ISIS WORKSHOP

Processing for MGS/MOC Imaging

Given by:

Astrogeology Team, USGS

Tammy Becker

Eric Eliason

Kris Becker

Acknowledgements:

United States Geological Survey/Mars Global Surveyor Project/

Planetary Data System/Planetary Geology and Geophysics Program

ISIS OVERVIEW

INTEGRATED SOFTWARE FOR IMAGES AND SPECTROMETERS

ISIS provides a comprehensive, user-friendly, portable tool for image processing, analyzing, and displaying remotely sensed image data.

ISIS primarily handles 2-D image data (single-band cubes) and 3-D data (multi-spectral cubes) from imaging spectrometers

Supported Missions:

Mars Global Surveyor, Clementine, Galileo NIMS, Galileo SSI, Viking 1 & 2, Voyager 1 & 2, Mariner, Mars Pathfinder, DS1, Cassini

ISIS ONLINE DOCUMENTATION

- http://wwwflag.wr.usgs.gov/isis-bin/documentation.cgi

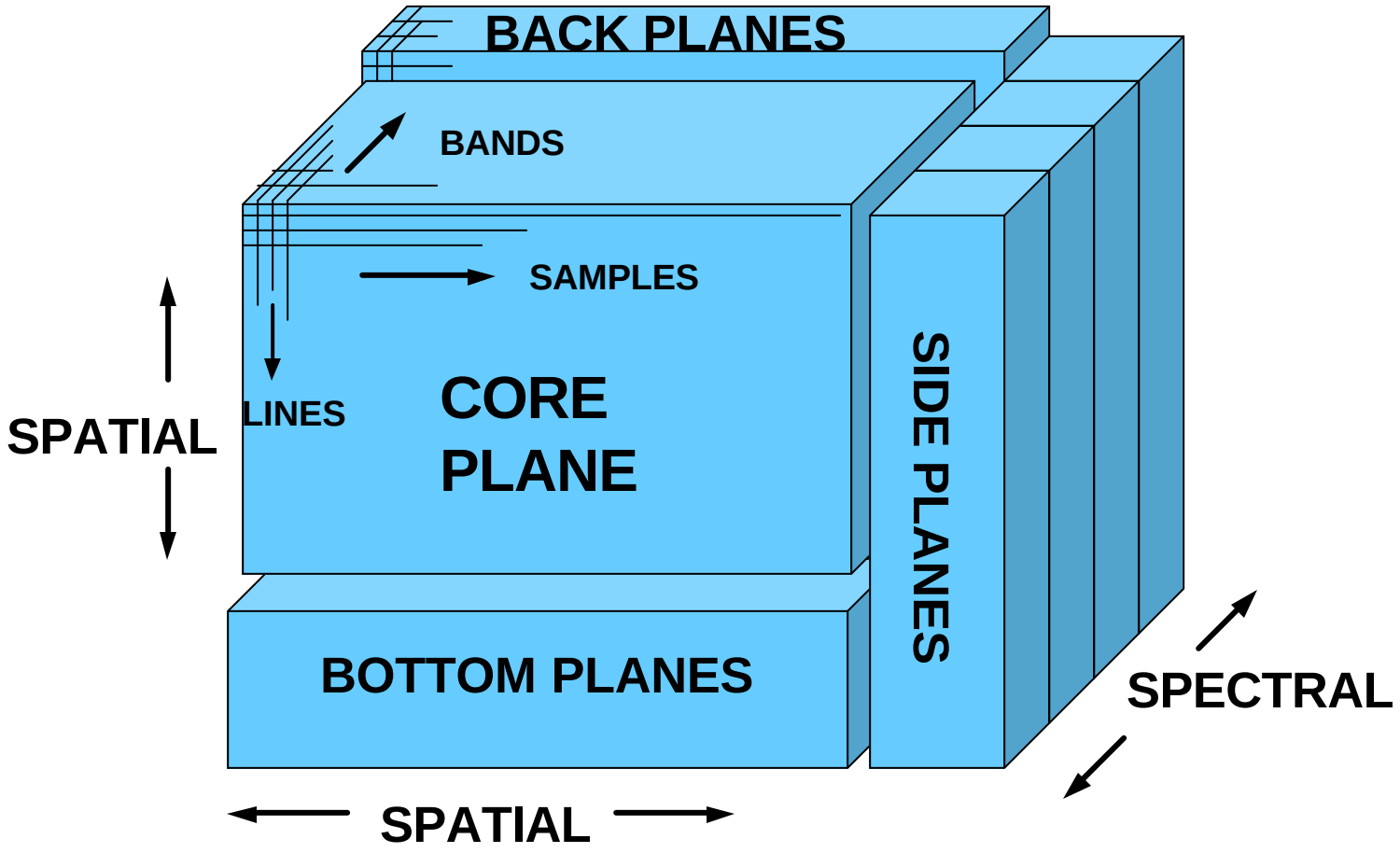
  Overview of ISIS Architecture (Appendix A-G)
  ISIS Program Lists
  ISIS Spectral Libraries
  Mission-Specific Image Processing (MGS/MOC)

- Categorized list of ISIS programs within the user-interface TAE.
  Command: *TAE>menu*

- tae>isisdoc from=*moclev0*.pdf (Creates a formatted *moclev0*.doc)

- isis-support@flagmail.wr.usgs.gov
  (e-mail address for problems and questions. Message is sent to a staff of administrators, programmers and advanced users that will assist).

LOGICAL VIEW OF *ISIS* CUBE

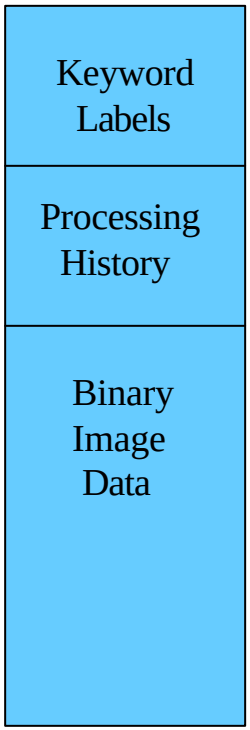


- **Suffix planes (back,bottom,side) are always stored as floating point**
- **Core plane can be 8bit, 16bit, or floating point**

ISIS LABELS

- ISIS cube labels are ASCII
  To view the keywords, type ‘more file.cub’

- Important keywords to be familiar with:



- CORE_ITEMS = (# of Samples, # of Lines, # of Bands)
- CORE_ITEM_BYTES = 1 (8 bit), 2 (16 bit), 3 (floating point)
- CORE_ITEM_TYPE = unsigned_integer, integer, real (core)
- CORE_BASE = 0.0
- CORE_MULTIPLIER = 1.0

- SUFFIX_ITEMS = (# of side, # of bottom, # of back)

- FILE_STATE = CLEAN or DIRTY

ISIS Pixel Data Formats

- ISIS Pixels are always considered to have REAL (Floating Point) values

For 32-bit pixels - a pixel value stored in the core directly contains the REAL pixel value as a 32-bit floating point number
(CORE_BASE=0.0, CORE_MULTIPLIER=1.0)

NOTE: CORE_BASE/MULTIPLIER & ORANGE parameter are ignored in 32-bit.

It is recommended to process in 32-bit (otype=3)

Convert to 8 or 16-bit with end-products only

ISIS Pixel Data Formats, *cont*

For 8 and 16-bit pixels - CORE_BASE and CORE_MULTIPLIER label keywords are used by the ISIS programs to convert the 8/16-bit stored values in the core to represent the REAL ISIS pixel values to be represented.

Real Pixel Value = (Float(stored value) * Mult) + Base)
Represented

Example: A radiometric calibrated image file in I/F units will generally have a pixel value range between 0.0 -> 1.0.

With the Type Conversion Parameters (CORE_ BASE and CORE_MULTIPLIER), the 0.0 -> 1.0 values will be represented by the ISIS programs as real, regardless of the stored bit type (8 or 16-bit).

Refer to ORANGE.PDF (tae>help orange) for explanation of how the ORANGE parameter is used in dealing with different bit types.

SPECIAL PIXEL VALUES

Indicates that a pixel does not contain a valid numerical value.

ISIS programs do not use Special Pixel Values in a computation.

Five types of Special Pixel Values:

NULL	No data available
HIGH_INSTR_SATURATION (HIS)	Instrument saturation (high)
LOW_INSTR_SATURATION (LIS)	Instrument saturation (low)
HIGH_REPR_SATURATION (HRS)	Process-introduced sat (high)
LOW_REPR_SATURATION (LRS)	Process-introduced sat (low)

8-bit images contain only NULL & HRS Special Pixel Values

SFROM PARAMETER (SUBCUBE SPECIFIER)

Format for Core Data

SFROM = “ss-es(sinc):sl-el(linc):sb-eb(binc)

Where ss	=	Starting Sample
es	=	Ending Sample
sinc	=	Sample Increment
sl	=	Starting Line
el	=	Ending Line
linc	=	Line Increment
sb	=	Starting Band
eb	=	Ending Band
binc	=	Band Increment

Format for Backplane Data:

SFROM = “::S( )”

Refer to SFROM.PDF (**tae>*help sfrom***)

"CLEAN" FILE CONCEPT

- "CLEAN" Files are assumed to be okay
- "DIRTY" Files indicate possible data corruption
- Occurs when:
  - Files are opened for write access and process interrupted, the file is not closed properly.
  - The processing history section of the cube labels becomes full and cannot be updated.
  - Two users have one file opened for write access at the same time.

- Recovering from a "DIRTY" ISIS file
  - "cleanlab" ISIS application will set FILE_STATE=CLEAN
  - To initialize the history section of the labels
    (*cleanlab from=file.cub clrhst=yes*)
  - Rerun application program and recreate the file

- To ignore the value of FILE_STATE
  - setenv ISIS_IGNORE_INTEGRITY TRUE

ISIS User-Interface - *TAE*

To invoke TAE: ***tae***<cr>

TAE>***help*** *moclev0* - General help and documentation of specific programs
TAE>***t*** *moclev0* - Tutor a program for parameters and execution
TAE>***menu*** - Categorized menu for available ISIS programs
TAE>***ush*** *cp* - Shell command to unix commands
TAE>***exit*** - Exit TAE

In tutor mode within a program:

?***help **** - Main general documentation of program
?***help*** *parameter* - Documentation on the specific parameter within the program (exit to return to program parameter page)
?***from=****image.cub* - Load parameters with values
?***run*** - Execute the program
?***save*** - Save parameter values in *program*.par file
?***restore*** - Restore parameters saved in *program*.par file
?***restore last*** - Restore previous parameter values executed (*last.par*)
?***exit*** - Exit program

Acronym Definition

PDS = Planetary Data System Imaging Node

(NASA's curator for the archives of digital images that are acquired and distributed by NASA flight projects)

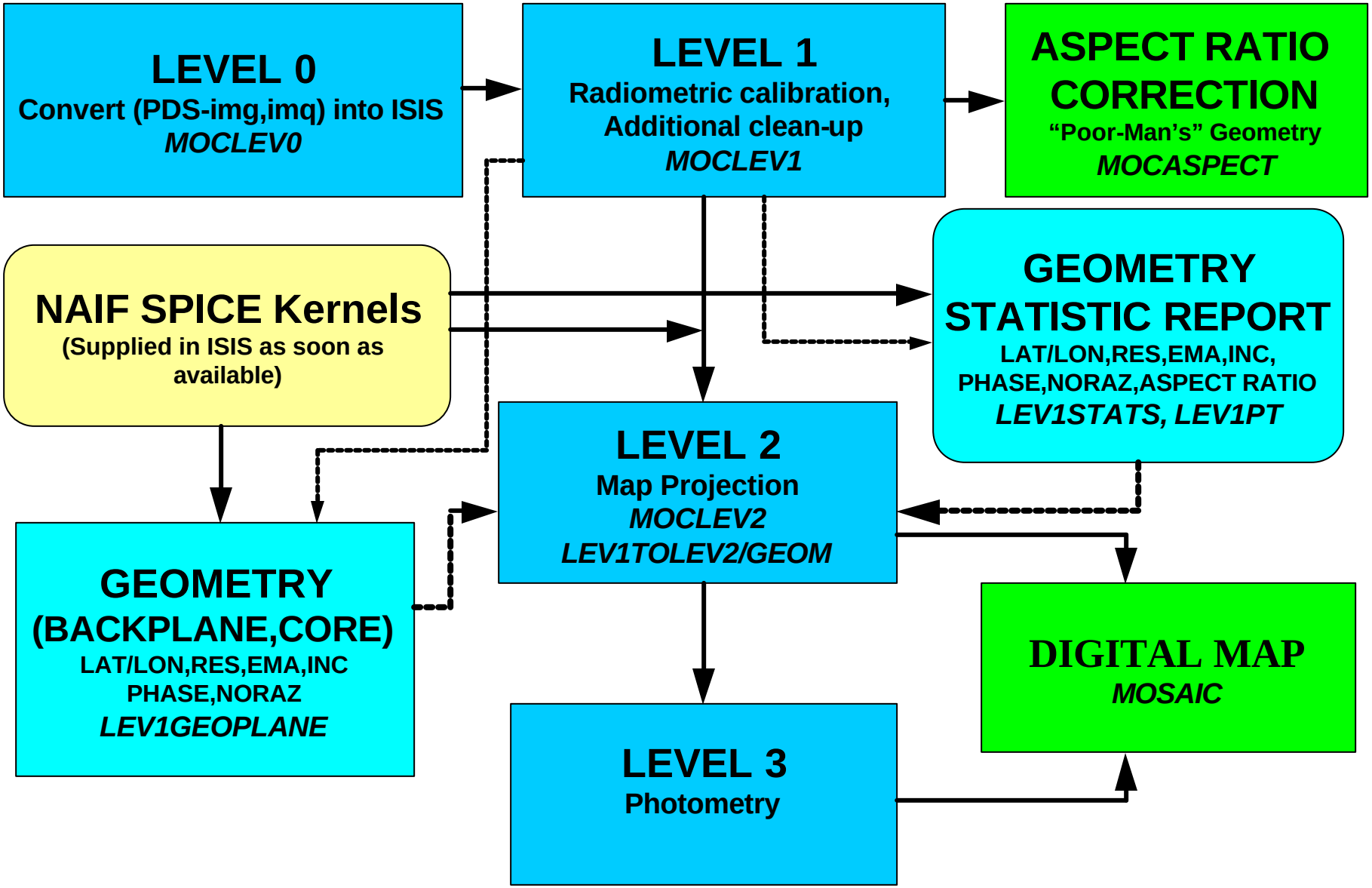
NAIF = Navigation and Ancillary Information Facility

SPICE = Spacecraft and Planet ephemerides, Instrument, C-Matrix (camera pointing), and Event kernel.

MOC = Mars Orbiter Camera

MOLA = Mars Orbiter Laser Altimeter

LEVELS OF ISIS MGS/MOC PROCESSING



MGS/MOC LEVEL SEQUENCE PROCESSING STEPS

- Public Release (PDS) data
  MOCSEARCH, MOCFTP

- Convert PDS .imq, .img images to ISIS
  MOCLEV0

  qview - interactive 'tvdoctor' function to remove erroneous data

- Radiometric calibration
  MOCLEV1

- Project images to map projection
  MOCLEV2

MOCLEVALL
perform Levels 0,1,2

- Retrieve and Process MOLA data
  MOLASEARCH
  MOCFTP
  MOLA2ISIS

MOCMOLA

MGS/MOC *ISIS* SUPPORT

Radiometry:

• **NA** - All cases are complete with continued refinements on W0 (absolute calibration) through use of in-flight calibration (Herkenhoff, et. al.)

•**WA/Red** - All cases complete with first order coefficients, improvements continue
- Non-variable and Variable Summing supported
- WAGO (WA Gain Offset) tables supported

• **WA/Blue** - Lacking radiometric coefficients

Geometry:

- Updated boresight pointing that ensures registration of MOC/NA/WA and MOLA data for simultaneous observations and timing offset corrections

• **NA** - Planned tests for optical distortion, most likely negligible

• **WA/Red** - Variable/Non-Variable Summing supported*****

•**WA/Blue** - Variable/Non-variable Summing supported*****

*****Variable Summing Mode software release is eminent

Photometry:

- Software not available for MOC, more analysis required for coefficients**.**

MGS/MOC Imaging System

• Mars Obiter Camera (MOC): a three-component imaging system one narrow angle camera (NA) and two wide-angle (WA) cameras designed to take high spatial resolution pictures of the surface and to obtain lower-resolution synoptic coverage of the surface and atmosphere. The cameras employ push broom technology, acquiring one image line at a time as the spacecraft passes over the Martian surface.

• Narrow Angle Camera: 2048 detector array capable of acquiring imaging at 1.4 meters/pixel resolution in ground swaths of 2.8 kilometers crosstrack and up to 25.2 kilometers downtrack. Onboard pixel summing and edit of images allows for greater downtrack areal coverage at reduced pixel resolution.

•Wide Angle Cameras: 3072 detector array capable of viewing Mars from horizon to horizon at lower resolutions (230 meters/pixel) using narrow-band "red" (557-625 nm) and "blue" (400-450 nm) filters. As with NA camera onboard pixel summing and edit of images allows for reduced resolution images.

Acquiring MGS/MOC Image Data

• MGS/MOC data deliveries to science community occur at six month intervals in April and October. Next scheduled release in April, 2001. Allow approximately 6 weeks for the PDS to validate, master, replicate released volumes for distribution to the science community.

• As of October/2000 57,857 MOC images have been released on 108 CD-ROM volumes, data acquisition time coverage from September/1997 to March/2000.

• Volumes MGSC_0001 - MGSC_0010 - Decompressed Standard Data Products - Pre-systematic mapping imaging: aerobraking and science phasing loops in decompressed format.

•Volumes MGSC_1001 to MGSC_1098 - Standard Data Products - Systematic mapping phase imaging in compressed format.

• MGS/MOC CD-ROM volume set available at no cost to NASA-funded investigators. Contact PDS Imaging Node (*pdsmgr@usgs.gov*).

• Others can obtain the MGS/MOC CD-ROM volume set at nominal cost from the National Space Science Data Center (http://nssdc.gsfc.nasa.gov/planetary/planetary_home.html).

MGS/MOC WWW and FTP Internet Resources

• PDS MGS/MOC Data Access Web Site (URL: http://ida.wr.usgs.gov/graphical.htm). This system provides a simple graphical point-and-click interface to the MGS/MOC data collection. This web site organizes access to MOC images by geographic location.

•PDS Planetary Image Atlas (URL: http://www-pdsimage.jpl.nasa.gov/PDS/public/Atlas/Atlas.html). General purpose search and retrieval system for MGS/MOC data products and other planetary image data sets. Data can be searched by a wide variety of search criteria including: location, viewing conditions, camera operating modes, resolution, etc.

• Malin Space Science Systems, responsible for MGS/MOC operations, web site (URL: http://barsoom.msss.com/newhome.html)

•PDS MGS/MOC FTP Access (URL: http://ida.wr.usgs.gov/d_access.htm) data available for "bulk download" of large volumes of data through anonymous ftp access at ida.wr.usgs.gov.

MOC File Naming Conventions

- **Systematic Mapping Phase - *xxxyyyyy*.imq**

  where:

  xxx = instrument planning period
  - cal = calibration phases
  - fha = fixed high-gain antenna phase
  - m00 = Instrument planning period, one month duration (m05 and m06 not used) two digits now represent month of MGS mapping.

  yyyyy = sequence number of image acquired within a planning period.

  imq = designates image file in compressed format

- **Pre-Systematic Mapping Phase - *xxxyyyzz*.img**

  where:

  xxx = mission phase
  - ab1 = aerobraking phase 1
  - sp1 = science phasing orbits 1
  - sp2 = science phasing orbits 2

  yyy = orbit number used in pre-systematic mapping phase

  zz = sequence number of image within an orbit.

  img = designates image file in uncompressed format.

MGS/MOC CD-ROM Volume Contents

<root>
• **aareadme.txt** - introductory descriptive information on volume.
• **errata.txt** - anomalies and problems with MGS/MCO data set.

<index>
• **imgindex.tab** - ASCII "Flat file" containing ancillary data for all images on the volume.
• **imgindex.lbl** - PDS label file describing contents and format of the imgindex.tab file.
•**cumindex.tab, .lbl** - like the imgindex.tab file but contains a cumulative index of all images on current and previous volumes. (Note: cumulative index of latest volume converted to Excel spread sheet and placed in $ISISMGSDATA directory and named: "mgsc_1098_cumindex.xls")

<document>
•**mocsis.pdf, .txt** - MOC camera Software Interface Specification.

<catalog>
•Contains information about the mission:
- •**mission.cat** - information of MGS mission.
- •**instrument.cat** - information of MOC camera.
- •**insthost.cat** - information of MGS spacecraft.
- •**dataset.cat** - information of data format and organization.
- •**ref.cat** - references for MGS/MOC camera.

<software>
•**readmoc** - decompression software and source code. Available for PC/Windows and Sun/solaris platforms.

<data directories named by first 6 characters of product id>
• Data directories for image data products.

Simple MGS/MOC search/retrieval tools in ISIS

Mocsearch - find images of interest based on geographic location, resolution, and "LsubS". The procedure will search the cumindex.tab file (located in $ISISMGSDATA directory). The output of **mocsearch** is a list file, defined by the **"to"** parameter, containing a list of images that match the search criteria. Users can view the output list to locate images on their MGS/MOC CD-ROM volume set or the list can be passed to **mocftp** to download the images from the PDS MGS/MOC ftp site.

TAE>**mocseach**
to=*list.lst*
latrng=(*min lat, max lat*)
lonrng=(*east long, west long*)
res=(*min res, max res*) - default (0.0, 0.6 kilometers/pixel)
lsubs=(*min lsubs, max lsubs*) - default (0.0, 360), Seasonal definition of northern hemisphere (0=first day of spring, 90=summer, 180=fall, 270=winter)
imgindex="*$ISISMGSDATA/cumindex.tab*" - optional input

Mocftp - Automatically download MOC images or MOLA/PEDR files from the ida.wr.usgs.gov ftp site. Optionally a URL can be specified for an alternate internet resources.

TAE>**mocftp**
fromlist=*list.lis* - list file can be created by mocsearch program.
url="*optional URL location of MOC/MGS image data*"

MOLA PEDR data files can also be downloaded from the PDS Geosciences node. The URL for the PDS Geosciences node is url="http://wundow.wustl.edu/geodata/mgscd"

Moclev0 - Ingestion of MOC Images into ISIS

The **moclev0** procedure reads PDS-formatted MOC images into the ISIS system and sets up pointers in the labels for access to SPICE and cartographic definition files.

How to run:

TAE>**moclev0**

from=*image.imq* *or* ***image.img*** - process a single image

-or-

fromlist=*raw.lis* - process all images listed in the ***"raw.lis"*** file.

delinput=no or yes - default no, a user may optionally delete input files as output files are created.

transtab="*translation table*" - optional, a user may optionally enter a PDS to ISIS translation table.

targdef="*target definition file*" - optional, a user may optionally enter a target definition file (see documentation in **levinit** program for more information).

kernlst="*MOC SPICE kernel pointer file*" - optional, a user may optionally enter a pointer file that provides information about location of SPICE kernels. It is possible for a user to construct a pointer file in order to use a different set of SPICE kernels.

instpars="*Instrument parameter definition file*" - optional, a user may optionally enter an instrument parameter file to change boresight pointing, camera focal lengths, etc.

lonsys=360 or 180 - default 360. A user may optionally specify the longitude system to be used in cartographic mapping. 360 = 0 to 360 longitude system, 180= -180 to 180 longitude system.

latsys=*ographic* or *ocentric* - default ographic. A user may optionally specify a planetographic or planetocentric coordinate system.

Moclev0 - Ingestion of MOC Images into ISIS

Considerations:

• Input for the **moclev0** procedure is PDS-formatted MOC Standard Data Products (.imq extension) or MOC Decompressed Standard Data Products (.img extension)

• Output images are named the same as input with the **".imq"** or **".img"** extension replaced with extension **".lev0.cub"**.

• Check the **moclev0.err** and **moclev0.prt** files for any errors encountered in processing.

• **Warning**: novice ISIS users should never attempt to use the parameters: **transtab, targdef, kernlst, instpars**

• When processing image sets that are geographically located near or on map longitude system boundaries, you should set the **lonsys** boundary to the alternate longitude system.

•The **moclev0** procedure tests to see if an image crosses a map longitude system boundary. If the boundary is crossed then the procedure automatically changes to the alternate longitude boundary system.

Details:

moclev0 procedure runs the following programs:

• **readmoc** - decompress a MOC standard data product image (image with **.imq** extension only).

• **pds2isis** - convert PDS decompressed image to ISIS.

• **moc_fixlabel** - Modify ISIS labels for MOC specific processing.

• **levinit** - Update ISIS labels to setup pointers to SPICE and definition files for cartographic processing.

• **lev1stats** - Determine if MOC images cross the longitude system boundary and if necessary modify longitude system indicated in labels.

Example ISIS label of MOC image

```
CCSD3ZF0000100000001NJPL3IF0PDS200000001 = SFDU_LABEL

/* File Structure */

RECORD_TYPE = FIXED_LENGTH
RECORD_BYTES = 512
FILE_RECORDS = 20432
LABEL_RECORDS = 30
FILE_STATE = CLEAN

^HISTORY = 31
OBJECT = HISTORY
END_OBJECT = HISTORY

^QUBE = 81
OBJECT = QUBE
/* Qube object description */
/* Qube structure */
  AXES = 3
  AXIS_NAME = (SAMPLE,LINE,BAND)

/* Core description */
  CORE_ITEMS = (512,20352,1)
  CORE_ITEM_BYTES = 1
  CORE_ITEM_TYPE = PC_UNSIGNED_INTEGER
  CORE_BASE = 0.0
  CORE_MULTIPLIER = 1.0
/* "true_value" = base + (multiplier * stored_value) */
  CORE_VALID_MINIMUM = 1
  CORE_NULL = 0
  CORE_LOW_REPR_SATURATION = 0
  CORE_LOW_INSTR_SATURATION = 0
  CORE_HIGH_INSTR_SATURATION = 255
  CORE_HIGH_REPR_SATURATION = 255

/* Suffix description */
  SUFFIX_BYTES = 4
  SUFFIX_ITEMS = (0,0,0)
  CORE_NAME = RAW_DATA_NUMBER
  CORE_UNIT = NONE
  DATA_SET_ID = "MGS-M-MOC-NA/WA-2-DSDP-L0-V1.0"
  PRODUCT_ID = "M08/08066"
  PRODUCER_ID = "MGS_MOC_TEAM"
  PRODUCT_CREATION_TIME = "2000-09-23T02:28:04.000Z"

  SOFTWARE_NAME = "makepds 1.7"
  START_TIME = "1999-10-31T19:59:09.240Z"
  UPLOAD_ID = "UNK"

DATA_QUALITY_DESC = "ERRORS"
  IMAGE_NUMBER = "9103108066"
  IMAGE_KEY_ID = "6258608066"
  RAW_MAXIMUM = 254
  RAW_MINIMUM = 1
  RAW_MEDIAN = 157
  RAW_MEAN = 157.127991
  RAW_STANDARD_DEVIATION = 9.203691

  GROUP = ISIS_INSTRUMENT
    SPACECRAFT_NAME = MARS_GLOBAL_SURVEYOR
    FIRST_LINE_SAMPLE = 1025
    CROSSTRACK_SUMMING = 2
    DOWNTRACK_SUMMING = 2
    FOCAL_PLANE_TEMPERATURE = 269.8
    GAIN_MODE_ID = "4A"
    INSTRUMENT_ID = "MOC_NA_A"
    LINE_EXPOSURE_DURATION = 0.482100
    MISSION_PHASE_NAME = "MAPPING"
    OFFSET_MODE_ID = "36"
    ORIGINAL_SPACECRAFT_CLOCK_COUNT = "625867174:106"
    RATIONALE_DESC =
      "Long traverse across 'smooth' surface in west Sinus Meridiani"
    ORBIT_NUMBER = 2898
    LINES = 20352
    SAMPLES = 512
    INSTPARS = "/usgs/isisd/mgsdata/moc_parameters.def.1"
  END_GROUP = ISIS_INSTRUMENT

  GROUP = ISIS_TARGET
    TARGET_NAME = MARS
    LONGITUDE_SYSTEM = 360
    LATITUDE_SYSTEM = "OGRAPHIC"
    TARGDEF = "/usgs/isisd/data/targets/mars.def.2"
  END_GROUP = ISIS_TARGET

  GROUP = BAND_BIN
    BAND_BIN_FILTER_NAME = "BROAD_BAND"
    BAND_BIN_ORIGINAL_BAND = 1
    BAND_BIN_UNIT = MICROMETER
    BAND_BIN_CENTER = 0.70
    BAND_BIN_WIDTH = 0.40
  END_GROUP = BAND_BIN
```

Example ISIS label of MOC image

```
GROUP = ISIS_INGESTION
    PDS2ISIS_VERSION = "2000-12-28"
    PDS2ISIS_TRANSLATION_TABLE =
      "/usgs/isisd/mgsdata/moc2isis_translation.def.1"
  END_GROUP = ISIS_INGESTION

  GROUP = ISIS_GEOMETRY
    BASE_KERNELS = ("/usgs/isisd/mgsdata/naif0007.tls",
      "/usgs/isisd/mgsdata/MGS_SCLKSCET.00032.tsc")
    SPACECRAFT_KERNELS = ("/usgs/isisd/data/de405.bsp",
      "/usgs/isisd/mgsdata/mgs_map3.bsp")
    CAMERA_KERNELS = "/usgs/isisd/mgsdata/mgs_sc_map3.bc"
    KERNLST = "/usgs/isisd/mgsdata/moc_kernels.def.1"
    NAIF_SOFTWARE_VERSION = "CSPICE_N0050"
    LEV_SOFTWARE_VERSION = "MGS_MOC_1.0"
  END_GROUP = ISIS_GEOMETRY
END_OBJECT = QUBE
END
```

MOC STATISTIC TOOLS

LEV1STATS - Compute the minimum, maximum, average and standard deviation values for the following geometric parameters: Latitude, Longitude, Sample Resolution (km), Line Resolution (km), Aspect Ratio (line/sample resolution), Phase Angle, Incidence Angle, Emission Angle, North Azimuth Angle. The output values are reported to the standard log file, print.prt, or an optional ascii file in column form.

How to run:

TAE>**lev1stats**

from=*image.lev1.cub*	- Process a single image (Must be level 0 or 1)
-or-	
fromlist=*lev1.lis*	- Process a list of input images
to=*geo.txt*	- Default=null, no ascii output file generated
toerr=*null (default)*	- Optional output file with a list of images that were unsuccessful.
inc=*(1,1) (default)*	- Sample/Line increment for calculations. For efficiency, the user can successfully specify an inc=(10,10).

MOCAVGSTATS - Sort *lev1stats* output (print.prt or ascii) based on latitude, longitude and/or resolution constraints. The output ascii file contains the image filenames and their average statistics that fall within the constraints.

LEV1PT - This program generates geometric information for a single sample/line or lat/lon position in a cube or list of cube files. The information listed includes the following geometric parameters: Sample, Line, Latitude, Longitude, Sample Resolution, (km), Line Resolution (km), Aspect Ratio, Phase Angle, Incidence Angle, Emission Angle, North Azimuth Angle

How to run:

TAE>**lev1pt**

from=*image.lev1.cub* - Process a single image (Must be a Level 0 or 1)

-or-

fromlist=*lev1.lis* - Process a list of input images

to=null - Optional. Filename of an output ascii file containing a list of results. Default destination of information in standard log file (print.prt)

toerr=*lev1pt.err* - Optional. Filename of an optional output ascii file containing the image filenames that were unsuccessful. Default=null, no output error file generated.

samp=/line= - Sample and Line coordinate of desired location in the input image

-or-

lat=/lon= - Latitude and Longitude coordinate of desired location in the input image.

GROUP = ISIS_INSTRUMENT

```
     SPACECRAFT_NAME = MARS_GLOBAL_SURVEYOR
    FIRST_LINE_SAMPLE = 1
    CROSSTRACK_SUMMING = 3
    DOWNTRACK_SUMMING = 3
    FOCAL_PLANE_TEMPERATURE = 270.1
    GAIN_MODE_ID = "6A"
    INSTRUMENT_ID = "MOC_NA_A"
    LINE_EXPOSURE_DURATION = 0.482100
    MISSION_PHASE_NAME = "MAPPING"
    OFFSET_MODE_ID = "36"
    ORIGINAL_SPACECRAFT_CLOCK_COUNT = "632523722:237"
    RATIONALE_DESC =
      "Edge of partly exposed crater rim in SE central Terra Meridiani"
    ORBIT_NUMBER = 3841
    LINES = 6784
    SAMPLES = 672
    INSTPARS = "/usgs/isisd/mgsdata/moc_parameters.def.1"
```

GROUP = ISIS_TARGET

```
     TARGET_NAME = MARS
     LONGITUDE_SYSTEM = 180
      LATITUDE_SYSTEM = "OGRAPHIC"
     TARGDEF = "/usgs/isisd/data/targets/mars.def.2"
```

GROUP = ISIS_GEOMETRY

```
     BASE_KERNELS = ("/usgs/isisd/mgsdata/naif0007.tls",
      "/usgs/isisd/mgsdata/MGS_SCLKSCET.00035.tsc")
    SPACECRAFT_KERNELS = ("/usgs/isisd/data/de405.bsp",
      "/usgs/isisd/mgsdata/mgs_washu.bsp")
     CAMERA_KERNELS = "/usgs/isisd/mgsdata/mgs_sc_washu.bc"
     KERNLST = "/usgs/isisd/mgsdata/moc_kernels.def.2"
     NAIF_SOFTWARE_VERSION = "CSPICE_N0050"
     LEV_SOFTWARE_VERSION = "MGS_MOC_1.0"
```

Moclev1 - MOC Radiometric Correction

Radiometric correction results in an "ideal" image, density numbers (DN) proportional to scene brightness, output is "I/F" (alternatively counts/milliseconds). Corrects for 1) global gain and offset instrument operating modes 2) variable sensitivity of each detector in push broom array, 3) even/odd detector offset, 4) normalize sun-target distance, and 5) low-high-low spike at 50 pixel intervals caused by power supply synchronization pulse.

How to run:

TAE>**moclev1**

from=*image.lev0.cub* - process a single image

-or-

fromlist=*lev0.lis* - process all images listed in the ***"lev0.lis"*** file.

delinput="no" - optionally delete the input images after output images are created

Considerations:

- Input images to the **moclev1** procedure must have the **".lev0.cub"** extension in file name.
- Output images will have the extension **".lev1.cub"**.
- Check the **moclev1.err** and **moclev1.prt** files for any errors encountered in processing.

Details:

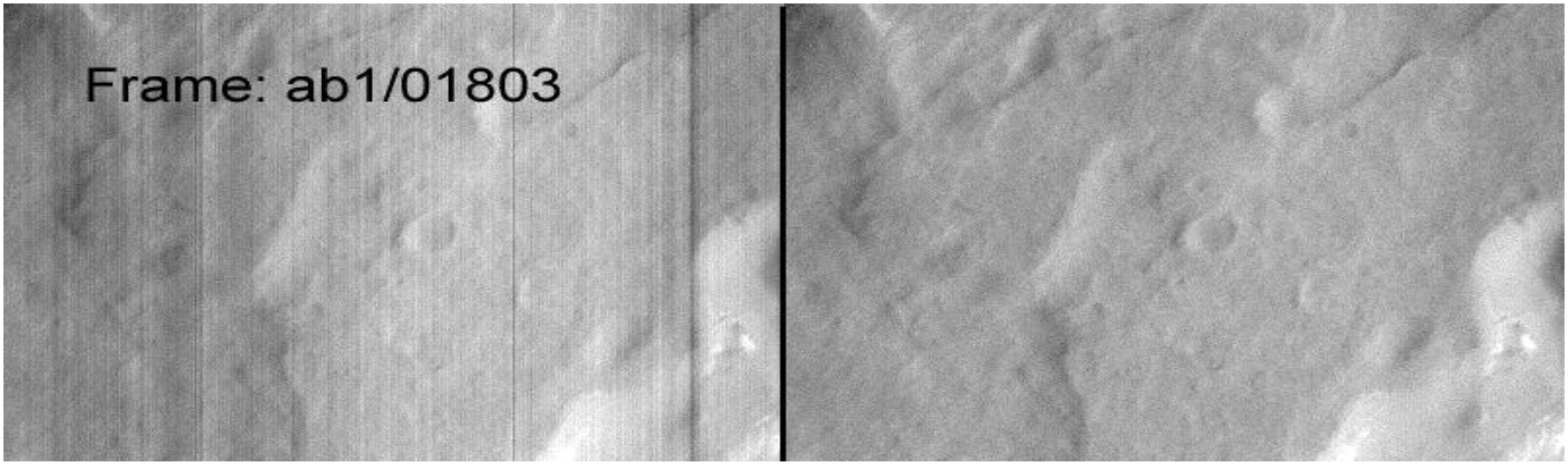
moclev1 procedure runs three programs:

- **moccal** - global gain and offset correction., variable response of push broom detectors, conversion to I/F
- **mocnosie50** - low-high-low spike at 50 pixel intervals
- **mocevenodd** - correct for even/odd detector offset

Moccal - Radiometric Calibration, Caplinger, et. al.

$$O_{(i,j)} = (((I_{(i,j)} - z + off)/gain - g - (dc * exp)) / (exp*dt))*g_{(j)} + o_{(j)})/ (W_0/D_s^2)$$

$O_{(i,j)}$	=	Output pixel at line i, sample j
$I_{(i,j)}$	=	Input pixel
z	=	Instrument offset
off	=	Instrument offset mode value (5 * OFFSET_MODE_ID found in labels)
gain	=	Instrument gain mode value (derived from GAIN_MODE_ID found in labels) (pre-systematic mapping images, gain is the "real" instrument gain, see moccal documentation)
g	=	Instrument fixed dark current at line integration time = 0
dc	=	Dark current build up as a function of line integration time
exp	=	Exposure (line integration time)
$g_{(j)}$	=	Detector dependent gain
$o_{(j)}$	=	Detector dependent offset (currently set to 0 for NA and WA camera)
W_0	=	Counts/millisecond for a 100% lambertian reflector with photometric angles = 0 at 1AU.
D_s	=	Target-Sun distance in A.U.
dt	=	Down-Track summing value for NA, dt = 1 for WA images

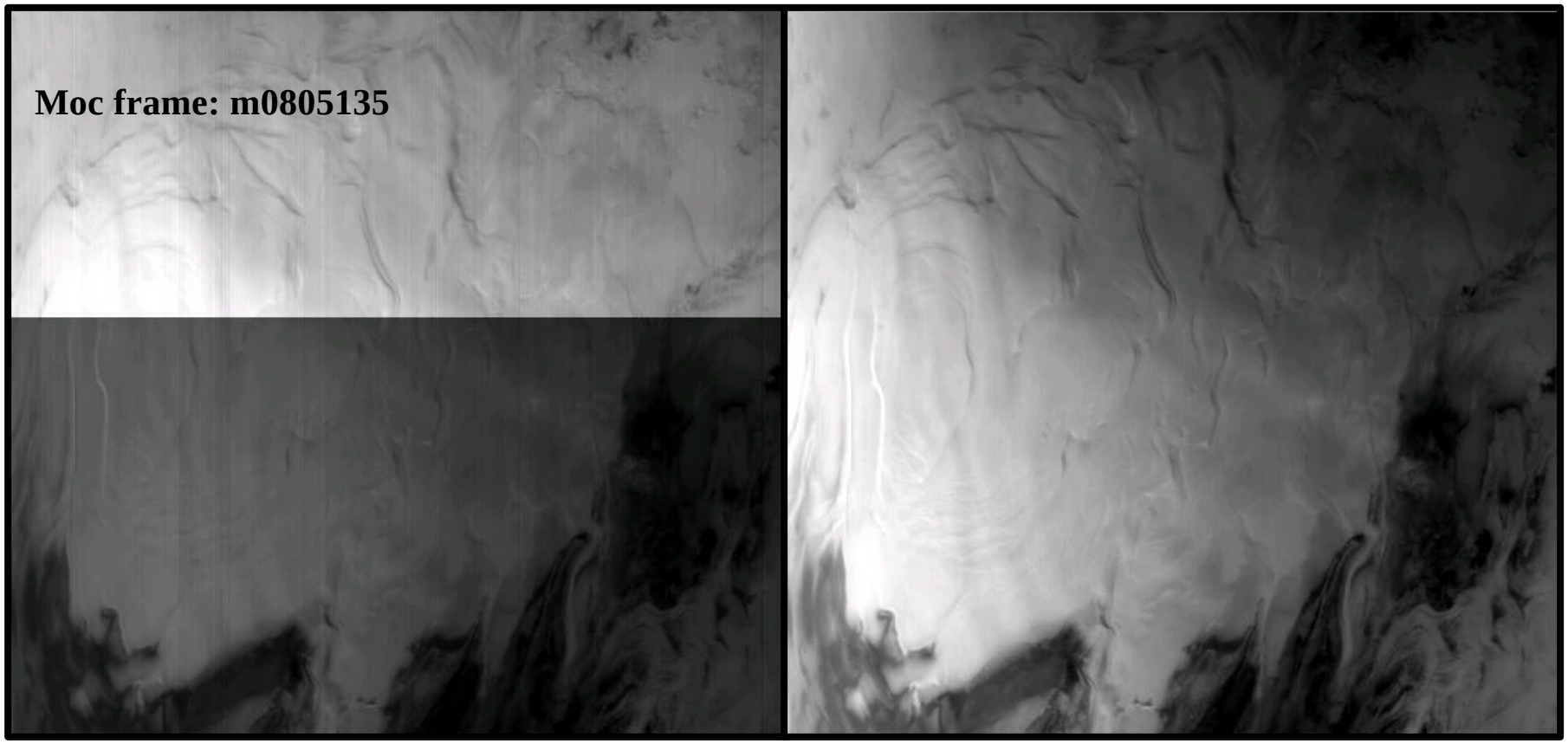


Moccal - Radiometric Calibration

• Wide Angle Gain/Offset Table (wago.tab) defines the times at which the WA camera undergoes a gain or offset camera mode change.

• The times are converted to image line positions and the gain/offset values are updated as moccal corrects the image.

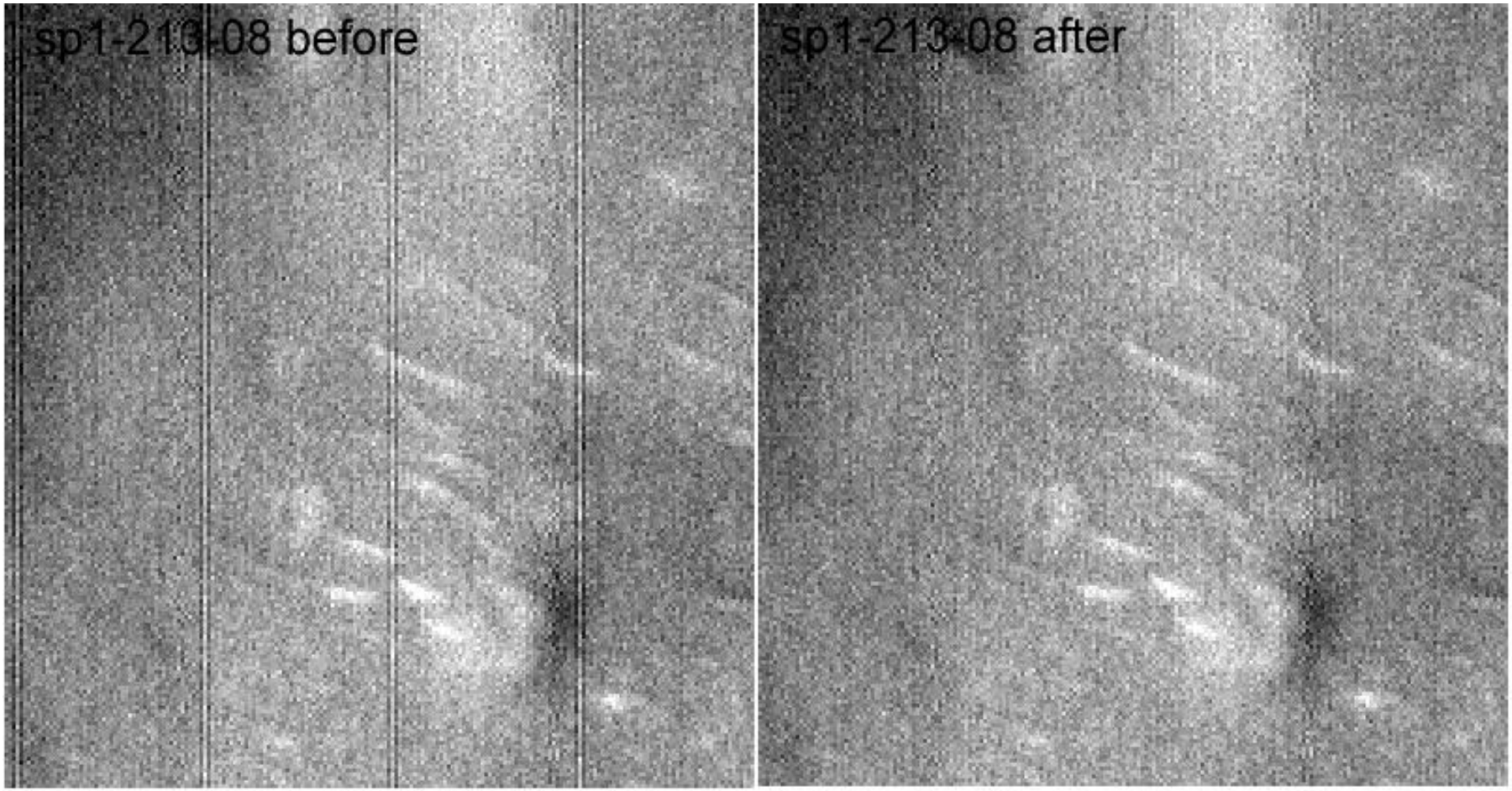
• Time dependent gain/offset modes for WA observations allow the 8-bit imaging system to maintain full dynamic range when transitioning through the pole-to-pole brightness variations.



Mocnoise50 - Correct for Power Supply Synchronization Pulse

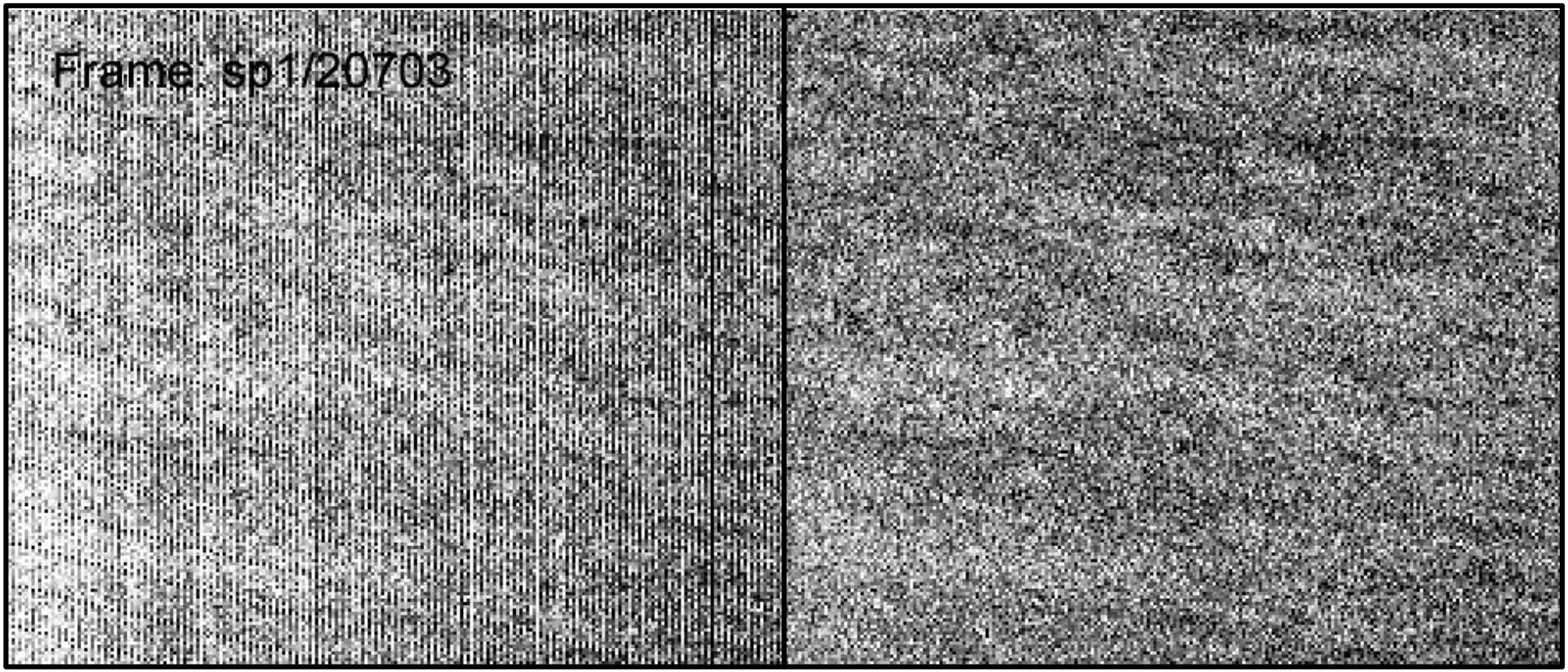
USGS science for a changing world

•Power supply synchronization pulse seen in MOC/NA, 1:1 crosstrack summing imaging .
•Noise problem seen in images with low signal/noise.
•Low-high-low spike seen at exactly 50 pixel spacing.
• Noise pattern seen as vertical strip or diagonal pattern depending on line integration time.
• Program determines amplitude of noise spike using statistics of entire scene. Starting location of noise spike determined for first line, a delta to next line is determined using scene wide statistics.



Mocevenodd - Offset Differences in Even/Odd Detector Readout

- Even/odd detectors have different readout channels resulting in residual offset differences.
- Seen only in MOC/NA 1:1 crosstrack summing images with low signal/noise
- Program obtains averages of even and odd detectors, splits the difference.
- Applies additive correction to detectors.



Moclev1 - MOC Radiometric Correction

Future enhancements:

• WA Blue filter images - Detector dependent gain corrections added to moccal

• WA Red filter images - Improvements on "low-frequency" detector dependent gain corrections

• WA/NA - Improved coefficients on W0 for converting imaging to I/F.

Problems to watch out for:

• Gain/offset global parameters not always correct. Gain/offset parameters found in image labels do not always provide a consistent I/F value. MSSS attributes problems to occasional onboard instrument/software resets.

• Telemetry "bit-hits" can cause MOC image lines to be incorrectly placed in image array. Parts of images can be misaligned from the WA gain/offset table resulting in appearance of improper calibration.

• We have noted that WA gain/offset tables are not always consistent with gain/offset breaks in imaging. More work required to characterize problems.

Mocaspect - MOC Aspect Ratio Correction

"Poor mans" geometry correction. Provides approximate geometry of an image by correcting for the pixel aspect ratio and left-right flip (mirror) for WA/RED images. The aspect ratio is defined as (pixel height)/(pixel width).

Advantages:
- Fast.
- Correction is adequate for many interpretive geology applications.
- SPICE data not required, uses the image index file (**"imgindex.tab"** or **"cumindex.tab"**) from published CDs.
- No rotation of image, efficient storage of output (no black space).

Disadvantages:
- Geometry correction is approximate.
- Additional geometric/photometric processing not possible.
- North is not necessarily up (north azimuth angle defines direction of north, see **mocaspect.log**)
- No correction to image skew (approximately 10 degrees)

How to run:

TAE>**mocaspect**

from=*image.lev1.cub*

-or-

fromlist=*aspect.lis* - process all images listed in the "**aspect.lis**" file.

imgindex="*$ISISMGSDATA/cumindex.tab*" - user may optionally specify a cumulative index file.

delinput=no or yes - default no, a user may optionally delete input files as output files are created.

Considerations:
- Input images to the **mocaspect** procedure must have the **".lev1.cub"** extension in file name.
- Output images will have the extension **".asp.cub"**.
- Geometry information about image is provided in **mocaspect.log** file (contains information about new scale (meters/pixel), and north azimuth angle.
- Check the **mocaspect.err** and **mocaspect.prt** files for any errors encountered in processing.

Mocaspect - MOC Aspect Ratio Correction

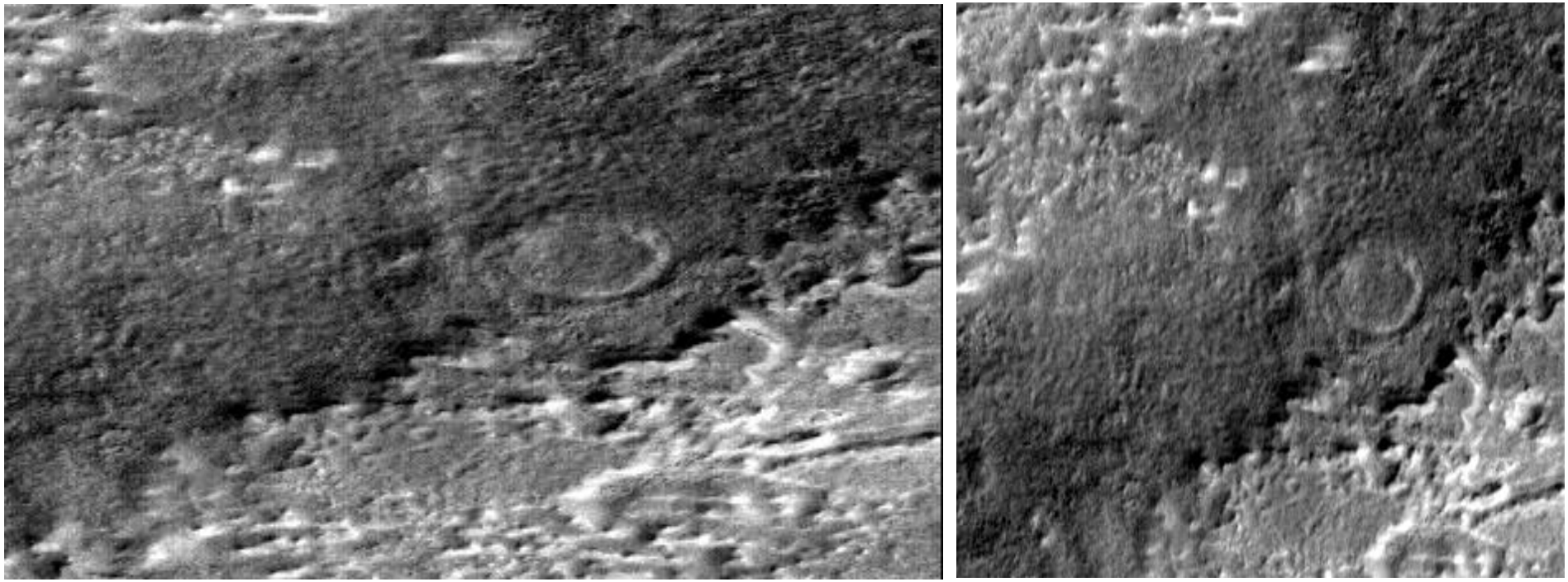
Details:

The **mocaspect** procedure performs the aspect ratio correction by "splitting the difference" in the line and sample direction according to the equation:

line_scale = (aspect_ratio - 1.0)/2.0 + 1.0
sample scale = line_scale/aspect_ratio

For example, an aspect ratio of 1.5 produces a corrected image by increasing the line direction by 1.25 and decreasing the sample direction by .833333.

Before/After Aspect Ratio Correction



GEOMETRY INFORMATION

LEV1GEOPLANE - Compute a choice of latitude, longitude, emission angle, incidence angle, phase angle, resolution for every pixel of an input MOC image. The geometric/photometric information will be stored in the backplanes of the input image cube or in the core planes of a separate output cube.

How to run:

TAE>**lev1geoplane**

from=*image.lev1.cub*	- Process a single image
to=*image.geo.cub*	- To=null (default) will not generate an output cube, but will place the information in the backplanes of the input cube.
Ema = *(y/n)*	- y = Compute the emission angles
Inc = *(y/n)*	- y = Compute the incidence angles
Phase = *(y/n)*	- y = Compute the phase angles
Lat = *(y/n)*	- y = Compute the latitude values
Lon = *(y/n)*	- y = Compute the longitude values

Details:

- Input images must have been processed through Level 1
- Results can be viewed and interactively accessed in *qview* or IDL>*cv*

LEVINIT - This program initializes the geometric processing information for a cube. The ISIS_TARGET, ISIS_GEOMETRY and ISIS_INSTRUMENT group labels are updated. The user has the option of modifying a copy of the system default files that *levinit* refers to. Programs such as *lev1tolev2* and *lev1stats* use the geometry information that is placed in the image labels by *levinit.* Levinit can be run on the Level 0 or 1 images multiple times for desired modifications to the input file image labels.

How to run:

TAE>**levinit**

from=*input.lev1.cub* - Process a single image (must be a Level 0 or 1)

targdef="*target definition file*" - Optional, Example modification: Changing the radii value for map projection purposes

kernlst="*SPICE pointer file*" - Optional, the user may modify a copy of the system default file that gives the location and filename of MOC SPICE kernels.

instpars="*Instrument parameter definition file*" - Optional. This file contains the instrument parameters such as focal length, boresight, etc.

lonsys=360 *(default)* - Longitude System. For Lonsys=180, the longitude values will range between -180 to 180. Lonsys=180 is recommended for images that center on longitude 0 deg. For lonsys=360, the longitude values range between 0 to 360.

latsys=*ographic (default)* - The user may optionally specify a planetographic or planetocentric coordinate system.

Level 2 - Map Projection

Geometrically transform MOC images to a map projection. A user can specify the map projection (Sinusoidal, Mercator, Simple Cylindrical, Polar Stereographic, etc.).

How to run:

TAE>**moclev2**

from=*image.lev1.cub*	- process a single image
-or-	
fromlist=*lev1.lis*	- process all images in "*lev1.lis*" file
mappars="*SIMP:0*"	- Sinusoidal Equal Area, center longitude will default to center of input image(s) unless otherwise specified TAE>***help mappars*** for parameter details
km = 0.24	- default (*null*) is input image resolution in kilometers/pixel

Requirements:

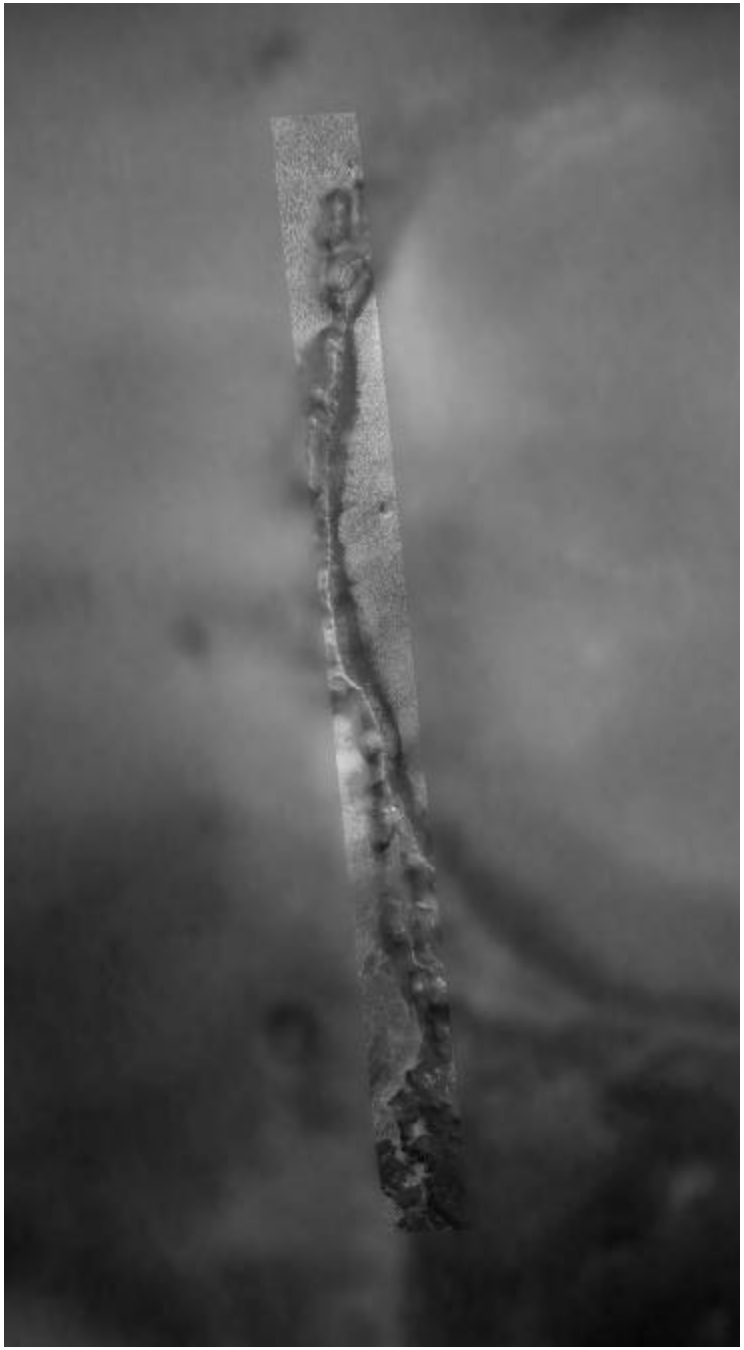
- Input images to moclev2 must have the ".lev1.cub" filename extension
- Output images will have the filename extension ".lev2.cub"
- Check the moclev2.err and moclev2.prt log files for any errors encountered during processing

Details:

moclev2 procedure runs two programs:

- **lev1tolev2** - Generate the transformation file (tfile.dat) for level1 to level2 conversion (input must be a Level 1)
- **geom** - Utilize the transformation file for the geometric transformation of the image ISIS cube file

Note: *Images that are to be mosaicked together <u>MUST</u> be projected to the <u>same</u> center latitude/longitude and map resolution*.



Mosaic of NA-m1102401 overlaid on it's WA-context partner image-m1102402.

Results of adjusted geometry parameter efforts of Eric Eliason and Randy Kirk.

Map projection: Both images projected to Simple Cylindrical
Center Longitude = 0.0
Map Resolution = 0.024 km/pix

Moclevall - Perform level 0, 1, and 2 processing

The **moclevall** procedure performs levels 0, 1, and 2 processing on an image (specified by the **from** parameter) or a list of images (specified by the **fromlist** parameter). This procedure simplifies the processing when a user intents to start with PDS-formatted image and desires to use only level 2 images.

How to run:

TAE>**moclevall**

from=*image.imq* *or* ***image.img*** - process a single image

-or-

fromlist=*raw.lis* - process all images listed in the ***"raw.lis"*** file.

mappars="SINU:" - map projection of level 2 image, default sinusoidal equal-area projection.

km= - kilometers per pixel resolution of output image, default is the resolution of input image.

delinput=no or yes - default no, a user may optionally delete input files as output files are created.

transtab="*translation table*" - optional, a user may optionally enter a PDS to ISIS translation table.

targdef="*target definition file*" - optional, a user may optionally enter a target definition file (see documentation in **levinit** program for more information).

kernlst="*MOC SPICE kernel pointer file*" - optional, a user may optionally enter a pointer file that provides information about location of SPICE kernels. It is possible for a user to construct a pointer file in order to use a different set of SPICE kernels.

instpars="*Instrument parameter definition file*" - optional, a user may optionally enter an instrument parameter file to change boresight pointing, camera focal lengths, etc.

lonsys=360 or 180 - default 360. A user may optionally specify the longitude system to be used in cartographic mapping. 360 = 0 to 360 longitude system, 180= -180 to 180 longitude system.

latsys=*ographic* or *ocentric* - default ographic. A user may optionally specify a planetographic or planetocentric coordinate system.

Moclevall - Perform level 0, 1, and 2 processing

Considerations:

• Input for the **moclevall** procedure is PDS-formatted MOC Standard Data Products (.imq extension) or MOC Decompressed Standard Data Products (.img extension)

• Output images are named the same as input with the **".imq"** or **".img"** extension replaced with extension **".lev2.cub"** to signify an output a level 2 image.

• Check the **moclevall.err** and **moclevall.prt** files for any errors encountered in processing.

• **Warning**: novice ISIS users should never attempt to use the parameters: **transtab, targdef, kernlst, instpars**

• When processing image sets that are geographically located near or on map longitude system boundaries, you should set the **lonsys** boundary to the alternate longitude system.

• The **moclevall** procedure tests to see if an image crosses a map longitude system boundary. If the boundary is crossed then the procedure automatically changes to the alternate longitude boundary system.

Details:

moclevall procedure runs the following procedures:

• **moclev0** - create a level 0 image.

• **moclev1** - create a level 1 image.

• **moclev2** - create a level 2 image.

• Only the level 2 images is kept as output.

LEAVING ISIS

DFORM - (Different format) Convert ISIS cube to raw, tif, or gif

How to use:
TAE>**dform**
from=*input.cub*
- Default output filename will be input.raw, input.tif or input.gif
- Output 8 bit - only

orange=*null (default)* - Optional DN range of values that will be converted to 8bit if necessary. Default will extract the valid DN range of the input file for the user.

dform calls:
- *isis2raw* - to convert to raw
- *dsk2dsk* - to retrieve the valid min/max DN values (default)
- *rawtopgm/pgmtoppm/ppmtogif* - from raw to convert to gif
- */usr/bin/X11/convert* - from raw to tif

Improved Coalignment of MOC and MOLA

For more information on MOC/MOLA coalignment analysis see:
Kirk, R.L, T.L. Becker, E.M. Eliason, J. Anderson, and L.A. Soderblom, 2001, Geometric Calibration of the Mars Orbiter Cameras and Coalignment with Mars Orbiter Laser Altimeter, *LPSC XXXII*, abstract no. 1863.

•Our analysis is based on projection of data obtained simultaneously to a common coordinate system so that the errors in our knowledge of spacecraft position information and ground elevation cancel out.

• We adopted the latest estimate of the MOLA boresight relative to the MGS spacecraft as defined by SPICE MOLA I-Kernel version 2.5, December 5, 2000 (see SPICE file: mola25.ti). This will allow us to use the MOLA adjustments to the MOLA ground track to make precision products from MOC.

• Several MOC NA images containing well defined topographic features (e.g. m0201121, m0202582) and simultaneous MOLA data were projected to the same map projection, taking into account the known timing corrections for the spacecraft attitude control system and the MOLA instrument.

• The 3 Euler angles for the MOC NA were adjusted by trial and error until topographic features identified in both image and MOLA profile agreed to a fraction of the 300-m spacing between altimetric samples. Current result places the MOLA track at pixel 1574 of the NA camera (agreement with S. Anderson analysis , written comm. 2000).

• Euler angles for MOC WA BLUE and RED cameras were then updated to coregister with the NA camera.

MOC and MOLA Euler Angles

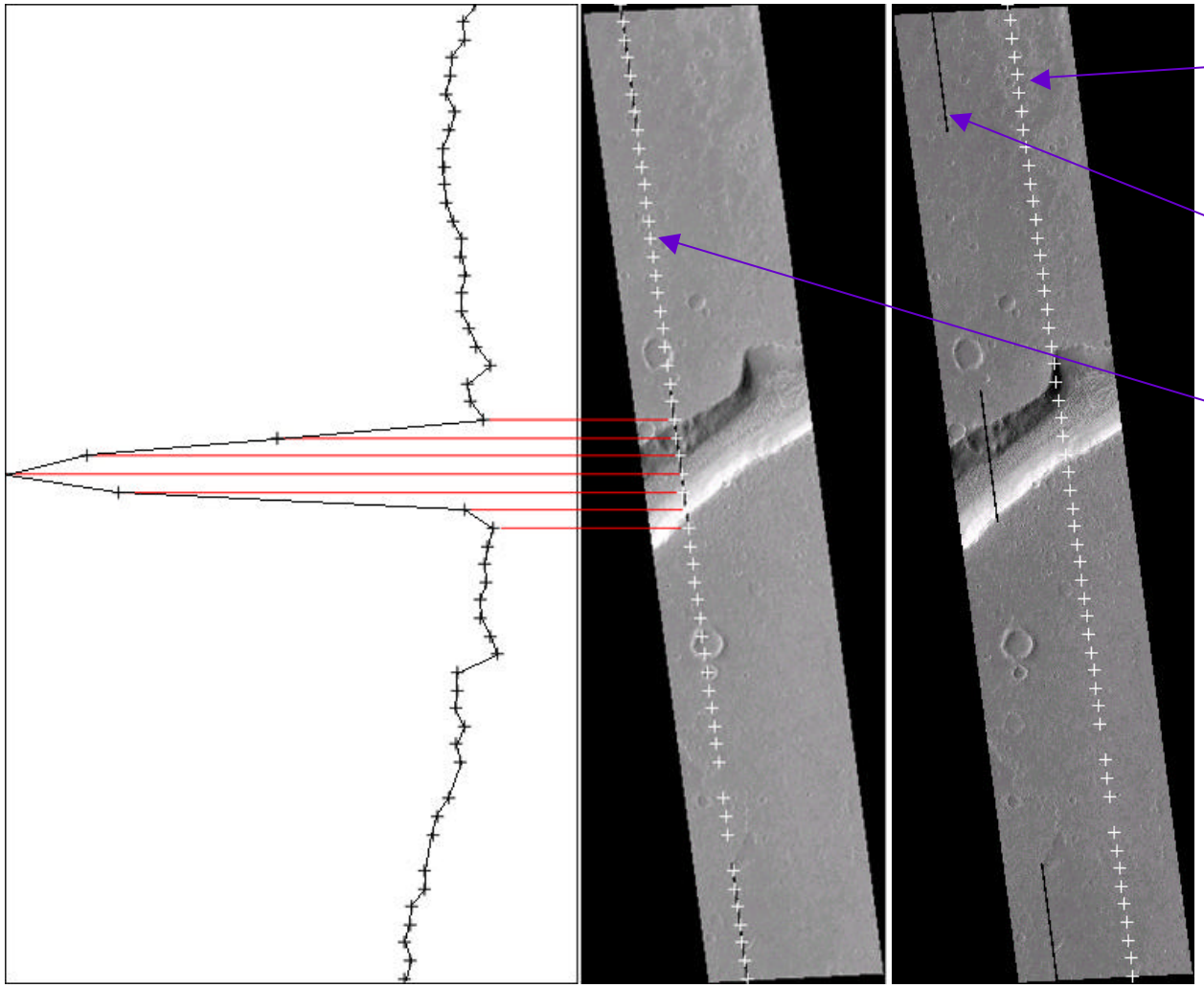
Instrument	Roll(°)	Pitch(°)	Yaw(°)
MOLA	-0.00290	-0.00860	0.05900
MOC-NA	0.11463	-0.07162	0.18000
MOC-WA R	1.04764	-0.45229	-0.78644
MOC-WA B	1.01022	-0.35472	-0.30189

MOC Focal Plane Shift and Scale Factor

Instrument	Scale Correction	Along-Track Pixel Shift (pixels)
MOC-WA R	1.000000	6.7785
MOC-WA B	1.000452	-0.8486

MOC and MOLA Comparative Analysis

Plot of MOC/NA image frame: M0201121 with corresponding MOLA altimetry ground track



Original MOLA altimetry ground track relative to MOC NA camera

MOC/NA detector position 1574 defined as boresight position of MOLA ground track.
(S. Anderson, written comm., 2000)

MOC/NA and WA pointing updated to match MOLA PEDR boresight pointing.

Mocmola - Create a MOLA Altimetry Table for a MOC Image

The **mocmola** procedure provides a convenient method for locating and extracting the MOLA altimetry ground track data that corresponds to the same areal and time coverage of a MOC image. The results of the **mocmola** procedure enables users to compare the MOLA altimetry ground track with a MOC image. **Mocmola** performs the following steps:

• The procedure interrogates the labels of a MOC image (specified by the **"from"** parameter) to determine the starting and ending time of the observation.

• The MOC image time range is compared with the PEDR cumulative index file to determine which PEDR file contains the MOLA ground track data corresponding to the MOC image.

• The MOLA/PEDR file is downloaded from the PDS Imaging Node ftp site (or optionally from a web site specified by the **"url"** parameter).

• After downloading the PEDR file, the procedure extracts only the MOLA ground track data that corresponds to the areal coverage of the MOC image.

• **Mocmola** recomputes the latitude and longitude coordinates of the PEDR ground track with the same SPICE kernels and cartographic assumptions used in the geometric processing of the MOC image. This ensures that the MOC and MOLA geographic positions will be internally consistent.

• An ISIS binary table is created with the MOLA altimetery data. Optionally, an ASCII ISIS table can be created with the **"asc**" parameter.

• The ISIS display program **qview** can be use to plot the MOLA ground track on a MOC image. Optionally, the ASCII table can be used by an analyst's own institutional graphics package.

Mocmola - Create a MOLA Altimetry Table for a MOC Image

How to run:

TAE>**mocmola**

from= ISIS cub file (with extension .cub) or PDS formatted file (with extension .img or .imq)
Specify the MOC frame for which you want the corresponding MOLA altimetry ground track.

tbl= output file - binary ISIS table file to store the altimetry ground track data for a MOC image.

asc= optional output file - ISIS ASCII table file containing altimetry ground track data for a MOC image.

url= optional URL specification for location of MOLA/PEDR data base. The URL defaults to the PDS Imaging Node ftp web site for downloading MOLA/PEDR data. (An optional URL specification for the PDS Geosciences Node can be used: "http://wundow.wustl.edu/geodata/mgscd/".

idir= optional directory specification for the location of MOLA/PEDR archive. This parameter can be used if the MOLA/PEDR archive already exists on your computer system. If **"idir"** is specified, **mocmola** will use this directory area in lieu of internet access to the MOLA/PEDR data products.

index= optional MOLA/PEDR modified cumulative index file. The default file is located in the ISIS data directory: **$ISISMGSDATA/pedcmidx_update.tab"**

Mocmola - Create a MOLA Altimetry Table for a MOC Image

Considerations:

• The **mocmola** procedure finds only the MOLA/PEDR data that were acquired simultaneously with the MOC frame.

• We recommend that you always name your ISIS binary table file the same as the MOC image name but with the extension **".tbl"**. This will make it easier to use the **qview** program for comparing MOC and MOLA data.

• Many of the MOC images acquired during the pre-systematic mapping phase of the mission do not have corresponding MOLA ground track observations.

• Occasionally a MOLA ground track does not exist for a MOC frame for the systematic mapping phase.

• A MOLA/PEDR file is typically about 30 megabytes so download times may be long. Use the **"url"** parameter when an internet site, such as the PDS Geosciences Node, may provide better download times for your particular institution.

• **Mocmola** is most time efficient when you are able to store the entire MOLA/PEDR archive on line. This avoids the need to use internet services to down load large MOLA/PEDR files. Use the **"idir"** parameter to point to the directory area of the MOLA/PEDR files.

• If it is more efficient to copy MOLA/PEDR files from your MOLA CD-ROM volume set then you can run the **molasearch** and **mola2isis** procedures in lieu of **mocmola**. (see details below).

Details:

mocmola procedure runs the following procedures:

• **molasearch -** find a MOLA/PEDR that corresponds to MOC image frame.

• **mocftp -** download a MOLA/PEDR from PDS Imaging Node ftp site.

• **mola2isis -** extract MOLA ground track from PEDR and update coordinate system using same SPICE kernels as the MOC image.

Molasearch/Mola2isis - Alternative to Mocmola Procedure

The **molasearch** and **mola2isis** procedures offer an alternative to the **mocmola** procedure when a user has inadequate web resources to efficiently download large MOLA/PEDR files. The procedures enable users to copy the MOLA/PEDR from their MOLA CD-ROM volume set in lieu of the internet download step.

• The **molasearch** procedure identifies the MOLA CD-ROM volume and PEDR file corresponding to the MOC image (specified by the **"from"** parameter).

• The user can copy the MOLA/PEDR file from the CD-ROM volume to his working directory area using standard unix commands.

• The **mola2isis** procedure reads the MOLA/PEDR file to create the ISIS table as described in the **mocmola** documentation.

How to run:

TAE>**molasearch**

from= ISIS cub file (with extension .cub) or PDS formatted file (with extension .img or .imq)
to=output file containing name of PEDR file (use this information for copying PEDR MOLA from CD volume)
start= optional, J2000 start time for desired MOLA ground track observations (in lieu of **from** parameter)
stop= optional, J2000 stop time for desired MOLA ground track observations (in lieu of **from** parameter)

Copy PEDR file from MOLA volume identified in the output file specified by **"to"** parameter.

TAE>**mola2isis**

from= ISIS cub file (with extension .cub) or PDS formatted file (with extension .img or .imq)
pedr= PEDR file identified by **molasearch**.
tbl= output file - binary ISIS table
asc= optional output file - ASCII ISIS table
idir= optional directory specification for location of PEDR file.
start= optional J2000 start time (in lieu of **from** parameter)
stop= optional J2000 stop time (in lieu of **from** parameter)

ISIS in the IDL Environment

Cube Visualization with CV

Cube Visualization program. Allows interactive display of ISIS cube files. (Most other 2D or 3D image files can also be displayed by using an ISIS detached label or by using the "raw2isis" program to convert the data to a standard ISIS cube file.) This includes functions such as displaying selected spatial-spatial or spatial-spectral slices through the cube, "movie" display of different slices, reporting cursor coordinates and associated pixel values, zoom and roam of displayed images, plotting spectra at selected spatial locations, plotting spatial profiles, and hard-copy output. "cv" can be used for visualization of both single-band images that contain a large spatial extent and imaging spectrometer data cubes that contain many wavelength bands**.**

How to run:

IDL> **cv**

Note: **cv** does not require any arguments as it is completely widget driven.

Features:

•Produces postscript output of plots and images
•Designed specifically for multi-spectral image analysis but can be useful for traditional 2-D images
•Extensive support for backplane visualization and interactive integration of spatial referencing
•Provides interactive Region Of Interest (ROI) analysis tools including storage and retrieval in ISIS cube (backplanes)

ISIS in the IDL Environment

Reading ISIS cubes into IDL

readisis ISIS data ingestion application. Provides a method to read ISIS cube core data directly into the IDL environment.

How to run:
IDL> **readisis**,'isis.cub', data[,sfrom=][,special=]

'isis.cub' - name of an existing ISIS cube file

data - an IDL variable that will return the ISIS data. This will be a 2-dimensional array of the first band if "sfrom" is not specified, otherwise will be 3-D.

Keyword Parameters
sfrom - ISIS subcube specifier. Defaults to "::1" which reads the first band from the cube

special - 1 -*or*- 5 floating point array of values to replace all ISIS special pixel values in cubes with (example: special=0). If not specified, no translation will occur.

Considerations:
•Always returns the data as floating point values
•ISIS floating point special pixel values are especially unfriendly representations and will almost always exist in ISIS cubes, so beware!
•Reads all 3 supported ISIS storage orders and types
•**readbackplane** can be used to read individual ISIS backplanes into IDL in the same fashion

ISIS in the IDL Environment

Writing ISIS cubes from IDL

writeisis ISIS export application. This is a routine that can be called from an IDL program and thus can be used for writing IDL programs that perform specialized processing of ISIS cube files.

How to run:
IDL> **writeisis**,'isis.cub',data[,special=]

'isis.cub' - name of output ISIS cube file

data - an IDL variable that contains floating point data.

Keyword Parameters

special - 1 -*or*- 5 floating point array of values that specify which actual data values to replace with ISIS special pixel values. Example: special=0.0 will replace all values of 0.0 in the array with the ISIS NULL special pixel value.

Considerations:

- Data arrays must always be IDL float data type
- Can write 2 or 3 dimensional arrays
- Doesn't write backplanes (see ISIS application **bandcopy** for this capability)

ISIS in the IDL Environment

ISIS data to other Image Formats

isis2std writes ISIS images to JPEG and TIFF data formats

How to run:
IDL> **isis2std**

Note there are no parameters to **isis2std** as it is completely widget driven

Considerations:

- Produces 3-band color images or black and white
- Supports interactive independent stretching of each band
- Resizes large images to 700x700 but writes output in full spatial size

Advanced IDL Programming Topics

Importing and Exporting data

ISIS provides access to a subset of its features in IDL using the **CALL_EXTERNAL** routine. This routine loads a shared object into the IDL environment and programmers can gain access to ISIS's higher level routines written in C and FORTRAN from within their programs. This object, named ***isis2idl.so***, resides in the $ISISIDLLIB directory. It contains specially designed routines that adhere to the **CALL_EXTERNAL** specifications. All input and output in this environment utilizes this scheme to import and export ISIS data into and out of IDL. Programs have been written to use a number of these routines that provide access to advanced features in the ISIS environment.

IDL intrinsics - getting other data into ISIS

IDL intrinsic routines such as **READ_JPEG, READ_TIFF,** etc, can be used to get data into IDL and then out to ISIS cube files.

These features can be combined to provide a powerfully rich environment in IDL without having to leave it.

Specialized programming techniques

The IDL/ISIS C routine, *I_init.c,* shows examples of a routine that is callable from IDL program using the **CALL_EXTERNAL** routine. It is very similar to main C program arguments, namely using "*argc*", "*argv*" variable arguments for *all* these types of routines.

The following is an example of a call from IDL to an ISIS external C routine:

```
ioflag = 1L
limits = [[1L,ns], [1L,nl], [band,band]]
ret =call_external(isislib, 'I_q_lio_cbrick', $
            fid, ioflag, limits, diskbuf)
if ret NE 0 then begin
    cv_err_msg, 'Error reading band number ' + string(band)
    goto, read_band_done
endif
```

"isislib" is set to the location of the shared ISIS object, ***isis2isl.so***:

```
isislib = getenv('ISISIDLLIB')
if strlen(isislib) EQ 0 then begin
    print, 'Before running READISIS, you must set the environment
variable ISISIDLLIB'
    print, '  to the directory that contains the "isis2idl.so" file'
    return
endif else isislib = isislib + '/isis2idl.so'
```

Programming Example

The IDL program, **readisismap**, will retrieve mapping parameters as well as data from the ISIS cube file and return it to the IDL environment. It illustrates a wide range of **CALL_EXTERNAL** routines.

How to run:
IDL> **readisismap**, `FILE, DATA, PROJ, LONDIR, MAPARS [,SFROM=] [,SPECIAL=]`

FILE - name of an existing ISIS level2 cube file
DATA - an IDL variable that will return the ISIS data
PROJ - returns the map projection name
LONDIR - returns direction of positive longitude
MAPARS - returns a 13 element **double** array (see documentation for these values). These keywords are all contained in the IMAGE_MAP_PROJECTION group in the ISIS label

Features:
•Performs **readisis** functions
•Demonstrates how specific ISIS keywords can be read from the labels and returned to IDL
•Shows how C strings and IDL strings are manipulated