could change the parameter *StimLo = 4*, to *StimLo = 1,2,3* to save more stim reads at the beginning of the ramp (within the &CVTIN namelist block in the *MIPS*_SLOPE_0.nl* file). However, this effectively changes the calibration of the data (e.g., stim slopes are lower so BCD values are higher). To constrain the effects on calibration, compare the stim response solutions (*interpstim.fits*) as a function of the number of the initial stim reads ignored, and make the appropriate calibration correction to your data (few--10%).

2.4.2 Saturated Source Data

In cases where the source of interest is strongly saturated, you may try the following. Change *DataLo = 0*, (within &CVTIN namelist block in the *MIPS*_SLOPE_0.nl* file) and change *Min_Num_Samples = 2*, within the &SLOPE namelist block. For online processing, we reject the first read in the data ramp (*DataLo = 1*) and require a ramp segment of 4 good samples for calculating the slope (*Min_Num_Samples = 4*). It is possible to calculate a slope with only 2 reads; however, if you change *Min_Num_Samples < 4*, the RADHIT module will NOT search for radhits within the segment (RADHIT requires 4 reads for checking the end points and for checking for positive vs negative jumps). This technique does not work well for extended regions with low coverage (will show too many radhits). The technique has proven successfully for recovering the core of a saturation source with good redundancy. For 160µm fast-reset data users could also try saving the reads after the reset in the middle of the DCE, i.e., change *number_pixels_to_ignor* within the RESETIN namelist block. Users changing namelist values should check the effects on calibration. In the case of recovering a saturated core, scale the unsaturated wings to match the default processing.

Chapter 3. GeRT Binaries

The binaries are stored in $SIRTF_BIN ($SOS_GeRT/bin).

3.1 CVTI2R4G

Ge-version of CVTI2R4 (which avoids FORTRAN specific compiler issues). Converts integers to real values and makes initial bmask and dmask files. Also marks missing data and saturation in the dmask. The bmask is set to 1 for stim DCEs (STMFL_70, STMFL160 > 0) and 0 for non-stim DCEs (STMFL_70, STMFL160 = 0.0).

IMPORTANT PARAMETERS:

```
&CVTIN
 DataHi = 0,
 DataLo = 1,
 SatHi = 65500,
 SatLo = 10,
 StimHi = 4,
 StimLo = 4,
```

DataHi: number of reads to ignore at end of non-stim DCE

DataLo: number of reads to ignore at start of non-stim DCE

SatHi: DN value for high saturation (set dmask to ignore for higher DN's)

SatLo: DN value for low saturation (set dmask to ignore for lower DN's)

StimHi: number of reads to ignore at end of stim period DCE. The last 4 frames are taken after the stim is turned off.

DataLo: number of reads to ignore at start of stim DCE (ignore stim warm-up period).

3.2 SATURATION

Sets dmask samples with DN values above the values given in the *MIPS*_SAT.fits* calibration file. Saturation can be set to different values for different pixels.

3.3 ELECNL

Applies electronic nonlinearity calibration to ramps as a function of DN using *MIPS*_ENL.fits* calibration file. Uses a cubic spline to interpolate between table values.

3.4 RESET

Checks for resets in the data ramps via header keywords. The reset will occur after the frame given by RSTP160/(2^COADD) for 160µm and RSTP_70/(2^COADD) for 70µm.

IMPORTANT PARAMETERS

```
&RESETIN
 number_pixels_to_ignor = 4,
```

number_pixels_to_ignor: number of reads to ignore after the reset.

3.5 RADHIT

Performs radhit detection using a Bayesian probability technique that checks for ramp discontinuities. A ramp jump above the threshold is declared as a radhit in the dmask and the ramp segments on each side of the jump are checked again for other radhits. The process continues until no more radhits are detected or until the maximum number of hits are detected. The module uses input readnoise and radhit statistics. It is possible to provide an input readnoise calibration file, which takes priority over the Readnoise namelist parameter (to account for possible pixel-to-pixel readnoise variations, e.g., *MIPS*_rnoise.fits*). One can tune up RADHIT separately for stim data (RADHITSTIM block) and non-stim data (RADHIT block).

IMPORTANT PARAMETERS

```
&RADHIT
 FITS_In_Readnoise = ./cal/MIPS70_rnoise.fits,
 Readnoise = 100,
 NominalRHMag = 5,
 RHPriorProb = 0.01,
 DeclThresh = 0.99,
 MaxNumHits = 16,
 NumSamplesMax = 40,
 Gain = 7.1,
 NumDeclareBadAfterRH = 4,
 ThreshDeclareBadAfterRH = 10000,
```

Readnoise: input readnoise in electrons

NominalRHMag: Typical RH mag in terms of x Readnoise (e.g. 5*readnoise)

RHPriorProb: Prior probability for a sample to be hit by a RH.

DeclThresh: Probability threshold for declaration of RH.

MaxNumHits: Maximum number of RHs in ramp before stop searching for RHs.

NumSamplesMax: Maximum number of samples to use in calculation. Longer ramps (e.g., 10sec = 80) are broken into two separate ramps to search for radhits. This is done for speed consideration, since RADHIT inverts a probability matrix the processing goes with ~n^2 instead of n. Breaking into 40 samples does not affect the module's ability to find radhits.
Gain: Conversion between DN to electrons, 7.1e-/DN.

NumDeclareBadAfterRH: Number of reads to ignore after a strong radhit with magnitude larger than ThreshDeclareBadAfterRH.

ThreshDeclareBadAfterRH: DN threshold for declaring reads bad after RH.

3.6 SLOPE

Slope performs a linear fit to the segments defined in the dmask. Requires at least 4 samples for a slope estimation. Standard linear regression is done and the scatter of the fit provides the uncertainty for the segment.

IMPORTANT PARAMETERS

```
&SLOPE
 Min_Num_Samples = 4,
```

Min_Num_Samples: Min number of samples needed for a slope calculation.

If user picks *Min_Num_Samples < 4*, one can fit for fewer samples, but RADHIT requires 4 samples to check the end-points properly.

3.7 FUSION

Does a weighted average of the slopes from the segments to derive final slope. For example, for two slope segments: s1+/-u1 and s2+/-u2, the final slope = wt1*s1 + wt2*s2 where wt1~1/u1^2, wt2~1/u2^2, and the slope uncertainty~([1/u1]^2 + [1/u2]^2)^-0.5.

IMPORTANT PARAMETERS

```
&FUSION
 Negative_Rejection = 3,
 Outlier_Rejection = 20,
```

Negative_Rejection: threshold "sigma" level at which negative slopes are ignored. If slope measurement is < -1*Negative_Rejection, then this slope segment is not included in the slope calculation.

Outlier_Rejection: threshold "sigma" level required for including segments in slope calculation. Some strong radhits can significantly change the responsivity of a detector such that the remaining part of the ramp should be ignored. If the slope measurement after the radhit is more than Outlier_Rejection times sigma different than the measurement before the radhit, the segment after the radhit is ignored. In general, "sigma" for the FUSION module significantly underestimates the true uncertainties in the slopes, so a higher Outlier_Rejection parameter is needed than would otherwise be expected. One could be more aggressive in fusion rejection at the expense of throwing away data.

3.8 MASKWRITE

Copies information from the pmask and dmask to the bmask. Information from the dmask to bmask are copied in cases where data within in ramp are missing, saturated, or contains a radhit.

```
ged_SAMPSMISSING -> geb_SAMPSMISSING
ged_SATURATEHI   -> geb_SATURATED
ged_RADHIT       -> geb_RADHIT
gep_BADHALF      -> geb_BADPIX
gep_BADPIX       -> geb_BADPIX
gep_NOISY        -> geb_NOISY
```

where ged =dmask, gep=pmask, and geb=bmask (see §3.15).

3.9 STACKLAYER

This program constructs a FITScube from single layers. Missing layers have the jam (e.g., NaN) value inserted. An index FITS file may be constructed from keyword values. (e.g. *SCLK_OBS.fits*). Only short, unsigned short (BITPIX = 16, BZERO=32768), and single-precision data types are supported.

USAGE

```
stacklayer -i <prototype layer to specify plane dimensions and type>
           -k  <keyword to create index file>, e.g., SCLK_OBS
           -o  <output fits cube>
           -l  <number of layers in output cube>
           -j  <jam value when nonexistent file>, e.g. NaN
           -d  (prints debug statements)
           -m  <list of conforming single layer file names>
           -v  (verbose output)
```

An input file is used to set the Naxis1 and Naxis2 dimensions. The "-l" parameter gives the Naxis3 dimension for the output cube. The "-j" option is used to fill NaNs for missing input current images and uncertainty images and "-j 0x4000" is used for missing bmasks.

3.10 INTERP

This module makes the stim response function by interpolating between stim-minus-background measurements.

IMPORTANT PARAMETERS

```
   StimVariability = 0.05,
   Comment = 'Method options: S = spline (global fit)',
   Comment = 'X for weighted linear least squares',
   Comment = 'P = piecwise linear (connect the dots)',
   Comment  =  'L  =  least  squares  (Order)  polynomial  fit  (do  not  use  L
   yet)',
   Method = S,
   IntegralWeight = 0,
   NBracket = 2,
   Power = 1,
   Comment = "Flag the pixels for X seconds after stim with the mask",
   FlagAfterStimTime = 11,
   FlagAfterStimMask = 32,
   Comment = " mask bit to mark pixels with extrapolated stims ",
   ExtrapolatedMask = 64,
```

StimVariability: %error associated with individual stim measurements. A 5% error gives reasonable errors for the final BCDs/mosaics.

Method: interpolation method. Spline (S) is used online. "P" simply linearly interpolates between measurements. Only the "S" and "P" methods have been validated. Additional options are available, but have yet been fully tested. "X" is a weighted linear fit that uses NBracket stims on each side of the current DCE and weights as a function of time or DCE number (see details below). A general least-squares polynomial fit ("L") does not work yet. "X" may or may not work properly on your system (new un-validated feature for S14).

FlagAfterStimTime: time in seconds after stim to mask bit FlagAfterStimMask as data near stim warning. Can be used by MOPEX to ignore data near stim in coadd process.

FlagAfterStimMask: 32 (bit 5) bit masked for DCEs near a stim.

ExtrapolatedMask: 64 (bit 6) bit to mask DCES with extrapolated stim solutions.

Note that there are separate namelist blocks depending on the observing mode, such that it is possible to tune up processing based on data types. The modes are determined via the header keywords EXPTYPE and FOVID/APERTURE. The different "modes" are:

SCAN:	scan mode science (SCI) observations, EXPTYPE = scn
PHT:	default-scale SCI photometry, EXPTYPE = pht
FINE:	fine-scale SCI photometry (only applicable for 70um) Science FINE set for EXPTYPE = pht + FOVID > 117
SED:	SED mode SCI observations (only applicable for 70um), EXPTYPE = sed
TP:	TPM mode SCI observations, EXPTYPE = tpm
DARK:	For DARK pipeline processing, EXPTYPE = d2a

EXPTYPES for different IC's are checked via hash table from *w_mips_cdfblock.pl* to choose the proper block.

The interpolation is performed using a table where the y value is stim-background. The background for a stim is the immediate preceding DCE where it exists, or the immediate following DCE if the stim DCE is the first in the FITScube. The uncertainty of the y value in the table is the root-sum-square of the uncertainty in the background, the uncertainty in the stim and a stim-to-stim variation calculated as StimVariability * (stim - bkg). The x value in the table is the time of the stim. Stims may be missing, and a missing stim is simply not entered into the table. In cases of double stims (e.g., when stacking multiple scan legs together), the 2nd stim frame is ignored.

For the SPLINE technique, the independent variable is SCLK_OBS for the DCE. Thus, stims may be missing and the smoothness of the spline fit is relied upon to provide a reasonable interpolation through missing stims. If an interpolation is needed outside the spline table (i.e. before the first stim or after the last stim) , the spline interpolation routine is designed to perform a linear extrapolation.

For the least-squares technique (method = "X"), the chi squared linear fit routine from Numerical Recipes is used. Several tunable parameters are available to control the fit method:

> **NBracket**: the number of stims on each side of the current DCE in fit, note: 0 means use all stims;
>
> **Power:** exponent of the time difference (k);
>
> **IntegralWeight**: = 0 for using the actual times differences as weights, or = 1 for using the ceiling of the time differences (weight as function of the number of Stim DCEs from the current DCE).

Weights are determined by the distance in time between the DCE time and stim time.

Two possibilities for weights calculations are:

If *IntegralWeight = 0* then:

$$wts = \left(\frac{fabs(T - Tstim)}{Tdelt} \right)^{-k}$$

Equation 3.1

and if *IntegralWeight = 1* then:

$$wts = ceil\left(\frac{fabs(T - Tstim)}{Tdelt} \right)^{-k}$$

Equation 3.2

where,
T = time of DCE for stim interpolation
Tstim = time of stimflash itself

Tdelt = interval between stims

Tdelt is calculated as the minimum delta-T between stims in the AOR. Tdelt is irrelevant to the calculation of the least squares coefficients if *IntegralWeight = 0* is chosen, but will allow the least squares weight calculation to be performed less often for the integral difference algorithm (*IntegralWeight = 1*). For example, with integral differences between T1 and T2:

```
time    T0        T1        T2          T3
weight  2**(-k)   1**(-k)   1**(-k)     2**(-k)
```

Note that integral weights may not be useful for photometry AORs because the delta-t between stims varies with the dither patterns.

The number of stims used to calculate the least squares coefficients is 2*NBracket, unless *NBracket = 0* then all stims in the AOR are used to calculate the least squares coefficients.

3.11 SLOPECAL

The SLOPECAL module carries out the calibration and filtering for the MIPS-Ge pipelines. The processing done by the module is controlled by the inputs such that operations are skipped when no inputs are provided.

If a DARK, IC, and FC are given as input, the pipeline performs the following science calibration:

$$I(t) = \frac{FC * \left\lfloor \frac{U(t)}{S * R(t)} - DARK \right\rfloor}{IC}$$

Equation 3.3

where:

I(t)	= unfiltered BCD product, nofiltercube.fits
U(t)	= input slope image, curcube.fits
S*R(t)	= stim response function from INTERP, interpstim.fits
DARK	= dark calibration file from cal/MIPS*_DARK.fits
IC	= IC calibration file from cal/MIPS*_ILCORR*.fits
FC	= flux conversion factor which is given via a cal file.

If no FC and DARK are given as input, then I(t) = U(t)/S*R(t). If no FC and IC are given, then I(t) = [U(t)/S*R(t) - DARK]. If no FC, IC, and DARK are given, then I(t) = U(t), which is used for 2nd pass filtering.

At 70µm and 160µm a high-pass median time filter is done on a pixel basis, and an additional column filter is done for 70µm. The filtered fbcd products form the *calcube.fits* file.

As with INTERP, there are separate namelist blocks depending on the observing mode, such that it is possible to tune up processing based on data types. The modes are determined via the header keywords EXPTYPE and FOVID/APERTURE. The different "modes" are:

SCAN:	scan mode science (SCI) observations, EXPTYPE = scn
PHT:	default-scale SCI photometry, EXPTYPE = pht
FINE:	fine-scale SCI photometry (only applicable for 70µm). Science FINE set for EXPTYPE = pht + FOVID > 117
SED:	SED mode SCI observations (only applicable for 70µm), EXPTYPE = sed
TPM:	TPM mode SCI observations, EXPTYPE = tpm
DARK:	For DARK pipeline processing, EXPTYPE = d2a)

EXPTYPES for different IC's are checked via hash table from *w_mips_cdfblock.pl* to choose the proper block.

IMPORTANT PARAMETERS

```
JanskyScaleFile = cal/MIPS70_fluxconv.tbl,
BUNIT = 'MJy/sr',
median_count = 16,
median_variance_option = 1,
Comment = 'nonzero means column filter is on',
colfilt = 1,
colfiltfirst = 1,
```

JanskyScaleFile: Calibration file for conversion between instrument units to MJy/sr)

median_count: high-pass time filter width in DCEs.

colfilt: only used for 70um. *colfilt = 1* means apply column filter

colfiltfirst: only used for 70 um. *colfiltfirst = 1* means apply column filter before high-pass filter.

Users may want to test the quality of the filtering corrections on their own data by switching the order of the filters (*colfiltfirst = 1/0*) and modifying the high-pass filter width (*median_count = 10 -- 50 DCEs*).

The high-pass median time filter cannot be run stand-alone (requires call to SLOPECAL). The median filter subtracts the median value of the surrounding DCEs on a pixel-by-pixel basis. If the namelist parameter *median_count* is nonzero, then up to *median_count* of samples will be extracted from the FITScube by looking for non-stim and non-NaN values in the following order: -1, +1, -2, +2, -3, +3, -4 +4 ... until *median_count* values have been found. Stims are not affected. Pixels that are NaN's are not affected. The selection process will not go outside the boundaries of the FITScube, which means the median buffer for the last non-stim pixel is the *median_count* non-stim pixels preceding it.

3.12 COLMN_FLTR

Applies column filter for MIPS-70 fbcds. Normally COLMN_FLTR is used as a function within the SLOPECAL module, but it can be run offline stand-alone.

INPUT

a) File: 3-D fits file.
b) File: corresponding bmask.fits.
c) File: corresponding uncertainties fits file.

OUTPUT

a) File: column_filtr.fits (input image after subtracting the column medians (double)).
b) File: column_filtr_uncertainties.fits (file with uncertainties (double)).

DISCUSSION

Column filter should subtract the median of the values of the good pixels in the column for each column for every MIPS-70 filtered-bcd. Bad readout region on good side of the array is ignored by using the information in the bmask.

3.13 GETLAYER

This program extracts a single plane from a fits cube and writes it as a simple fits file. The keywords from the input file are copied, with modifications to the naxis keyword. Only short, unsigned short (BITPIX = 16, BZERO = 32768), and single-precision data types are supported.

USAGE

```
getlayer -i  <input fits cube>
         -k  <input simple fits file with keywords to copy>
         -o  <output simple fits file>
         -z  <bzero>
         -h  <HDU>
         -l  <layer>
         -d  (prints debug statements)
         -m  <multifile list file> overrides -k and -o options
         -v  (verbose output)
```

The header from fits file "-k" is attached to the pixel data for a layer of a data cube "-i". Loop through the cube with index "-l". GETLAYER and STACKLAYER are used by the GeRT to move data in and out of cubes. The uncalibrated slope images need to be stacked into a data cube for calibration and filtering (software was designed this way). After calibration, the cube is split apart into individual BCD products and headers with pointing from the slope images are re-attached.

3.14 ge_mask_pointsource

Copy of the mask_pointsource script from MOPEX. Given a list of point source coordinates either in x,y (then FIF is required), or in RA,Dec decimal degrees, the module masks the user specified bit for the pixels within the user specified circle or box centered on the point sources. The number of point sources projected onto each mask is recorded in the header with the keyword NPSMASKD.

There are two usage modes specified by the *Input_Use* keyword. If *Input_Use* = *'copy_input_data'*, then the output masks are the result of "OR"-ing the input masks and the "point source" bit. If *Input_Use* = *'copy_input_header'*, then the output masks are created from scratch. In this case the data type of the output masks is a single byte (8 bits, BITPIX = 8).

For either usage mode a list of input images is required for the header information, such as image sizes and pointing. If *Input_Use* = *'copy_input_header'*, the input images don't have to be masks. One can use any images, e.g. BCDs, with the relevant header information.

Output file names can be specified either in the *OutputListFileName* or created from the input files file names. In the latter case an *Output_Prefix* should be specified. The output file names are created by replacing the path in the input file names with the *Output_Prefix*. If *Output_Prefix* contains subdirectories names, the subdirectories have to have been created. In order to overwrite the input masks with the output masks set *OutputListFileName* = *ImageListName*. For S14.0 users can use the MOPEX detect image for masking instead of a source list.

See the GeRT cleanup scripts to see how to call binary from the command line.

3.15 mips_ge_mask.h

The MIPS-Ge mask definitions are summarized below (*mips_ge_mask.h* is an include file used in the binaries). The PMASK is a static cal file, while the bmask is made from the processing for each BCD.

S14 PMASK:

Bit	Comments
2	Noisy pixel
3	Bad electronic nonlinearity
8	Bad half of 70um array
14	Bad pixel

S14 BMASK:

Bit	Comments
0	Stimflash DCE
1	
2	
3	Saturated sample(s) found in raw data
4	
5	Data near stimflash warning
6	Stim extrapolation warning
7	
8	Raw DCE has missing data
9	Radhit(s) found in raw data
10	Uncertainty Status
11	Bad pixel as defined by the pmask
12	Slope calibration failed
13	Slope calculation failed
14	Missing layer, NaN and/or bad data

Details with hex bit values.

Naming conventions:

gep prefix	-	pmask file definition
ged prefix	-	dmask file definition
geb prefix	-	bmask file definition
gec prefix	-	cmask file definition

Very often bits have identical semantics in different files because bits get copied from one file to another. It is highly desirable to have spelling identical (as far as it can be) in this case. Thus the prefix is invented.

PMASK (Static pixel mask file)

```
#define gep_NOISY           0x0004
#define gep_BADHALF         0x0100
#define gep_BADPIX          0x4000
```

DMASK

Ramp status bits for each pixel contained in a FITScube (third dimension is sample number in ramp). Produced by sloper pipeline

```
#define ged_DIDRHDET        0x0004
#define ged_SATURATEHI      0x0008
#define ged_SATURATELO      0x0010
#define ged_SEGMENT         0x0100
#define ged_RADHIT          0x0200
#define ged_BADSAMP         0x0800
#define ged_RHBADSAMP       0x2000
#define ged_SAMPSMISSING    0x4000
```

BMASK

Status bits for each pixel contained in an image plane. Produced by sloper pipeline, updated by caler pipeline.

```
#define geb_STIMDCE         0x0001
/* unused                   0x0002 */
#define geb_NOISY           0x0004
#define geb_SATURATED       0x0008
#define geb_USER4           0x0010
#define geb_USER5           0x0020
#define geb_STIMEXTRAP      0x0040
/* unused                   0x0080 */
#define geb_SAMPSMISSING    0x0100
#define geb_RADHIT          0x0200
#define geb_UNCERTSTATUS    0x0400
#define geb_BADPIX          0x0800
#define geb_CALERFAIL       0x1000
#define geb_SLOPERFAIL      0x2000
#define geb_MISSLAYER       0x4000
```

CMASK

Status bits for each pixel contained in an image plane. Produced by a calibration pipeline, supplied by caltrans to pipelines needing the cal file.

```
#define gec_NOISY          0x0004
#define gec_DARKFEW        0x0010
#define gec_ILCORRFEW      0x0020
#define gec_BADHALF        0x0100
#define gec_BADCAL         0x2000
#define gec_BADPIX         0x4000
```

3.16 IMHEADER

The binary imheader is a general tool used to read the headers of fits files.

USAGE

```
imheader  file [files ...]
```

3.17 MIPSFLAT

Module used by IC pipelines to make IC product. For pixels not identified as bad in the mask files, mipsflat takes a trimmed-average per pixel of the dark subtracted data cube to derive the IC value for each pixel. The namelist block control mipsflat is "FLATFIELDIN" within the *MIPS*_FLUXCAL_0.nl*.

IMPORTANT PARAMETERS

```
   cut_post_latent_dce  =  0,
   trim_prcnt           =  48,
   trim_prcnt_global    =  50,
```

cut_post_latent_dce: the number of DCEs after the stim flash DCE to ignore.

trim_prcnt: the +/- trimmed percentage of DCEs from the wings of the distribution before. trim_prcnt = 50 gives a median.

trim_prcent_global: controls the normalization of the IC product (*trim_prcnt_global = 50* gives median). Currently, the IC is normalized to the array median of the remaining good (non-bad and non-noisy pixels and non-NaN) pixels defined in the PMASK.

3.18 MIPSDARK

Module used by the DARK pipeline to make the DARK product. For pixels not identified as bad in the mask files, mipsdark takes a trimmed-median per pixel of the data cube after the stim response correction to derive the DARK value for each pixel. The namelist block control mipsdark is "MIPSDARKIN" within the *MIPS*_FLUXCAL_0.nl*.

IMPORTANT PARAMETERS

```
Cutoff_Outlrs = 2.5,
Outlrs_Percent = 5.0,
Skip_Post_DCE = 0,
Nan_Percent = 10.0,
```

Cutoff_Outlr: the sigma value about the median that is clipped before calculating the median per pixel.

Outlrs_Percent: the percentage threshold for the number outlier values per pixel before flagging in the output CMASK file.

Skip_Post_DCE: the number of DCEs after the stim flash DCE to ignore.

Nan_Percent: the NaN percentage threshold for the number NaN values per pixel before flagging in the output CMASK file.

Chapter 4. GeRT Scripts

The scripts are stored in $WRAPDIR ($SOS_GeRT/scripts).

4.1 gert*.pl

Main Perl script for running the GeRT described in sections 1-4. *gert.pl* is setup to run without Perl threads. If Thread.pm is available, use *gert_thread.pl* for increased speed. *gert_thread_rm.pl* shows a modified *gert_thread.pl* script that deletes the intermediate SLOPER products to save disk space.

4.2 cleanup*.tcsh

The *cleanup*.tcsh* scripts are described in Chapter 2. These scripts mask sources and then calculate the filtering corrections and make new filtered BCDs.

USAGE

```
unix% cleanup70.tcsh bcd70lis mask70.lis unc70.lis lirg70
```

INPUT

1. Before running the script, an input list of sources for masking in the *mask_pointsource.nl* must have been set up, e.g., *PointSourceList = 'source.tbl'.* See Chapter 2 for details.
2. *bcd70.lis*, *mask70.lis* and *unc70.lis* are the nput lists of BCDs, bmasks, and uncertainties, respectively.
3. *lirg70* is the output directory. The cleaned BCDs will be writted to *cclirg70/.*

DISCUSSION

The script deletes intermediate files.

- Attaches header pointing information to Bmask files by running *bcd2cube.pl* (stacklayer) and then *cube2bcd.pl* (getlayer). This is not the most efficient way to do this, but the scripts are already developed.

- Updates the Bmask files by masking out the location of the sources using *ge_mask_pointsource* script and controlled by the *mask_pointsource_*.nl* namelist file.

- Stacks the data into cubes for filtering using *bcd2cube.pl* (stacklayer).

- Runs column and high-pass time filtering on cube via slopecal with input namelist *caler_cleanup*.nl*. Bad pixels and source pixels are not included in the filtering corrections.

- Output cleaned up BCDs put in *cclirg70/*.

- See Chapter 2 for examples.

4.3 bcd2cube.pl

Offline BCD to cube Perl function.

USAGE

```
unix% bcd2cube.pl INlist CUBE
```

where
INlist: input list of original BCDs
CUBE: Output fits data cube

Uses system call to the downlink stacklayer binary (BCD headers are dropped).

4.4 cube2bcd.pl

Offline cube to BCD Perl function.

USAGE

```
unix% cube2bcd.pl INlist CUBE OUTdir
```

where
INlist: input list of original BCDs
CUBE: Clean fits data cube
OUTdir: Output directory of new BCDs

The headers are from the INlist files and the pixel values are from CUBE; uses a system call to the downlink binary *getlayer*.

4.5 mips/w_mips_rfitshead.pl

Perl script used by software to read FITS headers and put keywords into a hash table for processing.

4.6 mips/w_mips_cdfblock.pl

Perl script called from within *gert.pl* that picks the proper namelist block name based on keywords via calling yet another Perl script (*w_mips_ensemble_files.pl*).

FINE-scale photometry observations are picked for exptype = pht and aperture/FOVID > 117 (aperture -> fovid after online FPG), and exptype set to ffl for pfl and aperture > 117 (FINE-scale IC).

4.7 mips/w_mips_ensembles_files.pl

Perl script with hash tables mapping caltype with exptype.

```
# hash tables giving mapping from exptype to caltype
# must return an empty value for non calibration exposure types

# Hash to look up type from exptype
%exptypetab =
 (
 "scn" => "SCAN",
 "pht" => "PHT",
 "tpm" => "TPM",
 "d2a" => "DARK",
 "d2b" => "DARK",
 "f2a" => "ILCORR",
 "f2b" => "ILCORR",
 "fs"  => "ILCORRSED",
 "sed" => "SED",
 "d3"  => "DARK",
 "f3"  => "ILCORR",
 "sfl" => "SCAN",
 "ffl" => "FINE",
 "tfl" => "TFM",
 "pfl" => "PHT",
 "dfl" => "SED"
```

4.8 mk_*gert.tcsh

Scripts used to make public GeRT packages from SSC downlink areas. Used by internal SSC folks. Users can ignore this script. .