MIPS Ge Pipeline Description Document

Version 1.1 - August 2010

1. Introduction

This document is intended for science users of the Multiband Imaging Photometer for Spitzer (MIPS) as a guide to the data processing pipelines developed by the Spitzer Science Center. For a description of the arrays and the MIPS instrument, please refer to the MIPS chapter in the Spitzer Observer's Manual and the MIPS Data Handbook. Additional information relating to the data reduction algorithms employed in the various pipeline modules is also presented in Gordon et al. 2005 (PASP, 117, 503).

2. The MIPS Ge Pipelines

2.1. Overview

The particular pipeline and calibration files that are chosen for a given data set depend on the channel number (CHNLNUM = 2 for 70μm, CHNLNUM = 3 for 160μm) and the exposure type (header keyword EXPTYPE). The different pipeline possibilities are:

EXPTYPE = scn $\Longrightarrow$ Scan data, normal science pipeline is run.
EXPTYPE = pht $\Longrightarrow$ 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 $\Longrightarrow$ 70μm SED data, SED pipeline is run.
EXPTYPE = tpm $\Longrightarrow$ Total power mode (TPM) data, TPM pipeline is run.
EXPTYPE = d2a $\Longrightarrow$ DARK data, DARK pipeline is run.
EXPTYPE = sfl $\Longrightarrow$ Illumination Correction (IC) data from scanning, IC pipeline is run.
EXPTYPE = pfl $\Longrightarrow$ IC data from photometry, IC pipeline is run.
EXPTYPE = ffl $\Longrightarrow$ IC data from FINE-scale photometry, IC pipeline is run.
EXPTYPE = dfl $\Longrightarrow$ SED IC data, IC pipeline is run.
EXPTYPE = tfl $\Longrightarrow$ Total power IC data, IC pipeline is run.

In the following we outline the science pipelines that produce basic calibrated data (BCDs). 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, read every 1/8 sec). The cdf file controlling "SLOPER" processing is MIPS70/160_SLOPE_0.nl. The "CALER" processing is controlled via the cdf file MIPS70/160_FLUXCAL_0.nl.

2.2. MIPS Ge Science BCD Pipeline

An overview of the various steps in the pipeline processing is presented in Figure 1. The individual pipeline modules are described below. The namelist input block for each module is also provided.

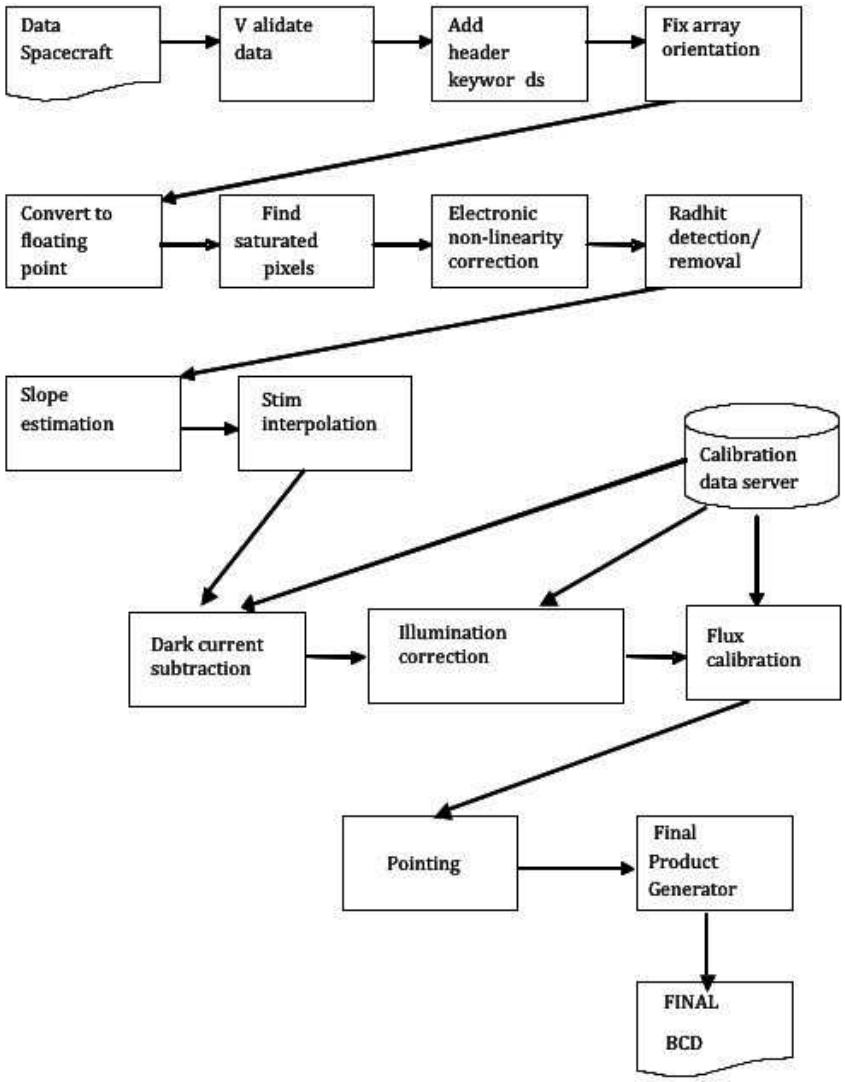

Fig. 1.— MIPS Ge:Ga Pipeline Overview.

2.2.1. TRANHEAD

The *tranhead* pipeline module translates and simplifies the DCE fits file header delivered by JPL/FOS in Multimission Image Processing Laboratory (MIPL) format to a standardized format for use in the pipelines. The data remain untouched by this process.

The instrument telemetry data stored in the MIPL header under their coded telemetry keywords are translated into alphanumerical mnemonic labels using a keyword dictionary. Derived keywords, such as the total integration time (EXPTIME), are computed from the data in the MIPL header and added to the raw.fits header. In addition, entries from the SSC operations database are written to the raw.fits header, e.g. the observer's name (header keywords OBSRVR) and the program ID (header keyword PROGID).

&TRANHEADIN
Comment = 'MIPS70/160 tranhead namelist block NO STRING SUBSTITUTIONS',
CentralFraction1 = 1.0,
CentralFraction2 = 1.0,
HeaderCommentFlag = 2,
CE_Side = 1,
&end

2.2.2. SWAP

The *swap* module is only part of the 160μm data processing pipeline. It reorders the pixel locations from the 28×2 input array to a 20×3 array layout. The 20×3 layout correctly represents the footprint of the array on the sky.

&SWAPIN
Comment = 'Generic namelist file for swap, default values.',
FITS_Image_Filename1 = 'tranraw.fits',
FITS_Out_Filename = 'transwap.fits',
Log_Filename = 'stdout',
&end

2.2.3. IMFLIPROT

The module *imfliprot* performs an image reorientation to ensure that the arrays have the expected FOV orientation. For both the 70 and 160μm array a flip about the vertical axis is applied.

&IMFLIPROTIN

Comment = 'imfliprot namelist block.
Log_Filename = 'stdout',
Comment = 'Flip: 1 = Flip, 0 = No Flip (in x, top left pixel is reference)',
Flip = 1,
Comment = 'Rotate: 0, 90, 180, 270 in clockwise rotation',
Rotate = 0,
&END

2.2.4. CVTI2R4

The *cvti2r4* module converts the input integer data values into real values. The module also creates the initial bmask and dmask files and 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).

&CVTIN
DataHi = 0,
DataLo = 1/2, for 70/160
SatHi = 65500/61000, for 70/160
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).

2.2.5. SANITY CHECK

Before the data proceed through the pipeline, they are checked to ensure that they are the type of data expected. In particular, header keywords are checked against their expected values, e.g. the dimensions of the data arrays for a specific channel number (see the tables below). At this point in the pipeline, the 70/160μm data are data cubes with the x and y dimensions corresponding to the size of the array (32×32 pixel and 20×3 pixel, respectively),

and the third dimension determined by the exposure time, i.e each sample that is read is stacked in succession to form a data cube with 24, 32, or 80 layers. Following one x,y pixel through the cube will reveal the data ramp for that pixel.

```
&SANITYDATATYPEIN
FITS_Image_Filename = ./tranhead.fits,
Data_In_Filename_Data1_In = ./cdf/MIPS70/160_SANCHK_0.tbl,
Data_In_Filename_Data2_In = ./cdf/MIPS70/160_SANINT_0.tbl,
Data_Out_Filename_Data1_Out = ./qa/sanity_out.tbl,
Data_Out_Filename_Data2_Out = ./qa/interp_out.tbl,
Log_Filename = 'stdout',
&end
```

MIPS70/160_SANCHK_0.tbl:
For MIPS Ge Slope Estimate Thread
Description for the column:
column "keyword" specifies the KEYWORD in FITS header to look up.
column "valid_test" specifies the criteria for the validation, i.e.
the criteria not to set up invalid flag.
column "valid_value" specifies the values for valid_test.

index	keyword	test	value
1	INSTRUME	=	MIPS
2	NAXIS	=	3
3	NAXIS1	=	32
4	NAXIS1	=	20
5	NAXIS2	=	32
6	NAXIS2	=	3
7	NAXIS3	>	2
8	CHNLNUM	=	2
9	CHNLNUM	=	3
10	DCENUM	≥	0
11	EXPID	≥	0
12	PIPENUM	≥	300
13	PIPENUM	<	400
14	MISSDATA	=	T
15	MANCPKT	>	0

MIPS70_SANINT_0.tbl
For MIPS Ge Slope Estimate Thread
Description of the columns:

column "status" specifies the the condition user wants to set.
column "criteria" specifies the criteria for each status
| status | criteria |
normal 1 && 2 && 7 && 10&& 11
ok-NAXISs (3&&5) || (4&&6)
ok-chnlnum 8 || 9
ok-pipenum 12 || 13

2.2.6. SATURATION

The *saturation* module sets bits in the dmask for saturated samples inside the data cubes, i.e. samples with DN values above the values given in the MIPS70/160_SAT.fits calibration file. Saturation can be set to different values for different pixels. For MIPS70 the saturation level for all pixels is set to 65500 in the calibration file, for MIPS160, the level is 61000 for all pixels.

&SATURATION
Comment = Saturation namelist file with default values.
Comment = Input charge ramp in dn,
FITS_In_Filename1= ./dce.fits,
Comment = Saturation level in dn for each pixel,
FITS_In_Filename2= ./cal/MIPS70/160_SAT.fits,
FITS_In_Filename3= ./dmask.fits,
FITS_Out_Filename1=qa/sat.fits,
FITS_Out_Filename2=./dmask_sat.fits,
Log_Filename = 'stdout',
Readnoise = 140/800, for 70/160
A2D_Bias = 0,
&end

2.2.7. ELECNL

This module corrects for the effects of electronic non-linearity (not to be confused with flux non-linearity effects). An electronic non-linearity calibration is applied to the ramps as a function of DN using the MIPS70/160_ENL.fits calibration file. The non-linearity is corrected by multiplication of the DN value of each sample by a factor obtained from a lookup table. The non-linearity is a function of DN value only. A cubic spline is used to interpolate between the calibration table values.

&MIPSNLSPLINE
Comment = "Applies nonlinearity calibration to charge ramp samples",
RampIn= ./dce.fits,
RampIn_unc = NONE,
RampOut = ./enlcorrd.fits,
RampOut_unc = ./enlcorrd_unc.fits,
SplineTable_fits = ./cal/MIPS70/160_ENL.fits,
Log_Filename = 'stdout',
&end

2.2.8. RESET

The *reset* module checks for extra resets in the data ramps via header keywords and sets the dmask appropriately. The reset will occur after the frame given by RSTP160/(2^{COADD}) for 160μm and RSTP_70/(2^{COADD}) for 70μm.

&RESETIN
Comment = Mark dmask with position of reset segments.',
FITS_Image_Filename1 = 'tranhead.fits',
dmask_Filename = './dmask_sat.fits',
dmask_Out_Filename = './dmask_rhit.fits',
number_pixels_to_ignor = 4,
reset_period_keyword = 'RSTP_70'/'RSTP160',
coadd_keyword = 'COADD',
Log_Filename = 'stdout',
&end

2.2.9. RADHIT

The module *radhit* checks for radhits in the ramps and sets the dmask at the location of radhit(s). The SSC pipeline identifies ramp discontinuities using a maximum likelihood technique (Hesselroth et al. 2000, Spaceborne Infrared Remote Sensing VIII, SPIE Conference Proceedings, 4131, 26). Radhits are assumed to cause discontinuous jumps in charge at particular samples. The radhit detector first finds the sample number with the highest likelyhood of having a radhit, then uses Bayes Rule to compute the probability that a radhit actually occurred there. In order to account for readout electronic glitches which may induce negative jumps, both positive and negative discontinuities are considered. The probability is then compared to a threshold and if a radhit is found, the algorithms are performed again on the ramp segments on each side of the jump. The process is iterated until the maximum

number of radhits are detected, no radhits are found or there are no segments long enough (4 samples) to perform the algorithms. The radhits are tracked by setting bits in the dmask file. 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., MIPS70/160_rnoise.fits). One can tune up RADHIT separately for stim data (RADHITSTIM block) and non-stim data (RADHIT block).

&RADHIT
Comment = Generic namelist file for radhit default values.,
Comment = DEI,
FITS_In_Charge_Ramp = ./enlcorrd.fits,
FITS_In_dmask = ./dmask_rhit.fits,
FITS_In_PMASK = ./cal/MIPS70/160_PMASK.fits,
FITS_In_Readnoise = ./cal/MIPS70/160_rnoise.fits,
FITS_Out_Radhit_Probs = NONE,
FITS_Out_Radhit_Samples = qa/rhits.fits,
FITS_Out_Radhit_Mags = qa/rhmags.fits,
FITS_Out_Slope = NONE,
FITS_Out_dmask = ./dmask_rhit.fits,
Log_Filename = 'stdout',
Readnoise = 100/700,
NominalRHMag = 5,
RHPriorProb = 0.01,
DeclThresh = 0.99,
MaxNumHits = 16,
NumSamplesMax = 40,
Gain = 7.1,
NumDeclareBadAfterRH = 4/100,
ThreshDeclareBadAfterRH = 10000/5000,
&end

&RADHITSTIM
Comment = RADHIT namelist block- stimflash only ,
Comment = DEI,
FITS_In_Charge_Ramp = ./enlcorrd.fits,
FITS_In_dmask = ./dmask_rhit.fits,
FITS_In_PMASK = ./cal/MIPS70_PMASK.fits,
FITS_Out_Radhit_Probs = NONE,
FITS_Out_Radhit_Samples = qa/rhits.fits,
FITS_Out_Radhit_Mags = qa/rhmags.fits,
FITS_Out_Slope = NONE,

FITS_Out_dmask = ./dmask_rhit.fits,
Log_Filename = 'stdout',
Readnoise = 100/800,
NominalRHMag = 5/6,
RHPriorProb = 0.01,
DeclThresh = 0.99,
MaxNumHits = 16,
NumSamplesMax = 40,
Gain = 7.1,
NumDeclareBadAfterRH = 4/100,
ThreshDeclareBadAfterRH = 10000/20000,
&end

Readnoise = input readnoise in electrons
NominalRHMag = Typical RH mag in terms of × 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 $\sim n^2$ 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.

2.2.10. SLOPE

The *slope* module calculates a linear fit to all slope segments with at least four consecutive good reads as indicated by the dmask. Standard linear regression is performed and the scatter of the fit provides the uncertainty for the segment. While the minimum number of good reads can in principle be set to lower values by adjusting Min_Num_Samples, the *radhit* module requires four samples to properly check the end-points of the ramps.

&SLOPE
Comment = Slope namelist file with default values.,
FITS_In_Filename1= ./enlcorrd.fits,
FITS_In_Filename2= ./dmask_rhit.fits,
FITS_In_Filename3= ./enlcorrd_unc.fits,
FITS_Out_Filename1= ./currents.fits,

FITS_Out_Filename2= NONE,
FITS_Out_Filename3= NONE,
FITS_Out_Filename4= ./currents_unc.fits,
Log_Filename = 'stdout',
Min_Num_Samples = 4,
Layer = 1,
Comment = Convert from MIPS Ge samples to seconds,
Constant = 7.62939453125,
&end

2.2.11. FUSION

The multiple slopes for each pixel calculated in the previous module are combined into a single slope value by the *fusion* module. The *fusion* module calculates a noise-weighted average slope image from the fitted segments based on the empirical errors estimated from the scatter of the data within the ramp segments. For example, for two slope segments, s1±u1 and s2±u2, the final slope image is wt1×s1 + wt2×s2 where wt1 $\sim 1/(u1)^2$, wt2 $\sim 1/(u2)^2$, and the slope uncertainty is $\sim ([1/u1]^2 + [1/u2]^2)^{-0.5}$.

The output of *fusion* is a two dimensional image and an associated uncertainty image.

&FUSION
Comment = Generic namelist file for fusion default values.,
FITS_In_Filename1= ./currents.fits,
FITS_In_Filename2= ./currents_unc.fits,
FITS_In_Filename3 = ./bmask_dce.fits,
FITS_Out_Filename1= ./current.fits,
FITS_Out_Filename2= ./current_unc.fits,
FITS_Out_Filename3 = ./bmask.fits,
Log_Filename = 'stdout',
Max_Num_Values = 1,
Negative_Rejection = 3,
Outlier_Rejection = 20,
&end

Negative_Rejection = threshold "sigma" level at which negative slopes are ignored. If the 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 the 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 × 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.

2.2.12. MASKWRITE

This module copies information from the pmask and dmask to the bmask. Information from the dmask to the bmask is copied in cases where data within a ramp are missing, saturated, or contain a radhit. This is the last step in the "SLOPER" part of the pipeline.

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.

&MASKWRITE
Comment = maskwrite namelist file with default values.,
FITS_In_Filename1= ./dmask_rhit.fits,
FITS_In_Filename2= ./bmask.fits,
FITS_In_Filename3= ./cal/MIPS70_PMASK.fits,
FITS_Out_Filename1=./bmask.fits,
Log_Filename = stdout,
Comment = bits when any set in pmask set all corresponding bits in bmask,
pmask_from_0 = 0x0000,
bmask_to_0 = 0x0000,
pmask_from_1 = 0x0000,
bmask_to_1 = 0x0000,
pmask_from_2 = 0x0000,
bmask_to_2 = 0x0000,
pmask_from_3 = 0x0000,
bmask_to_3 = 0x0000,
&end

2.2.13. INTERP

This module computes the stim response function by interpolating between stim-minus-background measurements. The input fits file to *interp* is a multi-layer cube consisting of the previously computed slope estimates at each DCE (layer) for each pixel for a given AOR. Interp extracts the stimflash data based on the information in the bmask. The stimflash slope is subtracted from that of the previous DCE (which has the same background) to yield the stimflash-minus-background slope. The different stimflash-minus-background slope measurements for all the stims are then interpolated as a function of time (SCLK_OBS) to provide the stim response function for each pixel of each DCE.

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 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

As an example, the namelist block of the SCAN mode *interp* input is shown here:

&INTERP_SCAN
Comment = 'Generic namelist file for interp- SCAN ',
FITS_In_Filename = 'current.fits',
FITS_Time = 'SCLK_OBS.fits',
FITS_Sigma_In_Filename = 'current_unc.fits',
FITS_Bmask_In_Filename = 'bmask_current.fits',
FITS_Out_Filename = 'interpstim.fits',
FITS_Err_Out_Filename = 'interpstim_unc.fits',
FITS_BMask_Out_Filename = 'bmask_interp.fits',
Log_Filename = 'stdout',
Fit = 0,
StimVariability = 0.05,
CauchyThreshold = 0.4,
ApproxThreshold = 0.2,
Comment = 'ignore second stim in a double stim sequence',
IgnoreDCE0Stim = 1,

Comment = 'Method options: S = spline (global fit)',
Comment = 'X for weighted linear least squares',
Comment = 'P = piecewise 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,

Comment = " mask bit meaning stim or background not present in data set ",
MissingStimMask = 0,

Comment = " mask bit meaning default stim came from a file ",
DefStimMask = 0,

Comment = "mask bit meaning no stims at all. interpstim=1 for all pixels ",
MissingAllStimsMask = 0,

&end

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

Method gives the 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 not yet been fully tested. "X" is a weighted linear fit which uses NBracket stims on each side of the current DCE and weights as a function of time or DCE number. "L" has not been implemented.

FlagAfterStimTime = time in seconds after stim to mask bit.
FlagAfterStimMask = 32 (bit 5) bit masked for DCEs near a stim. Can be used by mopex to ignore data near stim in coadd process.

ExtrapolatedMask = 64 (bit 6) bit to mask DCEs with extrapolated stim solutions.

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 - determines 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.

2.2.14. 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:

$$I(t) = FC * [U(t)/S * R(t) - DARK]/IC$$

where
I(t) = unfiltered BCD product
U(t) = input slope image

S*R(t) = stim response function from INTERP,
DARK = dark calibration file MIPS70/160_DARK.fits
IC = IC calibration file MIPS70/160_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.

For both the 70μm and 160μm a high-pass median time filter is applied on a pixel basis, and an additional column filter is applied for 70μm, resulting in the filtered fbcd products.

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 which also ensures that the appropriate calibration files (dark, flat) for a given mode are used. 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)

As an example, the namelist block of the SCAN mode *interp* input is shown here:

```
&SLOPECAL_SCAN
Comment = 'Generic namelist file for slopecal, SCAN instrument mode.',
JanskyScaleFile = cal/MIPS70_fluxconv.tbl,
BUNIT = 'MJy/sr',


CauchyThreshold = 0.4,
ApproxThreshold = 0.2,


obs_fitscube = 'current.fits',
dark_fits = 'cal/MIPS70_DARK.fits',
aorstim_fitscube = 'interpstim.fits',
skyflat_fits = 'cal/MIPS70_ILCORR.fits',
```

obs_sigma_fitscube = 'current_unc.fits',
dark_sigma_fits = 'cal/MIPS70_DARK_U.fits',
aorstim_sigma_fitscube = 'interpstim_unc.fits',
skyflat_sigma_fits = 'cal/MIPS70_ILCORR_U.fits',

obs_mask_fitscube = 'bmask_interp.fits',
dark_mask_fits = 'cal/MIPS70_DARK_C.fits',
aorstim_mask_fitscube = 'NONE',
skyflat_mask_fits = 'cal/MIPS70_ILCORR_C.fits',

FITS_Out_Filename = 'cal_current.fits',
FITS_Nofilter_Out_Filename = 'nofilter_current.fits',
FITS_Nofilter_Err_Out_Filename = 'nofilter_current_unc.fits',
FITS_Err_Out_Filename = 'cal_current_unc.fits',
FITS_BMask_Out_Filename = 'bmask.fits',

median_count = 16,
median_variance_option = 1,

FITS_Darksubt_Out_Filename = 'MIPS70_darksubt.fits',
FITS_Darksubt_Err_Out_Filename = 'MIPS70_darksubt_unc.fits',
FITS_Darksubt_bmask_Out_Filename = 'MIPS70_darksubt_bmask.fits',

Comment = 'nonzero means column filter is on',
colfilt = 1,
colfiltfirst = 1,

Log_Filename = 'stdout',
&end

median_count indicates the high-pass time filter width in DCEs. 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 nonstim 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 nonstim pixel is the "median_count" non-stim pixels preceding it.

colfilt is only used for 70μm. colfilt=1 indicated that the column filter is applied and and colfiltfirst=1 means that the column filter is applied before the high-pass filter. The column filter subtracts the median of the values of the good pixels in the column for each column for every MIPS70 BCD. The bad readout region on the good side of the array is ignored by using the information in the bmask.

2.2.15. Pointing Transfer and FPG

The *pointing transfer* pipeline is a separate script from the data reduction pipeline script, designed to insert pointing and distortion information into the FITS headers of BCDs. It operates on a per BCD basis.

The Final Product Generator (FPG) is executed at the end. The FPG reformats the FITS header, renaming certain keywords and adding additional keywords from the database.

2.3. RAW Pipeline

The raw.fits images are produced by running the *tranhead* and *imfliprot* modules for 70μm and *tranhead*, *swap* and *imfliprot* for 160μm data, followed by the pointing transfer thread and the FPG. Note that the raw data are still data cubes.

2.4. DARK Pipeline

Dark data is obtained with dedicated AORs during each Spitzer Campaign. These calibration data are processed by the pipeline in a very similar manner to that of the science data. In particular, the data proceed through all of the modules described above up to and including *slopecal*. Within slopecal, only the stim response function is applied for dark data. The *slopecal* namelist block is shown here:

&SLOPECAL_DARK
Comment = 'namelist file for slopecal when performing dark cal product pipeline.',
BUNIT = 'MIPS70_DARK',

CauchyThreshold = 0.4,
ApproxThreshold = 0.2,

obs_fitscube = 'current.fits',
dark_fits = 'NONE',
aorstim_fitscube = 'interpstim.fits',
skyflat_fits = 'NONE',

obs_sigma_fitscube = 'current_unc.fits',
dark_sigma_fits = 'NONE',
aorstim_sigma_fitscube = 'interpstim_unc.fits',
skyflat_sigma_fits = 'NONE',

obs_mask_fitscube = 'bmask_current.fits',
dark_mask_fits = 'NONE',
aorstim_mask_fitscube = 'NONE',
skyflat_mask_fits = 'NONE',

#nofilter product becomes bcd and filtered product is currently called nofilter,
FITS_Out_Filename = 'nofilter_current.fits',
FITS_Nofilter_Out_Filename = 'cal_current.fits',
FITS_Err_Out_Filename = 'cal_current_unc.fits',
FITS_BMask_Out_Filename = 'bmask.fits',

median_count = 0,

Log_Filename = 'stdout',
&end

The dark data output from slopecal subsequently enters the *mipsdark* module, which combines the individual darks obtained in the AOR. A median-filter algorithm is applied to the input cube on a pixel by pixel basis. Bad pixels for each DCE are flagged by the input bmask and not used in the determination of the median for a given pixel. All pixel values are examined to see if they are outliers, based on the value of the Cutoff_Outlrs parameter in the namelist block. Skip_Post_DCE determines the number of DCEs immediately following the stimflash frame that will be rejected to minimize post stim transients.

&MIPSDARKIN
Comment = 'Generic namelist file for DARK, default values.',
FITS_Image_Filename = 'cal_current.fits',
FITS_Bmask_Filename = 'bmask.fits',
FITS_Out_Filename = 'MIPS70/160_DARK.fits',

FITS_Out_Uncer_Filename = 'MIPS70/160_DARK_U.fits',
FITS_Out_Cmask_Filename = 'MIPS70/160_DARK_C.fits',
FITS_Out_Covmap_Filename = 'MIPS70/160_DARK_M.fits',
Cutoff_Outlrs = 2.5 ,
Outlrs_Percent = 5.0 ,
Skip_Post_DCE = 0 ,
Nan_Percent = 10.0 ,
Log_Filename = 'stdout',
&end

2.5. IC Pipeline

Illumination Correction (IC) data is obtained with dedicated AORs for each separate mode (scan, phot, fine, sed, tpm) at various intervals throughout the Spitzer mission. Since such data are acquired by simply observing an appropriate patch of the sky, these observations can be processed as both science and/or IC data. To produce an IC calibration product, the data proceed through all of the modules described above, up to and including *slopecal*. Within slopecal, a "_darksubt" output file is generated for each mode that can subsequently be passed to the *flatfield* module for IC processing. A median-filter is applied to the input cube on a pixel by pixel basis. Bad pixels for each DCE are flagged by the input bmask and not used in the determination of the median for a given pixel. While the IC calibration output files produced by the *flatfield* module were utilized for trending during the Spitzer mission, they were never directly applied as calibrations for the science data. The IC calibration files used for the science data were produced offline using a large number of IC data from multiple campaigns.

&FLATFIELDIN
Comment = 'namelist file for mips_flatfield',
FITS_Image_Filename1 = 'MIPS70/160_darksubt.fits',
bmask_Filename = 'MIPS70/160_darksubt_bmask.fits',
pmask_Filename = 'cal/MIPS70/160_PMASK.fits',
FITS_Out_Filename = 'MIPS70/160_ILCORR.fits',
FITS_Out_Filename_2 = 'MIPS70/160_ILCORR_U.fits',
FITS_Out_Filename_3 = 'MIPS70/160_ILCORR_C.fits',
FITS_Out_Filename_4 = 'MIPS70/160_ILCORR_M.fits',

cut_post_latent_dce = 0,
trim_prcnt = 48,
trim_prcnt_global = 0,

```
Log_Filename = 'stdout',
&end
```