Frequently Asked Questions on MIPS-24μm Processing

Frank Masci
fmasci@ipac.caltech.edu

Version 1.0
June 15, 2005

Hi, I'm Theo, and I'm not responsible for errors in this document, the author is!

FAQ Summary

1. Where is the latest mips24 offline tar-ball kept?
2. Where can I find a description and flowcharts of all mips24 pipelines?
3. Where can I find the latest bit-mask definitions for all types of masks involved in mips24 processing?
4. What are all the mips24 science-related pipeline products that we are archiving (but not necessarily distributing to users)?
5. How do I measure the x, y spot position in a list of flats or DCEs for trending and analysis?
6. What's the procedure to correct for pointing in all MIPS modes in a campaign due to scan-mirror wobble?
7. Is there any simple way to find the closest campaign(s) to a specific campaign of interest according to spot position?
8. How do I make scan-mirror position dependent flats offline?
9. Why do there appear to be duplicate flatfield products (exactly the same) for mirror positions separated by ~0.5 DAC bins?
10. How do I make synthetic (or interpolated) flats from a set of real scan-mirror dependent flats?
11. Why do some synthetic (or interpolated) flats contain no spot positions in their filename or header?
12. What are all the different ways we can flatfield DCEs on ops?
13. What happens if I (pipeline operator) am undergoing PAO and reprocessing at the same time for MIPS-1. Can I use two different flat-picking mechanisms from above?
14. How do I load new calibration products into the SODB:mips1fallback table?
15. Do mips24 calibration products have to have some special header keywords/format to be used in D/L pipelines?
16. Where are all the latest namelists kept for mips24?
17. How do I query for "super-flats" made from dedicated scan-mirror dependent flats in online operations?
18. Are the flatfields that are made from the spot-map and super-flat on a DCE basis being released to the user? Are they being archived so that the user can access them later?
19. Can I get a summary of how many BCDs were actually "spot-flattened", "mirror"-flattended, or were gradient-corrected on ops?
20. By simply looking at a science BCD header, how can I tell what calibration files were used to process that DCE.
21. Are any calibration filenames written to the BCD header?
22. What's the maximum number of images allowed for input into the "flatfield" module?
23. Why do I still see spots in photometry-mode mosaics after flat-fielding using spot-dependent flats?
24. What are all the mips24 threads currently in operations?
25. In what order do all the different mips24 threads run on ops for a given starting "initiplscriptId"?
26. How does saturation thresholding work in pipelines and what's the deal with pixel replacement in the SUR-science pipeline?
27. Is there a command-line tool to flip and rotate a list of FITS images?
28. Is there a tool which gives a summary of processing on a single pixel (i.e., history in temporal order) for all intermediate FITS files generated in a pipeline?
29. How do I "self-flatten" BCDs (or self-calibrate) prior to mosaicking?
30. I have a FITS image containing WCS information. Is there a script where I supply a list of RA, Decs and it will report the x, y locations and fluxes of all pixels?
31. Suppose I have a list of FITS images (with WCS) which may be overlapping. Is there a script where I supply a RA, Dec and it will report the x, y locations and fluxes of all pixels from all images in the stack?

32. Is there any command-line tool where I just supply a FITS image and it returns all 2MASS sources contained therein?
33. Is there a command-line tool which reports a quick and dirty measure of the RMS noise in an image and other statistics?
34. Is there a command-line tool which crops FITS images to whatever size you like and saves it to a new FITS image?

1. Where is the latest mips24 offline tar-ball kept?

The latest version is: /stage/ssc-pipe/fmasci/MIPS24_pipeline/offline_pl/S13.0_delivery/ \ S13.0_DL_Offline_MIPS24_vsn32.tar.gz

See the <u>README_Solaris</u> file in this same directory for installation instructions.

Also, the document: "MIPS24_pipelines.pdf" included in the tar-ball has some useful information. Examples on how to execute the pipelines using test-data that is also included in the tar-ball are at the end of this document.

<u>NOTES</u>:

- No Linux version of the offline mips24 pipelines exist. It's not a simple matter of just replacing the Solaris binaries and libraries with Linux equivalents. Much re-coding is needed (for Fortran in particular).
- Please <u>*do not*</u> replace the binaries or libraries in the offline mips24 tar-ball with the latest from CVS. The binaries and some libraries were modified for specific offline use in the Perl scripts. The latest you have is S13.0 and I doubt any will change.

2. Where can I find a description and flowcharts of all mips24 pipelines?

See MIPS24_pipelines.pdf or MIPS24_pipelines.doc in the offline tar-ball (location given in FAQ#1).

3. Where can I find the latest bit-mask definitions for all types of masks involved in mips24 processing?

See MIPS24_pipelines.pdf or MIPS24_pipelines.doc in the offline tar-ball (location given in FAQ#1).

4. What are all the mips24 science-related pipeline products that we are archiving (but not necessarily distributing to users)?

Products from the Spitzer archive (as of Thu Feb 3 09:37:29 PST 2005):
e.g., for AOR: 12747776:

<u>*bcd (plscriptid 1030)*</u>:

dce23898672_dp30815132_a153126005_bunc.fits (BCD uncert. image)
dce23898672_dp30815132_a153126006_bbmsk.fits (BCD mask image)
dce23898672_dp30815132_a153126007_diff.fits (difference plane image)
dce23898672_dp30815132_a153126008_slope.fits (slope plane image)
dce23898672_dp30815132_a153126009_slunc.fits (slope uncert. image)
dce23898672_dp30815132_a153126010_dfunc.fits (difference uncer. image)
dce23898672_dp30815132_a153126011_slpmk.fits (mask for slope image)
dce23898672_dp30815132_a153126013_dfmsk.fits (mask for difference image)
dce23898672_dp30815132_a153126015_raw.fits (raw/unproc. image with pointing)
dce23898672_dp30815132_a153126017_ptg.log (log of pointing transfer)
dce23898672_dp30815132_a153126019_bcd.log (log of science DCE processing)
dce23898672_dp30815132_a153126021_qaDCE.qa (QA diagnostics on DCE)
dce23898672_dp30815132_a153126025_qa.qa (QA diagnostics on BCD)
dce23898672_dp30815132_a153876878_brmsk.fits (Outlier mask from post-BCD proc.)
dce23898672_dp30815132_bcd.fits (main BCD with saturated slope pixels replaced by diff. image)

bqd (fif: plscriptid 1040):
bqd483394_ep662499_a3532029_fiflog.tbl (log of Fiducial Image Frame gen.)
bqd483394_ep662499_fif.tbl (Fiducial Frame Table for mosaic)

(mopex: plscriptid 1048):
bqd483402_ep662530_a3532154_msaic.log (log of post-BCD processing)
bqd483402_ep662530_a3532155_msunc.fits (mosaic uncert. image)
bqd483402_ep662530_a3532156_mscov.fits (mosaic coverage image)
bqd483402_ep662530_a3532157_cdfid.fits (log of Control Data Files used)
bqd483402_ep662530_a3532158_mosnl.nl (mopex [mosaic+apex] namelist)
bqd483402_ep662530_a3532162_xtrct.tbl (source extraction table)
bqd483402_ep662530_a3532163_bmed.fits (median of BCD stack)
bqd483402_ep662530_a3532164_bmedu.fits (uncert. for median of BCD stack)
bqd483402_ep662530_msaic.fits (main mosaic image)

***Note:** the above product names are assigned by the product archiver, not mips24 pipelines. For the mips24 SUR-mode science pipeline with pointing and FPG run, the final (internal) products copied to the sandbox are:

bcd_fp.fits (main bcd = sat. slope pixels replaced by diff. pixels)
bcd_mask_fp.fits (corres. mask to above)
bcd_uncert_fp.fits (corres. uncertainty image to above)
bmask_diff_fp.fits (bmask for difference image)
bmask_slope_fp.fits (bmask for slope image)

diff_fp.fits (difference image)
diff_uncert_fp.fits (difference uncertainty image)
raw_tranhead_fp.fits (raw image with translated header)
slope_fp.fits (slope image)
slope_uncert_fp.fits (slope uncertainty image)

5. How do I measure the x, y spot position in a list of flats or DCEs for trending and analysis?

In the mips24 offline tar-ball (for location see FAQ#1), execute the script "SpotPosForFITSlist.pl" with no command-line arguments for a tutorial.

The general output product is a text/table file of spot-statistics with column fields:
[X_pos, Y_pos, status(ignore), CSM_PRED, FITS_filename]

For your information, an existing set of spot-statistics files for campaigns MC1-20 made from photometry-mode flats can be found under:
"/stage/ssc-testdata-mips/fmasci/RealFlatObs/Spot_Pos_summary".

6. What's the procedure to correct for pointing in <u>all MIPS</u> modes in a campaign due to scan-mirror wobble?

The pointing can be corrected in reprocessing of a campaign once you know the mean spot position offset for that campaign relative to a campaign with a-priori known good *absolute* pointing. A description of the algorithm is outlined in the document "PtgCorrFromSpots.pdf" which is included in the offline mips24 tar-ball (see FAQ#1 for location). The relative spot positions from a campaign with good absolute pointing are computed using the script "AbsPtgDiffFromSpots.pl" which is also in the mips24 offline tar-ball. You can type this with no command-line arguments to get a tutorial and an example. Here's a quick recipe on how to compute the mean relative spot offsets for a list of campaigns and how to use them in reprocessing of each campaign so that the pointing in BCD can be corrected.

1. First identify a campaign which you think has the best *absolute* WCS pointing. In my testing, I found that MIPS Campaign (*abbrev*. MC) 17 came pretty close.

2. For all the campaigns for which you desire a mean relative spot offset (relative to the absolute campaign in 1), you must have first generated a "spot-statistics" file for each using the script mentioned in FAQ#5. Call these "spot_stats.txt_MCi" where i=1, 2, 3 etc… Also ensure you have generated a spot-statistics file for the absolute

campaign in 1 above. Call this latter file "spot_stats.txt_MCabs". An existing set of spot-statistics files made for campaigns MC1-20 can be found under: "/stage/ssc-testdata-mips/fmasci/RealFlatObs/Spot_Pos_summary".

3. Once you have all the spot-statistics files, you can now execute the script "AbsPtgDiffFromSpots.pl" to compute the actual relative offsets using all the available spot versus mirror position information in these files. Here's an example run, which is executed in the same directory where all the spot-statistics files reside:

   AbsPtgDiffFromSpots.pl -i spot_stats.txt_17 -d . -p 2.598 -o Results.txt -u 0.1

   If you were clever enough to read the tutorial for this script, I would not need to describe what each of the command-line inputs mean, but I will anyway. First, the file proceeding the "-i" option is the *absolute* campaign mentioned in 1 above. The "." proceeding the "-d" option means use all the "spot_stats.txt_MCi" files in the PWD. The "-p" option specifies the mips24 native pixel scale, the "-o" option the output results file, and the "-u" option allows the user to specify the centroiding uncertainty for the spot positions, which is typically +/-0.1. The output "Results.txt" file in the above example will have *three* columns: [*Campaign number* (MCi); *relative spot position (in arcsec)*; *uncertainty in relative spot pos. (arcsec)*].

4. Before reprocessing of a specific campaign on ops, you now take the "*relative spot position (in arcsec)*" (let's call it ***p***) listed in the "Results.txt" file mentioned in 3 above for that campaign, and write that number in the "\real YspotPosDiff = ***p***" field in the mirrorparameters.tbl calibration file for all MIPS channels. And that's it!

7. Is there any simple way to find the closest campaign(s) to a specific campaign of interest according to spot position?

Yes, execute the script: "Find_closest.pl" from the offline mips24 tar-ball (location given in FAQ#1). Execute with no command-line arguments for a tutorial.

8. How do I make scan-mirror position dependent flats offline?

First, you query for and retrieve all post-tranheaded FITS files with pointing from ops, list them in a file (one per line) and execute the "MakeMirrorDepFlats.pl" script in the mips24 offline tar-ball (location given in FAQ#1). Here's an example:

Type on command-line with no arguments for a tutorial. The usual/standard command-line for making SMD flats is:

MakeMirrorDepFlats.pl -i ImageList.txt -o PROD -d 0.5 -e 0.5 -f 0.5 -g 0.5 -h 1 -j 1

In this run, the final (output) scan-mirror dependent flats will appear in the directory: "./PROD/FLAT_products". Directly under the "PROD" directory, you will find the input image filelists that went into making each scan-mirror dependent flat.

To use these flats online, you will need to load these into the SODB:mips1fallback table with a specific [sclkstart, sclkend] range. See FAQ#14 for instructions.

9. Why do there appear to be duplicate flatfield products (exactly the same) for mirror positions separated by ~0.5 DAC bins?

You'll see this only after executing MakeMirrorDepFlats.pl, *not* in the spot-dependent flatfield libraries. This is because the full DAC range is subdivided into *N* 0.5 DAC bins (or whatever was specified on the command-line) and a DCE's mirror position can fall into two adjacent DAC bins. The scan-mirror dependent flats are made and named from DCE lists compiled from members in each DAC bin. So if two bins have the same DCEs, you'll get the same flat for each mirror DAC bin.

10. How do I make synthetic (or interpolated) flats from a set of real scan-mirror dependent flats?

They are made using the script: "MakeSyntheticFlatsFromRealFlats.pl" from the offline mips24 tar-ball (location given in FAQ#1). After sourcing the regular_env.csh in the tar-ball, type this script on the command-line with no arguments for a tutorial and examples.

11. Why do some synthetic (or interpolated) flats contain no spot positions in their filename or header?

Because these are cases where a spot could not be measured - i.e., it fell too close to an edge or was too faint (it was contaminated by signal). These will never be picked-up in processing. They are included in the library for completeness so that the entire interpolation grid probed is represented.

12. What are all the different ways we can flatfield DCEs on ops?

1. Use the spot-dependent, interpolated (synthetic) flat library directly with _no_ gradient corrections to the campaign being processed. These have characteristic caltype = "flatfieldspot". To use these, set "pick_spot_match_flat = 1" and "SpotMatch_with_Gradient_Corr = 0" in the caltrans.nl cdf.

   If a spot matching flat is not found, it will default to a regular (dedicated) mirror dependent flat for the given CSM_PRED, CSM_RATE made from some earlier campaign (now it's MC17). If in turn a mirror-dependent flat cannot be found, it will default to some specific default superflat for the corresponding CSM_RATE with generic name for example "mips24_flatfield_default_<rate>.fits", where <rate> = phot, slow, medi, fast.

2. Use a specific set of dedicated mirror-dependent flats that matches the campaign being processed. These have characteristic caltype = "flatfieldmirr". To use these,
   - set "pick_spot_match_flat = 0" and "SpotMatch_with_Gradient_Corr = 0" in the caltrans.nl cdf.
   - this option assumes you have made the mirror-dependent flats offline using FAQ#8 and have loaded them into the database with a sclkstart, sclkend range corresponding to the campaign of interest as described in FAQ#14

3. Use the "spotmap" library together with a specific campaign superflat to correct for gradients in the DCEs. The spot-map and superflat are multiplied in processing to create an effective internal flat. The spotmaps have characteristic caltype = "spotmap" and the superflats have characteristic caltype = "flatfield".

   There are two sub-options here:

   (i). Let the pipeline automatically pick the campaign superflat that is closest in time prior to the DCE observation-time. This assumes that there are valid entries in the SODB:mips1metadata table whose quality has been blessed by the IST. This is most useful in initial PAO processing where superflats are made and registered early in the campaign before the main science-DCE processing. To use this option,
   - ensure "pick_spot_match_flat = 1" and "SpotMatch_with_Gradient_Corr = 1" in the caltrans.nl cdf.
   - under the "CALTRANS_MIPS1SUR_405" blockname in caltrans.nl, ensure that: CalRule = '0;0;0;0;0;0;100',

(ii). Make pipelines pick a specific campaign superflat that was pre-loaded into the SODB:mips1fallback table. This is most useful in reprocessing(s) where the IST wants to use a specific, good set of superflats made offline beforehand. To use this option,

- ensure "pick_spot_match_flat = 1" and "SpotMatch_with_Gradient_Corr = 1" in the caltrans.nl cdf.
- under the "CALTRANS_MIPS1SUR_405" blockname in caltrans.nl, ensure: CalRule = '0;0;0;0;0;0;0',
- Ensure that the correct superflat from mips1fallback is picked (caltype="flatfield") by setting the sclkstart, sclkend for these records to the time range for the campaign being reprocessed.

Like in 1. above, if a spot-map is not found, it will default to a regular (dedicated) mirror-dependent flat for the given CSM_PRED, CSM_RATE made from some earlier campaign (now it's MC17). If in turn a mirror-dependent flat cannot be found, it will default to some specific default superflat for the corresponding CSM_RATE with generic name for example "mips24_flatfield_default_<rate>.fits", where <rate> = phot, slow, medi, fast.

13. What happens if I (pipeline operator) am undergoing PAO and reprocessing at the same time for MIPS-1. Can I use two different flat-picking mechanisms from above?

Yes indeed. Here are the options that ISTs might request, and the database updates that need to be done by the PO (with instructions in FAQ#14):

Preamble: you will never have to change the [sclkstart, sclkend] interval for the "flatfieldspot" and "spotmap" caltypes in mips1fallback. There will always be one set of these and they will be applicable to all DCEs acquired at all times. The only caltypes where the [sclkstart, sclkend] will need to be toggled is "flatfield" (representing superflats) and "flatfieldmirr" (representing dedicated mirror-dependent flats).

a) If you are PAO processing and re-processing at the same time and have been told to use option "3(i)" in PAO processing and option "2" in reprocessing, simply ensure the sclkstart, sclkend for the caltype="flatfieldmirr" [option 2] is set to that of the campaign being reprocessed.

b) If you are PAO processing and re-processing at the same time and have been told to use option "3(i)" in PAO processing and option "3(ii)" in reprocessing: first,

since there is only one "caltrans.nl", you will need to place the correct caltrans.nl containing either CalRule = '0;0;0;0;0;0;100' [option 3(i)] or CalRule = '0;0;0;0;0;0;0' [option 3(ii)] into different respective PLID subdirectories on the drones. Second, for option "3(ii)", ensure that the sclkstart, sclkend for specific caltype = "flatfield" (superflats) is set to the time range for the campaign being reprocessed.

c) If you are PAO processing and re-processing at the same time and have been told to use option "1" in PAO processing and option "3(ii)" in reprocessing, simply ensure that the sclkstart, sclkend for specific caltype = "flatfield" (superflats) is set to the time range for the campaign being reprocessed.

I can't think of any other combinations that the ISTs may request. My guess is that they'll want to use option (b), since it is very robust. Anything else is asking for trouble. Before using either a, b or c, please ensure you read FAQ#12 above to ensure you haven't missed anything in the deployment (e.g., cdf updates etc..).

If you are not undergoing simultaneous PAO and campaign reprocessing, then you can simply pick the sub-option that applies from either a, b or c.

14. How do I load new calibration products into the SODB:mips1fallback table?

Follow these steps (the first six steps are mainly set-up and only need to be done once).

1. All the latest mips24 cal-files are located under:
   /stage/ssc-testdata-mips/fmasci/Mips24FBCal/

   Regular *non-flat* related cal-products are all the “mips24_*” files in this directory.

   The following subdirectories contain flat-related cal-products:
   ./allsynthflats/ — set of synthetic flats [caltype = "flatfieldspot"]
   ./mirror_dep_flats/ — default set of mirorr-dependent flats [caltype = "flatfieldmirr"]
   ./spotmaps/ — set of spot (response) maps [caltype = "spotmap"]
   ./superflats/ — specific sets of campaign superflats for optional mips1fallback loading (used for gradient correction) [caltype = "flatfield"]

2. The primary fallback loading script is under the ./FallBackDBLoading/ subdirectory named "loadfallback.csh". All ancillary files required by this script are also under this directory. Because there are thousands of flat-related cal products, the loadfallback.csh script can be automatically updated to include these by simply

executing the "GenFlatEntries_forFBloadscript.pl" script. The regular (non-flat related) cal-products are very static for mips24 and their entries in loadfallback.csh don't need to be updated as often.

3. So to start, I suggest you copy the /stage/ssc-testdata-mips/fmasci/Mips24FBCal/ directory intact, *EXCEPT* for the four flat-related sub-directories mentioned above (since they're huge) to your private area.

4. Create the four (empty) flat-related subdirectories mentioned above under the same level where the ./FallBackDBLoading/ subdirectory is located. You're now free to populate these four sub-directories with specific files that the IST gives you.

5. Take a look at the header of the loadfallback.csh script and ensure that first, the environment variables under the section "# Execution variables:" are correct for your setup. Second, ensure that the environment variables: OUTPUT_LOCATION, OUTPUT_LOC_MIRR_FLATS, OUTPUT_LOC_SPOT_FLATS, OUT_LOC_SPOT_MAPS and OUT_LOC_SUPERFLATS are set to directories on the ops/processing network that are visible by all the drones. This is usually where the PO will archives the cal-files. Note that the "OUTPUT_LOCATION" environment variable points to the location of "regular" non-flat related cal. files. All other variables point to subdirectories under the directory specified by "OUTPUT_LOCATION". I advise that you stick to this strategy to avoid confusion.

6. Unless the IST says otherwise, keep the "STATUS" environment variable set to 2048.

7. If you have new flat-related cal-products in either of the above mentioned sub-directories, execute the "GenFlatEntries_forFBloadscript.pl" script to update "loadfallback.csh" with flat entries as follows (all on one line):

   ```
   GenFlatEntries_forFBloadscript.pl -m ../mirror_dep_flats -s ../allsynthflats
         -p  ../spotmaps -u ../superflats -t loadfallback.csh
   ```

8. If you want to load records into mips1fallback with a specific time-range of applicability, update the SCLK_Start, SCLK_End fields in the correct ensemble_info_FB_*.txt file (under the ./FallBackDBLoading/ directory) for the desired cal product(s). Note that the actual "caltype" that's loaded into mips1fallback is also in these ensemble_info_FB_*.txt files.

9. You're now ready to execute "loadfallback.csh" under ./FallBackDBLoading/ and see colorful fireworks.

15. Do mips24 calibration products have to have some special header keywords/format to be used in D/L pipelines?

Yes! See the "Mips24CalFitsFormat.pdf" document in the offline mips24 tar-ball (location given in FAQ#1).

16. Where are all the latest namelists kept for mips24?

They are in /stage/ssc-testdata-mips/fmasci/Mips24cdf/. Note that the IST also has all the copies that have ever been deployed on ops sorted by date.

17. How do I query for "super-flats" made from dedicated scan-mirror dependent flats in online operations?

Use dbaccess on a sops machine with the SQL query:

```
set role table_read;
select b.title, a.filename
        from ensembleproducts a, requests b
        where a.reqkey = b.reqkey
        and a.vbest > 0
        and (b.title like '%smd%')
        and (a.filename like '%flat%');
```

18. Are the flatfields that are made from the spot-map and super-flat on a DCE basis being released to the user? Are they being archived so that the user can access them later?

Strictly no way José. This is a complicated matter. I put forward the suggestion of archiving this as a product along with the BCD, but the IST doesn't want to. So, this is purely an internal product and is *not* copied to sandbox. It can be reconstructed by the user from the spot-map and super-flat by simply multiplying the two together. The user *will* have access to the spot-maps and the super-flat to perform this if needed.

19. Can I get a summary of how many BCDs were actually "spot-flattened", "mirror"-flattended, or were gradient-corrected on ops?

Yes, on a sops machine, execute the "SpotFlatsStats.pl" script contained under: $SOS_VERSION/downlink/perltools/ after editing the inputs defined between the "======" delimeters in the script, e.g., update the variables:

```
#=====================================================================
# Input variables.

  $INFORMIXSERVER = 'sodb1';
  $TARGETDB = 'archive';
  $SODB_ROLE = 'table_read';
  $AORKEY = 5547776;
  $PlScriptId = 1030;
#=====================================================================
```

You should only need to update the "$AORKEY" variable above. When done, simply execute "SpotFlatsStats.pl" with no command-line arguments.

20. By simply looking at a science BCD header, how can I tell what calibration files were used to process that DCE.

You'll need someone who has basic database experience. Depending on whether cal-products were queried from the mips1fallback or mips1metadata SODB tables, the FBID* or EPID* keywords respectively will tell you. Here are the possible keywords you're most likely to see.

EPIDDARK - Dark ensemble product ID
EPIDFLAT - Flat ensemble product ID
EPIDSFLT - Super-flat ensemble product ID
EPIDSPOT - Spot-map ensemble product ID
EPIDNONL - Non-lin. ensemble product ID
EPIDROWF - Row-flux corr. ensemble product ID
FBIDDARK - Dark fallback product ID
FBIDFLAT - Flat fallback product ID
FBIDSFLT - Super-flat fallback product ID
FBIDSPOT - Spot-map fallback product ID
FBIDNONL - Non-linearity fallback product ID
FBIDROWF - Row-flux corr. fallback product ID

FBIDPMSK - Pixel-mask fallback product ID
FBIDDNT - DN-to-Flux single file fallback ID

21. Are any calibration filenames written to the BCD header?

Yes, only for flatfields since these are the most important types of calibration.

- If a spot-dependent or mirror-dependent flat was used (e.g., options 1. or 2. from FAQ#12 above) then you'll see the following keyword in BCD headers:
  FLATUSED - Flatfield filename used

- If a spot-dependent flatfield was used, then the value of "FLATUSED" will be of the form: mips24flat_<rate>_<csm_pred>_x<x_position>_y<y_position>.fits

- If a dedicated mirror-dependent flatfield was used, then the value of "FLATUSED" will be of the form: mips24_flatfield_<scan_rate>_<csm_pred>.fits

- If instead a specific campaign superflat with a spot-matching spot-map was used to correct for gradients (option 3 from FAQ#12 above), the you'll only see the following keyword in headers:
  SPOTMAP - Spot-map used as part of gradient correction.

22. What's the maximum number of images allowed for input into the "flatfield" module?

Cuurrently, it’s 2500 DCEs, therefore, ensure you don't schedule a flat-cal AOR with a number of DCEs exceeding this. Otherwise, it will blow up on operations and you'll get no flatfield product to use as a superflat for your gradient corrections.

23. Why do I still see spots in photometry-mode mosaics after flat-fielding using spot-dependent flats?

You will <u>always</u> see spots (or dark smudges) in the mosaic, especially for photometry-mode with many redundant mirror positions. There's an unavoidable fact of life called centroiding uncertainty, and this is associated with all measured spot positions. I'm going back to basics now. A typical spot in a DCE has a centroiding error of at least 0.1 pixel and that in the matching flat has about 0.05 pixel error. After a match is performed, the error in the radial separation between DCE-detected spot and flat-detected spot is at least ~0.12 pixel. This means you are always bound to get residual spots (no matter how tiny) in the BCDs after flat-fielding, just because of centroiding error. Lots of mirror positions at the

same telescope pointing in phot-mode will exacerbate the appearance of spots in a mosaic. Remember, a spot will be coadded and have it's SNR increased just like a star. These tiny BCD spot residuals should be less apparent in scan-mosaics since there is more variation in mirror positions over many more telescope pointings (remember the telescope is scanning too), so they're likely to be dithered-out. There's no way to make it better. We are currently interpolating flats to 0.1 pixel and there's no point to go smaller since the centroiding error is of at least the same size.

24. What are all the mips24 threads currently in operations?

The following is a dump of records from the plscripts table relevant for mips24 as of Wed Jun 1 14:31:28 PDT 2005:

plscriptid	plscriptnum	followonplscriptid	exposuretype	readoutmode
1000	401	1030	d1	SUR
1001	407		d1	SUR
1005	402	1030	d1	RAW
1006	408		d1	RAW
1010	403	1030	l1	RAW
1011	409		l1	RAW
1012	404	1030	s1a	SUR
1013	404	1030	s1b	SUR
1014	404	1030	st1	SUR
1015	404	1030	f1	SUR
1016	410		f1	SUR
1017	404	1030	ft	SUR
1018	404	1030	ft	
1019	406	1030	tfl	RAW
1020	405	1030	scn	SUR
1021	405	1030	pht	SUR
1022	405	1030	cal	SUR
1023	405	1030	p1	SUR
1024	405	1030	fp1	SUR
1025	406	1030	rsc	RAW
1026	406	1030	rph	RAW
1027	406	1030	scn	RAW
1028	406	1030	pht	RAW
1029	406	1030	tpm	RAW
1031	404	1021	pfl	SUR
1032	410		pfl	SUR

1033	404	1020	sfl	SUR
1034	410		sfl	SUR
1035	411		lat	SUR
1036	406	1030	ft	RAW
1037	405	1030	tpm	SUR
1040	130		reffr	
1041	131		sfpse	
1042	400		refin	
1047	137		bmerg	
1048	139		mopex	
5001	412	1030	zip	

25. In what order do all the different mips24 threads run on ops for a given starting "initiplscriptId"?

See the $SIRTF_OPS/thread_order file. The first column is the initiplscriptId:

1000 1030:1001e:1002e
1005 1030:1006e:1007e
1010 1030:1011e
1012 1030:1016e
1013 1030:1016e
1014 1030:1016e
1015 1030:1016e
1017 1030:1016e
1018 1030:1016e
1031 1032e:1021:1030:1040e:1048e
1033 1034e:1020:1030:1040e:1048e
1019 1030:1040e:1048e
1020 1035e:1030:1040e:1042e:1048e
1021 1035e:1030:1040e:1048e
1022 1035e:1030:1040e:1048e
1023 1035e:1030:1040e:1048e
1024 1035e:1030:1040e:1048e
1025 1030:1040e:1048e
1026 1030:1040e:1048e
1027 1030:1040e:1048e
1028 1030:1040e:1048e
1029 1030:1040e:1048e
1036 1030:1040e:1048e

1037 1035e:1030:1040e:1048e
5001 1030

26. How does saturation thresholding work in pipelines and what's the deal with pixel replacement in the SUR-science pipeline?

On-board the spacecraft, the difference image (a 0.5 MIPS-second exposure) is set to zero everywhere except for those pixels whose count rate is large enough to lead to an A-to-D (analog to digital) saturation in the data ramp over the total image integration. This is performed on-board using a nominal (and conservative) difference threshold value of 600 DN/sample-time (where 1 sample-time = 0.5 MIPS-second = 0.524288 real sec.) over a 30 MIPS second integration, or equivalently, ~50.3 MJy/sr. Later in pipeline processing, a larger difference image saturation threshold of 1000 DN/sample-time (for a 30 MIPS-second integration), or ~83.7 MJy/sr is used above which slope pixels are replaced with difference pixels. In the end, this yields more reliable fluxes for bright sources. More specifically, the pipeline re-scales this nominal threshold according to the image integration time since a shorter integration can tolerate a larger count-rate before the ramp can start to saturate and hence bias the slope value. All this is done in the "satmask" module. The following scaling is used:

Difference_Sat_Threshold = 1907.34 * (30 / EXPTIME) * 0.04391 [MJy/sr],

where the prefactor "1907.34" is [1000/0.524288] DN/real-sec, i.e., the nominal threshold for a 30 MIPS-second integration, EXPTIME is the image integration time in MIPS seconds, and the postfactor "0.04391" is the conversion factor from DN/real-sec to MJy/sr. Of course, some sources may saturate the ramp even in the first read, leading to what's called "hard saturation" and causing the difference values to be zero. The latter cannot be corrected and the sources will appear to have "holes" in the centers of the PSFs after processing. Finally, suspected saturated pixels are replaced by those from the first difference image to create reliable slope data where possible, resulting in a larger dynamic range.

27. Is there a command-line tool to flip and rotate a list of FITS images?

Yes, execute the script: “FlipRotateImages.pl” from the offline mips24 tar-ball (location given in FAQ#1). Execute with no command-line arguments for a tutorial.

28. Is there a tool which gives a summary of processing on a single pixel (i.e., history in temporal order) for all intermediate FITS files generated in a pipeline?

Yes, execute the script: “pixelhistory.pl” from the offline mips24 tar-ball (location given in FAQ#1). Execute with no command-line arguments for a tutorial.

29. How do I “self-flatten” BCDs (or self-calibrate) prior to mosaicking?

Simply execute the script: “MakeSelfCalFlatsAndFlatten.pl” from the offline mips24 tar-ball (location given in FAQ#1). Type script on command-line with no arguments for a tutorial and example.

30. I have a FITS image containing WCS information. Is there a script where I supply a list of RA, Decs and it will report the x, y locations and fluxes of all pixels?

Indeed there is, otherwise I would not have posed such an obtuse question. The script is called "RADec2xyFluxForRADecList.pl" in the offline mips24 tar-ball (location is given in FAQ#1). Type this with no command-line arguments for a tutorial.

31. Suppose I have a list of FITS images (with WCS) which may be overlapping. Is there a script where I supply a RA, Dec and it will report the x, y locations and fluxes of all pixels from all images in the stack?

Indeed there is, otherwise I would not have posed such an obtuse question. The script is called "RA2xyForImageList.pl" in the offline mips24 tar-ball (location is given in FAQ#1). Type this with no command-line arguments for a tutorial.

32. Is there any command-line tool where I just supply a FITS image and it returns all 2MASS sources contained therein?

Indeed there is. It's more powerful than you think. It is called "Query2MASSperBCD_withXYconv.pl" in the offline mips24 tar-ball (location is given in FAQ#1). Type this with no command-line arguments for a tutorial. More specifically, you can supply a list of FITS images and it will create an IPAC table for each input image containing a summary of the 2MASS source RA, Dec, errors, J, H, K, ****AND**** also the x, y positions of the 2MASS sources in the image pixel frame. Aren't you lucky!

33. Is there a command-line tool which reports a quick and dirty measure of the RMS noise in an image and other statistics?

Yes there is. It is a C-binary called "imagenoise" located in the "../bin/" directory of the offline mips24 tar-ball (location given in FAQ#1). Ensure you first source your regular_env.csh before using. Also, there's a tutorial when executed with no command-line arguments. Here's an example run:

```
imagenoise   -i raw_tranhead_fp_AOR11373056_001.fits

Program: "imagenoise", Version 1.0, Wed Jun 15 10:15:44 2005

IMAGE NAME: raw_tranhead_fp_AOR11373056_001.fits
IMAGE SIZE = 128 x 128 pixels
NTOTSAMPLES = 16384
NEGATIVE SAMPLE FRAC. = 0.000000e+00%
NEGATIVESAMPLES = 0
NGOODSAMPLES = 16384
NBADSAMPLES = 0 (NaNs)
MINVALUE = 0.000000
MAXVALUE = 769.000000
MEAN (of good samples) = 689.390808
MEDIAN (of good samples) = 701.000000
NLTMEDIAN (Num. samples < MEDIAN) = 8046
RMSMEDIAN (RMS of NLTMEDIAN samples from MEDIAN) = 70.636658
```

34. Is there a command-line tool which crops FITS images to whatever size you like and saves it to a new FITS image?

Indeed there is. It is a C-binary called "cutoutfits" located in the "../bin/" directory of the offline mips24 tar-ball (location is given in FAQ#1). Ensure you first source your regular_env.csh before using. Also, there's a tutorial when executed with no command-line arguments. Note: this module also has an option to replace NaNs in the output image with whatever you like (pretty cool!).