MARS 2001 ODYSSEY

GAMMA RAY SPECTROMETER

GAMMA DATA PROCESSING

Version 4.1

07/01/2004

Prepared By:
GRS TEAM

Table of Contents

1.0 INTRODUCTION

The Mars Odyssey Gamma-Ray Spectrometer (GRS) is a suite of three instruments working together to collect geochemical data from the surface of Mars. The instruments are a gamma subsystem (GS), a neutron spectrometer (NS), and a high-energy neutron detector (HEND). The instruments are complimentary in that the neutron instruments have much better counting statistics and can sample to greater depths than the GS, but the GS determines the abundances of many more elements. A full accounting of the GRS instrument suite can be found in Boynton et al., 2004. The purpose of this document is to describe each step of the gamma data processing from ingestion of the raw spacecraft telemetry into the University of Arizona GRS database through spectral summation.

The GS collects a new gamma spectrum approximately every 20 seconds, 360 times per orbit, resulting in approximately 4200 such gamma spectra[1] collected each day. The ~20 second collection interval, corresponding to 1-degree on the planet's surface, was chosen due to data transfer rate limitations. Once a gamma spectrum is collected it is stored on the spacecraft and held for periodic download. The gamma spectra and telemetry data are downloaded periodically from the spacecraft for receipt by the Deep Space Network and insertion into the Jet Propulsion Laboratory's (JPL) Telemetry Data System (TDS). The University of Arizona (UA) queries the TDS for the most recent telemetry dataset. The dataset is output to a spooler that passes data to the UA. Raw telemetry data are received by the UA, and a number of automated computer processes are run to ingest the data into a database, and to transform the data into spatially, temporally and energetically registered spectra.

Data are retrieved from the TDS by the GRS_tl process (see Section 2.1) that reads the raw data packets, extracts the necessary information and outputs the extracted packets to a socket and then to a local spooler for ingestion into the database. The ingest process (see Section 3.1) is the mechanism by which GRS data are fed into (or "ingested" by) the UA database. The ingest process reads the locally spooled data and inserts it into the appropriate database tables. Initialization of the ingest process sets up the necessary connections to the SPICE kernel library information (see Section 3.2), so that the appropriate timing and spatial information can be processed for each collection interval. SPICE kernel information is obtained using a several additional processes.

Once data are ingested into the UA GRS database, additional processes are run to transform the telemetry into a scientifically useful data product. The first steps in the processing are the

---

[1] Depending on the context, a collected gamma spectra might also be referred to as a "collection interval", or as a "GRS pixel", which has been shortened, for better or worse, to "pixel". Be warned that the word "pixel" is technically inappropriate in describing gamma spectra, for it is by no means a picture element, per se.

calculation and addition of timing information, spatial information and spacecraft orbital information to the raw telemetry (see Section 3). These processes include calculation of the time at the center of each collection interval, the spacecraft position and ground position at the center of each collection interval, insertion of the spacecraft orbit number and insertion of GRS commanding information into the UA database. The completion of these processing steps results in the Experimenters Data Record (EDR) Planetary Data Systems (PDS) data products.

The second level of gamma data processing includes all the steps necessary to convert raw gamma spectra into energetically calibrated gamma spectra (Corrected Gamma Spectra Intermediate Data Record, CGS IDR data product). This processing begins with Engineering Smoothing (eng_smooth, see Section 4.1). The Engineering Smoother uses a Gaussian-weighted smoothing algorithm to remove noise from the GS engineering data (e.g. temperatures, currents and voltages), and inserts the results into the smoothed column of the engineering tables. The next process (b170K_calc, see Section 4.2) calculates the temperature for an electrical component in the GS, (170K board) that is not measured directly. The modeled temperature, which is used in later processing, is then inserted into the database. The third process (digital_hk, see Section 4.3) extracts GS electronic settings information known as digital housekeeping from the raw data and stores it in the database. The engineering interpolator (interp_eng, see Section 4.4) then queries the database for all gamma spectra records where the associated engineering data fields have not yet been computed by previous runs of the interpolator (i.e., the engineering fields are null). The program queries the engineering tables for the closest engineering records collected within an hour before and an hour after the gamma record of interest and does a linear interpolation between the two engineering records to fill in the engineering fields in the gamma record of interest. If an engineering record does not exist within the hour before or the hour after the gamma record of interest, than the gamma record is marked as having no engineering readings, and is not processed further. The next process, fill_apps_gain, (see Section 4.5), uses the extracted digital housekeeping information that was stored by the digital housekeeping program and inserts one of the settings into each spectrum record. The inserted setting is the shaping amp gain, which is the setting of the variable gain stage in the electronics. The gain values are used by the correction process. The auto_bad process (see Section 4.6) is then run to identify and flag any gamma spectrum records that do not meet particular criteria and should not be processed further. The correction process (correction, see Section 4.7) is used to calculate the gain, offset, and linearity of the individual spectra based on the measured temperatures of the various spacecraft subsystems. Since there are not enough gamma counts in any given collection interval to establish a calibration, the correction process re-bins the counts in each spectrum to align the channels in all of the spectra to a common energy scale. Once the correction process is complete, there is a check (vector_check, see Section 4.8) of the SPICE calculated spacecraft vectors, and the position on the surface of Mars being sampled during a collection interval.

The third stage of gamma processing combines the corrected gamma spectra to produce temporally and spatially binned data with sufficient counts to yield reliable statistics (Summed Gamma Spectra, SGS IDR Data product). The spectra are summed together over discrete temporal and spatial regions with the summit_db process (see Section 5.1). The resulting summed spectra are a scientifically useful product in that they contain enough counts to yield reliable statistics.

The following document describes in detail each process used to transform the raw 2001 Mars Odyssey Gamma-ray Spectrometer Gamma Sensor telemetry data into a series of scientifically useful data products. This document is intended as the definitive source of information for the gamma data processing algorithms and methodology.

2.0 LEVEL 0 PROCESSING

2.1 GRS_TL

GRS_tl is the process by which telemetry data down-linked from the 2001 Mars Odyssey spacecraft is transferred from the JPL TDS to the UA and prepared for ingestion into the UA GRS database. Data packets are wrapped with specifically formatted headers at each phase of data transfer. The GRS_tl program is designed to remove any or all of the header information and transform data packets to a useable form. GRS_tl receives input data from any of a number of input sources (e.g. raw telemetry), strips out the GRS specific data, and outputs that data in the requested format. In the case of telemetry data, a process on the JPL Spacecraft Operations Processing Computer (SOPC) called *stot* retrieves selected Standard Format Data Unit (SFDU) packets from the TDS via a query server, and sends the retrieved SFDU packets to a socket. A connection is made between the socket and GRS_tl, and packet data is passed to GRS_tl.

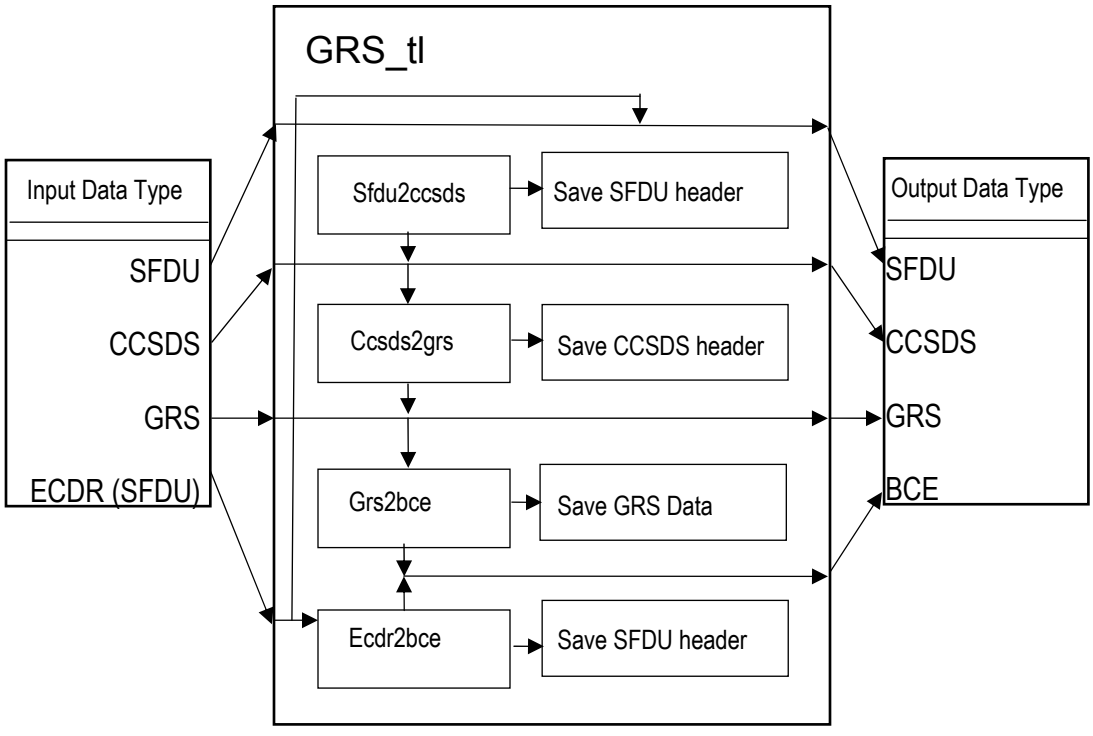


Figure 1. GRS_tl data flow diagram.

The stream of SFDU data packets is read in from the socket. The SFDU packet consists of a primary label, an aggregate header (Compressed Header Data Object, CHDO), up to 4 headers (Primary [required], secondary, tertiary, quaternary), and an optional data CHDO. The primary label and headers are stripped from the packet and are written to a file "SFDU.hdr." The remaining information is a Consultative Committee for Spacecraft Data Systems (CCSDS) packet, consisting of header information and data. The CCSDS headers are removed and written to a file "CCSDS.hdr." The remaining data is then in the form of a GRS packet, and includes the data as output by the GRS instrument suite. The GRS data packets consist of a telemetry header structure, a data type specific data structure, and an appended checksum. The last step in the process is to convert the GRS data packets to BCE (Bench Checkout Equipment) type packets. BCE type packets are a suite of data type specific packet formats with a common header structure that were developed for data transfer and handling during pre-flight tests. The packet definitions were found to work well, and have been modified only slightly during flight. BCE packets are formed by stripping the telemetry header from the GRS packet and replacing it with the BCE common header structure followed by the data type specific data structure. Any needed regrouping or decompression of data occurs in the translation from GRS to BCE data format.

Data with an invalid telemetry header is flagged as "invalid header." If the checksum is not correct, then the packet is flagged as "bad checksum." The BCE formatted data are sent either to a data spooler (similar to a print spooler) to wait for database ingestion or to be written to files. If BCE output is written to files, it is placed in a Central Electronics Box (CEB) Telemetry structured directory which can then be used for data validation. The CEB telemetry data structure is a top level directory with standard sub-directories categorized by data type.

GRS_tl can also be used to translate data from one packet type to another (Figure 1.) The input data stream can consist of SFDU packets, CCSDS packets or GRS/CEB data packets. The packets are read, parsed, and translated to an output packet format. Output types can be SFDU, CCSDS, GRS data or BCE formatted packets. The output packets can then be sent to a socket and/or written to a file. The only restrictions on data type transformation are that BCE packets can not be used as GRS_tl input, and the data types can not be "up converted", for example CCSDS packets can not be transformed into SFDU packets.

3.0 LEVEL 1-A PROCESSING

3.1 INGEST

The ingest process is the mechanism by which data is ingested into the UA GRS database. The input data used by ingest are the BCE packet type output from the GRS_tl program. Ingest receives data from the spooler and inserts it into the appropriate database tables.

Ingest initialization sets up the necessary connections to the SPICE kernel information (see section 3.2 for a detailed description of SPICE), the database, and the input data. The SPICE kernel files are opened and loaded, and a connection to the database is established. Once the appropriate connections are made, the data ingestion begins.

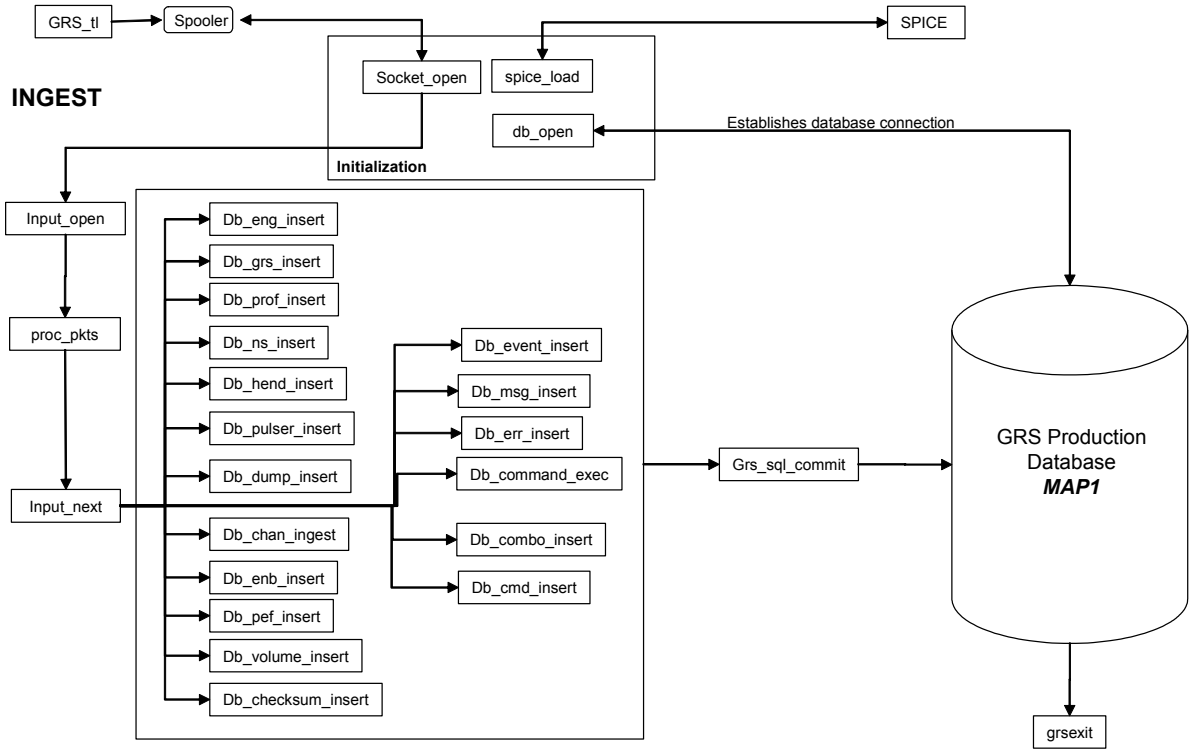


Figure 2. Ingest process data flow.

The *Ingest* function, which waits for the client on the socket, calls the *input_open* procedure. *Input_open* initializes the socket on the designated port. Once the socket is open, the *proc_pkts* function processes each packet of data through the *input_next* procedure. *Input_next* reads the socket header to get the total number of bytes in the packet, and then reads the common header to get the sequence bytes. The rest of the packet is then read.

The data type, read from the common header, is returned, determining the next step in the processing. One of eighteen different procedures is run to insert packet data into the database (Figure 2.) The insertion procedure is based on the data type being processed. If the data type returned is less than or equal to 0 an error message is returned stating that the packet was not inserted.

All data types insert records into a database table called the header table (see Table 1). This table is used to keep track of the time and type of each data packet inserted into the database. For each data packet the ingest process receives from the spooler, the spacecraft receive time and data type are checked against the records in the header table to ensure that the packet is not a duplicate of a previously ingested data record. If the packet is a new piece of information, a new

header table record is created. If the packet has a duplicate spacecraft receive time and data type entry in the header table than the packet is not ingested into the database.

Table 1. UA database Header Table

Header Table	
Data Column Name	Data Column Description
sc_recv_time	Spacecraft time of first packet, in ticks.
hdr_type	The type of header, GAMMA, NS, HEND.
utc	The UTC at the middle of the pixel.
utc_milliseconds	The millisecond portion of the UTC time.
orbit	The GRS orbit number for the pixel
sc_orbit	The Mars Odyssey orbit number. (see Section 3.4.1 for orbit differences)
bad_code	A code with various bad flags that have not yet been defined.
sc_ev_time	Spacecraft time at the middle of the pixel.
ceb_time	The CEB time at the beginning of the pixel.
earth_recv_time_day	The days since Jan 1, 1958 that the data was received.
earth_recv_time_mil	The milliseconds portion of the receive time.
mission_phase	The phase of the mission, Cruise = 0, Mapping = 1.
experiment_id	The experiment number, these have not yet been defined.
filled_sctime	Sc_time from the header record that the GRS Orbit was filled from, (see Section 3.4.2 for details).
eng_sctime	Unique time of the engineering packet
gam_sctime	Unique time of the gamma packet
pul_sctime	Unique time of the pulser packet
msg_sctime	Unique time of the message packet
dmp_sctime	Unique time of the dump packet
err_sctime	Unique time of the error packet
prf_sctime	Unique time of the profile packet
evp_sctime	Unique time of the event piece packet
evt_sctime	Unique time of the event combo packet

Header information is extracted from the data packet and a spacecraft event time (spacecraft time at the middle of the collection interval) and UTC time corresponding to the spacecraft event time are calculated by a SPICE function. In order to calculate these times, the pixel (collection interval) duration must be known. The pixel duration sub-routine determines the proper pixel duration by querying the parameter table for the most recent pixel duration record preceding the spacecraft receive time of the gamma record to be inserted into the header table. The following equation is then used to determine the number of seconds and ticks for input into the SPICE routines:

$$CEB\_time + (pixel\_duration / 2) / 1000 + delta\_time = Seconds.Partial\,seconds \quad \textbf{eq. 1}$$

$$Partial\,seconds * 256 = ticks \quad \textbf{eq. 2}$$

$$SPICEstring = seconds.ticks \quad \textbf{eq. 3}$$

where CEB time is the CEB time of the gamma pixel and Delta time is the time difference between the CEB time and the spacecraft clock time. The resulting SPICE string is then passed to a SPICE routine that returns the spacecraft event time and the UTC time at the middle of the collection interval.

Once the times are calculated the following data fields are inserted into the header table: spacecraft receive time, header type (equivalent to data type), UTC time, pixel number (1-360 per orbit), bad code, central electronics box (CEB) time, spacecraft event time (sc_ev_time, the time in ticks at the middle of the collection interval), earth receive time day, earth receive time milliseconds, mission phase, and experiment identification. An Oracle stored procedure inserts the spacecraft event time of the packet into the packet type specific time field (i.e. eng_sctime, gam_sctime, pul_sctime, msg_sctime, dmp_sctime, err_sctime, prf_sctime, evp_sctime, evt_sctime). Once the header table record is completely filled (see Table 1), the data type specific insertion process inserts the rest of the packet information into the appropriate data type specific database tables, such as one of the engineering tables, the gamma spectrum table, the neutron spectrometer table, the HEND table, the profile table, the pulser table, or one of the command or message tables.

There are approximately seventy identically formatted engineering data tables (see Table 2) in the UA database. The ingest process inserts the raw engineering data and spacecraft time (eng_sctime) into each engineering table based on the engineering parameter name. The raw engineering values as received from the spacecraft are in a digital number (DN) format. In order to facilitate data reduction the digital numbers are converted to engineering values by a database trigger that is called by the ingest process. The trigger uses an appropriate polynomial conversion factor for the value, and then inserts the converted value (eng_val) into the database table. The polynomial conversion factors for each engineering parameter were derived from ground calibrations and can be found in Appendix A.

Table 2. Engineering table format

Engineering Table Format	
Data Column Name	Data Column Description
Sc_time	The spacecraft time in 256^{th} of a second when the reading was taken.
Sc_fine_time	The spacecraft time in 65536^{th} of a second beyond the sc_time.
Ceb_time	The CEB time when the reading was taken..
Raw_val	The DN value of the reading.
Eng_val	The transformed value of the reading.
Smoothed_val	The smoothed value of the eng_val (see Section 4.1 Engineering Smoothing)

Once the data type specific insertion process has inserted the three science data type (GS, NS, and HEND) packet information into the appropriate database tables the spatial portion of these records (see Table 3) are calculated with the spatial information SPICE routine. The resulting spatial information is then inserted into each data record based on spacecraft time.

Table 3. Spatial data fields.

Spatial data fields associated with each GS, NS, and HEND record.	
Data Column Name	Data Column Description
Grs_lat	Latitude at the center of the collection interval
Grs_lon	Longitude at the center of the collection interval
Scpos_inert	The geometric position of M01spacecraft with respect to Mars in the 'MARSIAU' inertial frame at the input epoch 'et'. The units are km.
Scvel_inert	The geometric velocity of M01spacecraft with respect to Mars in the 'MARSIAU' inertial frame at the input epoch 'et'. The units are km/sec.
marspos_instr	Position of the sub-spacecraft point as seen from the spacecraft in the instrument frame. Units=km.
marsvel_instr	Inertial spacecraft velocity direction rotated to the NS instrument frame. units=km/sec.
scpos_mars	Spacecraft position in mars fixed coordinates.
instrboresight_mars	sub instrument boresight in mars fixed coordinates
subsc_mars	Sub-spacecraft vector in mars fixed coordinates. This value has been zeroed out, as it is essentially the same as the instrument boresite vector. This value must be within about 0.1 degree of the instrument boresite vector.
delta_theta	Difference between GSH +y direction and true north. Only calculated for gamma data.
pointing	was pointing data available
intersecting	does line of sight intersect planet
scalt	areocentric altitude of the sub-spacecraft point in Mars-fixed rotating frame ('IAU_MARS'.) The units are degrees and km.

3.2 SPICE PROCESSES

The Navigation and Ancillary Information Facility (NAIF) Node of the Planetary Data System (PDS) collects and maintains the SPICE information system. SPICE (Spacecraft, Planet, Instrument, C-matrix, Events) is a means for providing scientists with geometric and event data and related tools useful in the interpretation of science instrument observations returned from planetary spacecraft. SPICE data files, called kernels, exist for spacecraft trajectory (S); planet ephemeris and associated physical and cartographic constants (P); instrument information, including mounting alignment and other relevant geometric information (I); orientation of spacecraft structures upon which science instruments are mounted (C); and spacecraft and ground data system events, both planned and unplanned (E) (NAIF, http://pds.jpl.nasa.gov/naif.html). Table 4 is a description of each of the SPICE kernel files.

The SPICE toolkit, provided by NAIF, contains a collection of sub-routines that allow an exchange of information between GRS data and SPICE kernel information. For gamma data processing, the SPICE kernel information is necessary to convert spacecraft times to UTC times, and allows the calculation of the spatial properties of a collection interval. The three processes that follow describe how the SPICE kernel information is obtained and made useful to all the gamma data processing routines.

Table 4. SPICE Kernel Definitions

Kernel Name	Description
SPK	Space vehicle ephemeris (trajectory)
PcK	gravitational model of Mars, Mar_IAU_2000
IK	Instrument information, none applicable to GRS
CK	Instrument platform (e.g. spacecraft) attitude
FK	Definitions of and specification of relationships between reference frames (coordinate systems)
LSK	Leapseconds Tabulation, Used for UTC <--> ET time conversions
SCLK	Spacecraft Clock Coefficients, Used for SCLK <--> ET time conversions

3.2.1 Naif_ftp.tcl

Naif_ftp.tcl is a process that automatically updates the UA database SPICE kernels from the NAIF File Transfer Protocol (FTP) server. The program has four parts: new kernel retrieval, database table load, ingest process notification, and redo of the spatial calculations, if necessary. The program is set to run every six hours, with the new ck kernel retrieved on the 6:00 am local time run.

The program connects with the NAIF FTP server and looks for new kernel files. Only the spk and ck binary, and the fk, pck, lsk, and sclk non-binary kernel files are considered. A new kernel file is defined as having a date later then the latest kernel file stored in a directory on the UA file system. If there are new kernel files, an FTP get is performed to download and store the new kernel files. The new kernel files are stored in the appropriate sub-directories and the file times are modified to reflect the file times on the FTP server.

Table 5. Kernel table description

Date Column Name	Data Column Description
Type	Type of kernel
Start_time	The start time of the range that the kernel covers
Stop_time	The end time of the range that the kernel covers
Name	The name of the kernel file
Mod_time	Time of modification

A procedure is called that loads the database kernel table (see Table 5) with the new kernel information. The fields recorded are type (kernel name), start_time, end_time, and file name. Either the brief or ckbrief SPICE functions are called to determine the start_time and the

end_time for the binary files. The notify_ingest function then generates a list of the current kernels (kernels.list). If there is a new sp kernel, the kernel list generator must be modified by hand to ensure that post-aerobreaking sp kernels are included. The data ingest process is then notified if there are new kernels to use. The notification is sent via a socket connection using an external program called cmd_ingest.tcl.

3.2.2 Redo_Spatial

Redo_Spatial reprocesses the spatial information in each GS, NS and HEND data record (see Table 3) with new kernels downloaded by the naif_ftp.tcl program. In particular redo_spatial is called each time a new ck kernel is received. Redo_spatial inputs are time range in spacecraft time. Program outputs are database updates of the spatial information in the gamma spectrum table, the neutron histogram table and the HEND table.

3.2.3 KL

The kl ("kernel list") program provides a list of SPICE kernels valid over a time range of interest in a format that can be read by both the gamma processing routines and the SPICE toolkit routines. The kl program takes as input the beginning and ending times of data that are to be processed by any of the gamma processes and outputs a list of kernels that are valid for that time frame in the format which the SPICE routines will accept. The start_time of the kernel in the database table must be less than or equal to the input start_time and the stop_time of the kernel in the database must be greater than or equal to the input stop time. It is possible to have more than one of any type of kernel for a given period, as more than one kernel may be needed to cover the time range specified. All files that fall within the given period will be included. The kl output file is called “kernel.list.”

3.3 R-SQUARED

The Mars Odyssey spacecraft and the GRS instrument suite have several different clocks that keep track of various times. The 16-bit Central Electronics Box clock is a millisecond timer that keeps track of the GRS instrument suite time. The CEB time rolls over every 65536 milliseconds. A 32-bit software rollover counter keeps track of these CEB clock rollovers. The lower 16-bits of this rollover counter are placed in every data packet sent out of the CEB, and can accommodate 65536 rollovers, for nearly 50 days (2^16 rollovers * 2^16 msec = 49.7 days). The upper 16-bits of this rollover counter is stored in the UA database. We know when the CEB was turned on and we know how long it will take for 2^32 milliseconds to go by. When a CEB time is calculated, the upper portion of the rollover is added back into the CEB time that was sent with the telemetry.

R-squared is a utility to input the rollover data (see Table 6) into the database. A spacecraft time frame is input and the rollover counter field in the meta_roll_tab database table is incremented accordingly.

Table 6. Meta_roll_tab description

Date Column Name	Data Column Description
Sc_time	The spacecraft time at the rollover in ticks.
rollover	The rollover count.

3.4 ORBIT INFORMATION

In addition to timing and spatial information, the header table (see Table 1) also contains Odyssey spacecraft orbit and GRS instrument orbit information. Two processes are necessary as the GRS orbit number is different than the Odyssey spacecraft orbit number. The spacecraft orbit number is different from the GRS orbit number in that the spacecraft orbit counts the number of times the spacecraft has orbited the planet, is incremented on the descending equator crossing, and is obtained directly from NAIF. The GRS orbit number, on the other hand, is instrument specific, is incremented on the ascending equator crossing, and is reset to 0 each time the CEB is powered down. The following two processes are used to insert the orbital information into the header table.

3.4.1 Sc_orbit

The sc_orbit process fills in the Odyssey spacecraft orbit number for each record in the header table. The sc_orbit process matches the sc_recv_time in the header table records to the sc_time in sc_orbit table to determine the spacecraft orbit number to be inserted into the header table.

3.4.2 Fill_missing_orbit

The fill_missing_orbit process is used to fill the filled_sctime and the orbit data field in the header table. This process is necessary because the instrument suite specific GRS orbit information is only carried in the GRS engineering packets. One engineering packet is received from the spacecraft for each gamma packet. The GRS orbit number from the associated engineering packet is used to fill the GRS orbit information for each gamma packet. NS and HEND instrument suite specific GRS orbital information is filled in using the gamma header values.

First the NS and HEND headers within a given spacecraft time range are found and checked for NULL values in the orbit and filled_sctime fields. If these fields are NULL, the gamma header with the closest ceb_time that is less than or equal to the given NS or HEND header ceb_time is found. The NS and HEND header orbit and filled_sctime data fields are updated with the GRS orbit and the sc_recv_time of the closest previous gamma header. In case an engineering packet is missing for a gamma header record, fill_missing_orbit will use the GRS orbit value from the engineering packet with the closest ceb_time that is less than that of the gamma record.

3.5 COMMAND INSERTION

The command insertion tools provide a mechanism by which commands sent to the spacecraft can be recorded in the UA database. The following programs are used to move the command information from the JPL Science Operations Computer (SOPC) to the UA database.

3.5.1 FIS_Tool

The File Interchange System (FIS) is an interface which resides on JPL's computer network. The FIS Tool is an automated process that allows a user to submit commands, files, and experiment kernels (E-Kernel) to the FIS for upload to the spacecraft and to the UA GRS database for record keeping. The FIS tool uses JPL supplied tools to create full spacecraft activity sequence files (SASF) and place them in the FIS. It is command line driven to allow batch processing and simple terminal access. A GUI is also available to allow browsing of input files and creation of the E-Kernel.

3.5.2 Rad_Tool

A command that has been sent from JPL's FIS to the spacecraft is considered to have been "radiated". The rad_tool fills in the sc_sent_time field in the command_sent_table of the GRS database for a file of interest. A GUI interface allows a user to select an engineering kernel notebook (EKB) file which should already contain a FIS send time. By selecting an EKB file the rad_tool can automatically retrieve all the associated commands and FIS send times. The user then enters the Lockheed Martin Aerospace (LMA) id (which is obtained from LMA , the spacecraft vendor, once the command has been radiated) for the file/files and the radiation time for the first command in the file. Mimicking the FIS_tool's GUI interface, there is a separate section for each of the three types of command files: SASF, binary command file (BCF), and binary file load (BFL). The command files are parsed by the rad_tool and the individual commands are listed next to the filename in the information box on the left side of the window. Once the user is happy with the contents of each field the 'update database' button is pushed and the information is sent to the database.

A command line interface for the rad_tool uses the same process to update the database. Instead of prompting for user input, the command line version gets all the appropriate information from a specified input file. This option is used for batch processing large numbers of radiation times. If a command is radiated multiple times using a file that already exists on the FIS, the row in the database which records the first time the file was radiated is duplicated with the lma_id and sc_sent_time changed.

3.6 FIX_PIXEL

Fix_ pixel is a process that corrects the pixel durations of each GS, NS and HEND collection interval. This process is necessary because two timing issues related to hardware and software

interaction on the spacecraft were discovered. Pixel duration is a settable parameter on the GRS instrument suite. The parameter is currently set to 19750 milliseconds, yielding 360 pixels per orbit. The true pixel duration of each collection interval may not be 19750 milliseconds. Flight software gives the pixel duration hardware count down timer a nominal value (19750 ms). The hardware timer counts this number down to 0, at which time the software gets an interrupt, and gives the nominal value to the hardware again. It was found that there is up to 2 ms latency between the hardware and the software, so that the true duration between the hardware 0 times is slightly longer than 19750. For most pixels the duration is 19751, but can be 19752. Fix_pixel can be used to correct the latency issue because the CEB time is recorded for each pixel at the hardware 0 time.

In addition to the latency between the hardware and software there is another less common issue with the pixel duration flight software. Software interrupts used on a variety of counters have different priorities. If an interrupt with a priority higher than the pixel duration counter is active at the time when the pixel duration counter should execute, then the pixel duration interrupt is delayed until the higher priority interrupt is finished. Once the higher priority interrupt is finished, the software sends the nominal pixel duration again. The length of the delay between when the pixel duration interrupt should have been sent and when it was actually sent is unknown, but can be calculated by subtracting the CEB time of the previous pixel and the CEB time of the current pixel. The CEB times can be used because the CEB time is the time of the hardware 0.

Fix_pixel calculates the duration of a pixel by subtracting the CEB time of the current pixel from the CEB time of the previous pixel. The resulting CEB time is the revised pixel duration. The revised pixel duration is then used to calculate a revised sc_ev_time in ticks (see Equations 1-3), which is used to re-run redo_spatial in order to correct the spatial information that is associated with the time at the center of the pixel (see Table 3).

The following equations are used to calculate the pixel duration:

$$pixel\_duration\_rev = ceb\_time - ceb\_time\_last \quad \textbf{eq. 4}$$

$$sc\_ev\_time = \left(\left(ceb\_time\_last + 0.5 * pixel\_duration\_rev\right)/1000 + delta\_time\right) * 256 \quad \textbf{eq. 5}$$

4.0 LEVEL 1-B PROCESSING

During the Mapping phase of the Mars Odyssey Mission conditions on the spacecraft and in the GRS instrument suite change over time. During a single gamma collection interval to few gamma counts are received to produce a spectrum with sufficient statistics for scientific interpretation. Due to slight changes in temperature, voltages, and currents gamma spectra collected at different times have slightly different energy scales. Many gamma collection intervals must be summed in order to obtain a spectrum with the appropriate statistics for scientific analysis. In order to sum gamma spectra, they must be calibrated to a common energy

scale. The following sections describe the gamma specific data processing necessary to calibrate each gamma spectra to a common energy scale and to produce a summed gamma spectra.

4.1 ENG_SMOOTH

Engineering data used for corrections are smoothed to remove high frequency noise from the data when viewed as a time series. The engineering smoother process uses a Gaussian-weighted smoothing algorithm (Marchand & Marmet, 1983) to remove the noise from the engineering data, and returns the smoothed value to the smoothed_val field in the appropriate engineering tables (see Table 2). The smoothing for a single reading at point *k* (corresponding to time *T*) requires readings from *k*-*N* to *k*+*N* for the calculation, where *N* is the desired degree of smoothing for the type of engineering data. This means that the smoothing calculations for a given collection interval's engineering data can not be completed until *k*+*N* points are received.

The binomial smoothing algorithm developed by Marchand and Marmet is a computationally efficient form of least-squares polynomial (LSP) smoothing that avoids the high frequency "leaks" and phase shifts generally associated with LSP smoothing algorithms. It generates a smoothed value for a data point *k* based on the values of the data between points *k*-*N* and *k*+*N* using only additions and divisions by 2.

In short, to calculate the *N* point smoothed value *Y* from a dataset *X* at point *k*

Let $s = k - N$ and $e = k + N$ **eq. 6**

Also let $Y_0 = X_s, Y_1 = X_{s+1}, \cdots, Y_{2N} = X_e$ **eq. 7**

repeat the following *n* times:

$$Z_0 = \frac{(Y_0 + Y_1)}{2}, Z_1 = \frac{(Y_1 + Y_2)}{2}, \cdots, Z_{2N-1} = \frac{(Y_{2N-1} + Y_{2N})}{2}$$
$$Y_1 = \frac{(Z_0 + Z_1)}{2}, Y_2 = \frac{(Z_1 + Z_2)}{2}, \cdots, Y_{2N-1} = \frac{(Z_{2N-2} + Z_{2N-1})}{2} \quad \textbf{eq. 8}$$

The value at Y_N after *n* iterations is the *N* point smoothed value of X_k.

In practice, there are several complications in the smoothing algorithm. Some of the engineering parameter values are calculated based on the raw values of two engineering parameters. Many of the gamma engineering channels require a parameter called gamma_mux_offset in their raw to engineering conversion. Some of the current sensors require a reference voltage (ps_cur_ref or pc_cur_ref) in their calculations. See Appendix A. – Engineering Conversions for the raw value to engineering conversion algorithms. The engineering smoothing program accounts for the smoothing in both engineering parameters by first smoothing the gamma_mux_offset and the current reference values. The engineering values of the channels that require either the gamma_mux_offset or one of the current references are then recalculated using the smoothed

value of the reference parameters and the raw (DN) value of the engineering parameter of interest. The newly revised engineering values are then smoothed in the normal way.

Another complication in the engineering smoothing is data gaps. The smoother works on *k-N* to *k+N* temporally contiguous engineering records. If data gaps less than the average sc_time duration of *2N+1* records are present in the engineering time series the smoother will use a "bounce" method to continue the smoothing. The "bounce" method uses *X[k+i]* in place of the missing *X[k-i]* and *X[k-i]* in place of the missing *X[k+i]*. If an engineering data gap is longer than the average sc_time of *2N+1* records the last point to be smoothed is the last point *k* that has an actual engineering reading. The smoother continues (using the bounce method) after the data gap with the first point *k* that has an actual engineering reading.

There have been occasions over the course of the mission when engineering readings have been recorded that do not fit within the expected dynamic range of the engineering parameter. These outlier records are rare, and are most likely due to a failure of the electronics to reset quickly enough. Outlier points are excluded from the smoothing as they do not reflect the actual condition of the engineering channel. Outliers are identified by comparing the point *k* engineering value to the standard deviation (σ) of the *2N* points surrounding it. If the point *k* value is within *m*σ where *m* is the outlier sigma multiplier value (see Table 7) of the value then point k is included in the smoothing. If point k is outside of the *m*σ value then the point is excluded from the smoothing.

Table 7. Engineering smoothing constant values.

Engineering Parameter Name	*N* Value for smoothing	*m*, Outlier sigma multiplier value
GAMMA_MUX_OFFSET	30	16
CEB_PC_CURR_REF	30	10
CEB_PS_CURR_REF	30	10
GAMMA_DAC_0	30	10
GAMMA_DAC_1	30	10
GAMMA_DAC_2	30	10
GAMMA_DAC_3	30	10
GAMMA_DAC_4	30	10
GAMMA_DAC_5	30	10
GAMMA_DAC_6	30	10
GAMMA_DAC_7	30	10
GAMMA_ELEC_FAST	30	15
GAMMA_ELEC_SLOW	30	15
GAMMA_GPA_TEMP	30	15
GAMMA_HVBS_ENABLE_MNTR	30	10
GAMMA_HVBS_MNTR	30	10
GAMMA_HVBS_TEMP	30	10
GAMMA_MINUS_12V_HVBS_RAIL_CRNT	100	50

Engineering Parameter Name	*N* Value for smoothing	*m*, Outlier sigma multiplier value
GAMMA_MINUS_12V_PULSE_AMP_CRNT	100	50
GAMMA_MINUS_12V_RAIL_CRNT	100	50
GAMMA_MINUS_12V_RAIL_VOLT	30	15
GAMMA_PLUS_12V_HVBS_RAIL_CRNT	100	50
GAMMA_PLUS_12V_PULSE_AMP_CRNT	100	50
GAMMA_PLUS_12V_RAIL_CRNT	100	50
GAMMA_PLUS_12V_RAIL_VOLT	30	15
GAMMA_PLUS_5V_DIG_RAIL_CRNT	100	50
GAMMA_PLUS_5V_DIG_RAIL_VOLT	30	10
GAMMA_REF_VOLT_166	30	10
GAMMA_REF_VOLT_333	30	10
GAMMA_REF_VOLT_500	30	10
GAMMA_SPARE_1	30	10
GAMMA_SPARE_2	30	10
GAMMA_VDD_DIRECT_MNTR	30	15
GAMMA_VREF_TEMP	30	15
CEB_AD_TEMP	30	10
CEB_AGND	30	10
CEB_AGND_SPARE1	30	10
CEB_AGND_SPARE2	30	10
CEB_AGND_SPARE3	30	10
CEB_ALT_ACT_CURR	30	10
CEB_CPU_PLUS_5	30	15
CEB_CPU_PLUS_5_CURR	30	15
CEB_CPU_TEMP	30	15
CEB_HTR_CNTRL_TEMP	30	10
CEB_IS_TEMP_A	30	10
CEB_IS_TEMP_B	30	10
CEB_MAIN_ACTUATOR_CURR	0	0
CEB_MINUS_12V_CEB_AN	30	10
CEB_MNT_RNG_TEMP_A	30	10
CEB_MNT_RNG_TEMP_B	30	10
CEB_OS_TEMP_A	30	10
CEB_OS_TEMP_B	30	10
CEB_PLUS_12V_CEB_AN	30	10
CEB_PLUS_28_CURR	0	0
CEB_PLUS_5_CRYO	30	10
CEB_PS1_TEMP	30	10
CEB_PS2_TEMP	30	10

Engineering Parameter Name	*N* Value for smoothing	*m*, Outlier sigma multiplier value
CEB_SPARE_CURR_SENSE_2	0	0
CEB_SPARE_CURR_SENSE_3	0	0
HVPS_MNTR_1	30	15
HVPS_MNTR_2	30	15
PLUS_5V_ANLG	30	15
PLUS_5V_CRNT_ANLG	30	10
PLUS_5V_CRNT_DIG	0	0
PREAMP_TEMP	30	10
MINUS_5V_ANLG	30	15
MINUS_5V_CRNT_ANLG	30	10

4.2 B170K_CALC

GRS data exhibit gain, offset and Integral Non-linearity (INL, the deviation of the energy to channel transfer function from a straight line) shape drifts with temperature. These drifts are due to different subsystems located in different parts of the instrument. One of the temperatures of interest is the temperature of the 170K board. The 170K board, or front end electronics board, provides first stage signal amplification and transfers the signal to the gamma pulse amplifier (GPA). The 170k board temperature is not directly monitored, but can be modeled based on an outer stage temperature.

The B170K_Calc process models the 170k board temperature (°C) based on the outer stage B temperature sensor (CEB_OS_TEMP_B). For each outer stage engineering temperature in a time range of interest in the CEB_OS_TEMP_B database table a 170k temperature is modeled and stored in the B_170K_TEMP database table. An RC (resistance capacitance) thermal model with the following standard equation is used to calculate the 170K card temperature:

$$170K_T = 170K_{T-\Delta T} + \frac{OS\_TEMP_T - 170K_{T-\Delta T}}{R\frac{60}{\Delta T}} \quad \textbf{eq. 9}$$

where R(resitivity coefficient) = 32 for CEB_OS_TEMP_B and T is time in seconds.

The data collection times and raw engineering values from the CEB_OS_TEMP_B table are copied into the B_170K_TEMP table. The engineering value from the CEB_OS_TEMP_B table is used to calculate the 170k temperature, which is inserted into both the engineering value and the smoothed value fields in the B_170K_TEMP table.

4.3 DIGITAL_HK

Digital housekeeping (digital_hk) values record the settings of a number of registers in the GS analog pulse processing system (APPS) board. The APPS board converts the analog signal from the front end electronics into a digital signal. Digital housekeeping values are received in the raw instrument telemetry and are inserted into the GRS database by the ingest process. The digital housekeeping readings are stored in the GRS digital hk table. When the values are stored in the GRS digital hk table they consist of sixteen ten-bit values. The sixteen values contain the APPS register readings without word aligning each value. The digital_hk program word aligns the APPS register readings and stores the deciphered readings in fourteen different tables, similar in structure to the engineering tables (see Table 2 for the engineering table format). The APPS tables contain fields for both an engineering value and a smoothed value. The smoothed value field is not used.

Table 8. Digital housekeeping database table names.

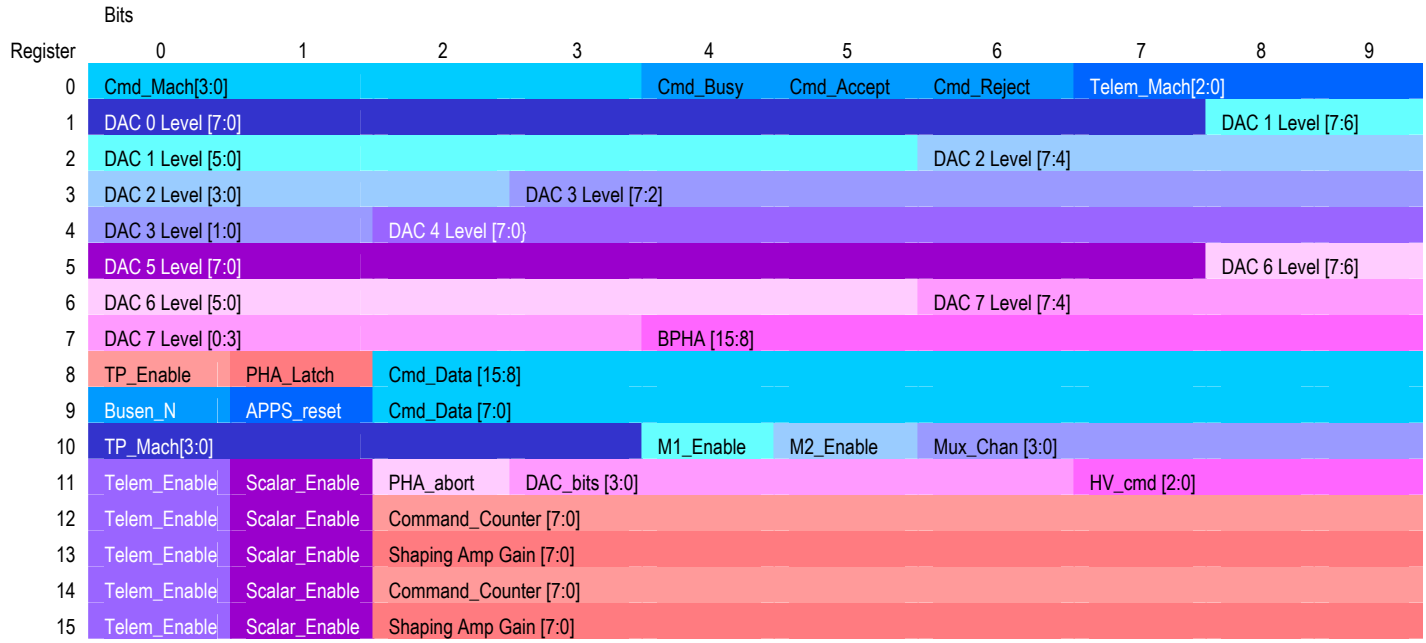
Database Table Name	Digital housekeeping parameter name and description
APPS_HK_COMMAND_COUNTER	Command counter
APPS_HK_COMMAND_DATA	Command data
APPS_HK_DAC0	Digital to analog converter 0 – lower level discriminator
APPS_HK_DAC1	Digital to analog converter 1 – L1 scalar
APPS_HK_DAC2	Digital to analog converter 2 – L2 scalar
APPS_HK_DAC3	Digital to analog converter 3 – L3 scalar
APPS_HK_DAC4	Digital to analog converter 4 – Upper level discriminator
APPS_HK_DAC5	Digital to analog converter 5 – Shaping amp gain
APPS_HK_DAC6	Digital to analog converter 6- open
APPS_HK_DAC7	Digital to analog converter 7 – High voltage
APPS_HK_ENABLE	The enable value is the TELEM_ENABLE and SCALAR_ENABLE flags, where SCALAR_ENABLE is lowest [0] bit of the APPS_HK_ENABLE value.
APPS_HK_SHAPING_AMP_GAIN	Shaping amp gain setting value.
APPS_HK_STATUS	CMD_BUSY, CMD_ACCEPT, and CMD_REJECT flags in that order
APPS_HK_TP_ENABLE	Test Pulser enable flag.

The digital_hk program takes as input the beginning and ending dates for which digital housekeeping values will be translated into the APPS tables in the database. The APPS readings are word aligned as described in Figure 3 and the resulting fourteen values of interest are stored in individual database APPS tables (see Table 8). Note that the Telem_enable, Scalar_enable, Command Counter, and Shaping amp gain registers are repeated.

In some cases the APPS readings stored in the GRS digital hk table may only contain a partial set of APPS reading or a set of digital housekeeping readings may be split between two consecutive

database records (rows). Upon examining each row in the digital hk table the digital_housekeeping program checks the header table for the next consecutive record. If the next consecutive header record is a dmp record (digital housekeeping dump) then the information in that record is combined with the previous digital hk record to yield a more complete data set.

Figure 3. Digital housekeeping word alignment.



4.4 INTERP_ENG

A number of different engineering, counter, spatial and timing values associated with each gamma spectrum are stored in the gamma spectrum database table. The interpolate engineering program (Interp_eng) is used to insert each gamma spectrum's associated engineering values into the gamma spectrum table. The engineering values (Table 9) are then used in the correction process.

Engineering readings are collected by the spacecraft for each gamma spectrum collected. The engineering readings are received from the spacecraft telemetry as discreet packages separate from the gamma packages. Both engineering and gamma packages are identified by time (see Table 1 Header table description.) The interp_eng program performs a linear interpolation of the smoothed engineering values collected just before and just after the gamma spectrum of interest. The interpolated engineering value is then inserted into the appropriate field in the corresponding gamma record. If a smoothed engineering value does not exist within one hour of the gamma spectrum then an interpolation is not made, and there is no update for that field. Gamma spectrum records that have NULL engineering values are marked in the Bad_flag (see Section 4.6) process as having an engineering data gap.

Table 9. Engineering values stored in the gamma spectrum table

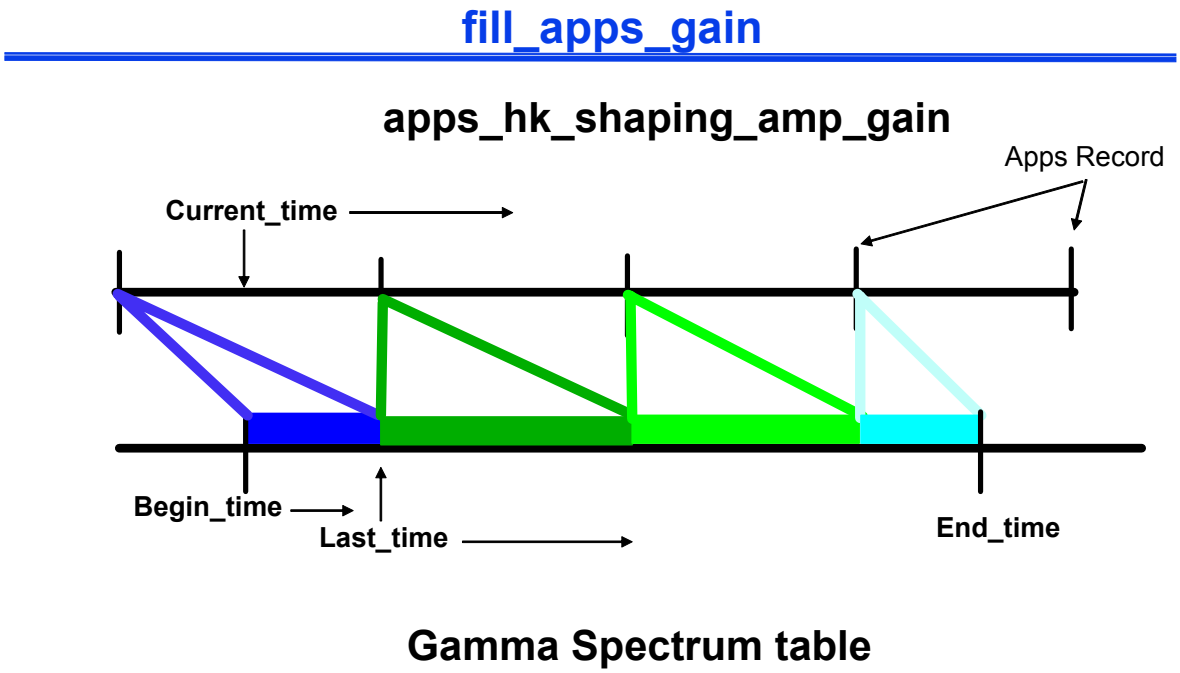
Engineering column Name	Description
hvbs_monitor	High voltage bias supply monitor value from engineering - *Converted to engineering units, smoothed, and interpolated to the center of the pixel)
is_temp_a	Inner stage temperature monitor a from engineering*
is_temp_b	Inner stage temperature monitor b from engineering*
os_temp_a	Outer stage temperature monitor a from engineering*
os_temp_b	Outer stage temperature monitor b from engineering*
gamma_vref_temp	Gamma voltage reference temperature from engineering*
gpa_temp	Gamma pulse analyzer temperature from engineering*
b170k_temp	Modeled 170K board temperature from 170k model

4.5 FILL_APPS_GAIN

The last parameter to be specified for each gamma spectrum before the correction process (see Section 4.7) is run is the analog pulse processing system (APPS) gain. The APPS gain is the setting of the variable gain stage in the APPS board (shaping amp gain) that is set to a value closest to our desired gain. The fill_apps_gain process uses the raw value from the apps_hk_shaping_amp_gain table and saves it into the gamma spectrum table, apps_gain_dhk field. The raw value in the apps_hk_shaping_amp_gain table was filled in by the digital hk program.

A begin time and end time are entered into the fill_apps_gain program. The program searches for the first occurrence of an apps_hk_shaping_amp_gain table record from the begin_time, the value is pulled out of the table, and inserted into all the gamma spectrum table records after the time of the shaping amp gain reading and before the time of the next apps_hk_shaping_amp_gain record. The new APPS gain value is then inserted for all the records after the time of the change. The last three spectra before a change in APPS gain are flagged as bad to ensure that the change in gain is noted. See Figure 4 for a diagram of the fill_apps_gain process.

Figure 4. Fill Apps Gain process.



4.6 BAD_FLAG

The Bad_Flag program is designed to examine the data in the gamma spectrum table to determine if the spectra and associated data fall within acceptable limits for a number of parameters. This automated check insures that only appropriate data is used in the subsequent processing. The result of the processing is a code number entered in the q_flag field of the gamma spectrum table. The q_flag is a number representing the bits that have been set for each of the bad codes. Q_flags are copied to the bad_code field in the gamma spectrum table only after inspection and verification by one of the GRS team data analysts. Table 10 is a list of the bad codes, the associated bits and decimal representations. All spectra are run through the bad_flag program and all spectra that meet the minimum requirements for the parameters of interest have a bad code of 0.

The bad_flag program works by reading in the time and the associated smoothed engineering values for the b170k_temp, is_temp_b, gamma_vref_temp, and gpa_temp for each collection interval over a period of interest. The program reads in a variable amount of data prior to the start of the period of interest in order to have enough data for the checks that require a time element to be made.

The first check of the data determines if there are data gaps in any of the data from the four temperature values in the time period of interest. If a data gap of greater than 1 hour exists for any of the four temperature readings associated with a gamma spectrum, than all gamma spectra within the data gap time frame are marked as "bad". Data gaps are identified by a q_flag/bad_code of decimal value 16384.

Table 10. Bad flag names, codes and decimal numbers.

Bad Flag Name	Code	Decimal Number	Program that sets flag
BAD_HV_STATE	0x00000001	1	Auto_Bad
BAD_PUL_COMB	0x00000002	2	Auto_Bad
BAD_GAIN_SHFT	0x00000004	4	Fill_Apps
BAD_TEMP_SHFT	0x00000008	8	Auto_Bad
BAD_GHOST_PK	0x00000010	16	Auto_Bad
BAD_SOLAR_PARTICLE_EVENT	0x00000020	32	Manual
BAD_COMP_SET	0x00000040	64	N/A
BAD_OTHER	0x00000080	128	Manual
BAD_NULL	0x00000100	256	Sum
BAD_HV	0x00000200	512	N/A
BAD_GAIN	0x00000400	1024	Sum
BAD_OFFSET	0x00000800	2048	Sum
BAD_NUM_CONT	0x00001000	4096	Sum
BAD_CONT_PEAK	0x00002000	8192	Sum
BAD_DATA_GAP	0x00004000	16384	Auto_Bad
BAD_PULSER_SHIFT	0x00008000	32768	Auto_Bad
BAD_SOLAR_FLARE	0x00010000	65536	Manual
BAD_RESOLUTION	0x00020000	131072	Manual
BAD_UHF_TEST	0x00040000	262144	Manual
BAD_HV_TRANSITION	0x00080000	524288	Manual
BAD_OFF_POINT	0x00100000	1048576	Manual
BAD_VECTOR	0x00200000	2097152	Vector_check

Once data gaps are identified, the remaining data are checked for periods of rapid temperature change. The data for the b170k_temp, is_temp_b, gamma_vref_temp, and the gpa_temp are compared to the temperatures in the previous collection interval. If there is a temperature difference between the current record and the previous record of greater than 1 degree for the b170K and is_temp_a, and 2-degrees for the vref and gpa temperatures then the datum is noted. The noted datum is averaged with the previous two records to find the start of the out-of-range temperatures by comparing the temperature in the record of interest with the average. If the temperature is greater than 5% different from the out-of-range value then the record is flagged with the BAD_TEMP_SHFT flag. The end of the out-of-range temperatures is found the same way, except that the next two temperature records are used instead of the previous two records for the average.

The next check performed is to check if pulser data is being combined with gamma data. Pulsers are a set of three internal reference signals that can be set in several configurations. Usually the pulsers are all turned on, but can be configured for all off, one on, or two on. In the normal course of operations the pulser signal is distinguished from the gamma spectrum. However if the

pulsers are set to the "combined" setting then the pulser signals are combined with the gamma spectrum. The combined pulser and gamma spectrum should not be processed in the same way as a regular gamma spectrum; therefore during those times that the pulsers are set to "combined" the BAD_PUL_COMB flag is set.

Ghost peaks are caused by an overload in the analog pulse processing system (APPS) electronics if the shaping_amp_gain is set at a level that shifts the upper level discriminator energy below that of the pulser energy. The pulsers are used as an indicator of this problem. If pulsers are enabled and if the channel of the pulser is above the upper level discriminator setting then the record is flagged with the BAD_GHOST_PK flag.

The final check run in the bad_flag program is to check if the gamma high voltage is on. The check also accounts for the high voltage being ramped and a period of time after the ramp to make sure the voltage is stable. If the high voltage is not on, is being ramped, or has not been stable long enough, the BAD_HV_STATE flag is set.

If the bad_flag program identifies and marks any spectrum as "bad", the program sends an e-mail to a data analyst enumerating the number of and types of bad codes that have been suggested by the setting of the q_flag. The data analyst then looks at each spectrum that has been marked with the q_flag to ensure that the spectrum is truly out of tolerance. Once the suggested bad_codes have been analyzed, the data analyst copies the verified q_flag field into the bad_code field.

Manual bad_codes are then inserted into the bad_code field by the data analyst if necessary. The BAD_SOLAR_PARTICLE_EVENT bad_code is inserted into all gamma records collected during solar particle events, the start and end of which are determined by direct examination of the gamma spectra. The BAD_OTHER bad code is rarely used, and is reserved for individual spectra that have some error that can not be readily explained or reproduced. The BAD_SOLAR_FLARE bad_code is used to indicate all spectra collected during a solar flare. The BAD_RESOLUTION bad_code is used to indicate any spectra collected during times when the spectral resolution of the GS detector was degraded due to radiation damage. The BAD_UHF_TEST bad_code is used to indicate any spectra collected during testing of the UHF antenna aboard the spacecraft. The BAD_HV_TRANSITION bad_code is a catch-all bad flag that is used to indicate that there was a significant event on the spacecraft that precluded the collection of "good" gamma data. The BAD_HV_TRANSITION bad_code is inserted into all records after the time of the spacecraft event until the return to nominal data collection.

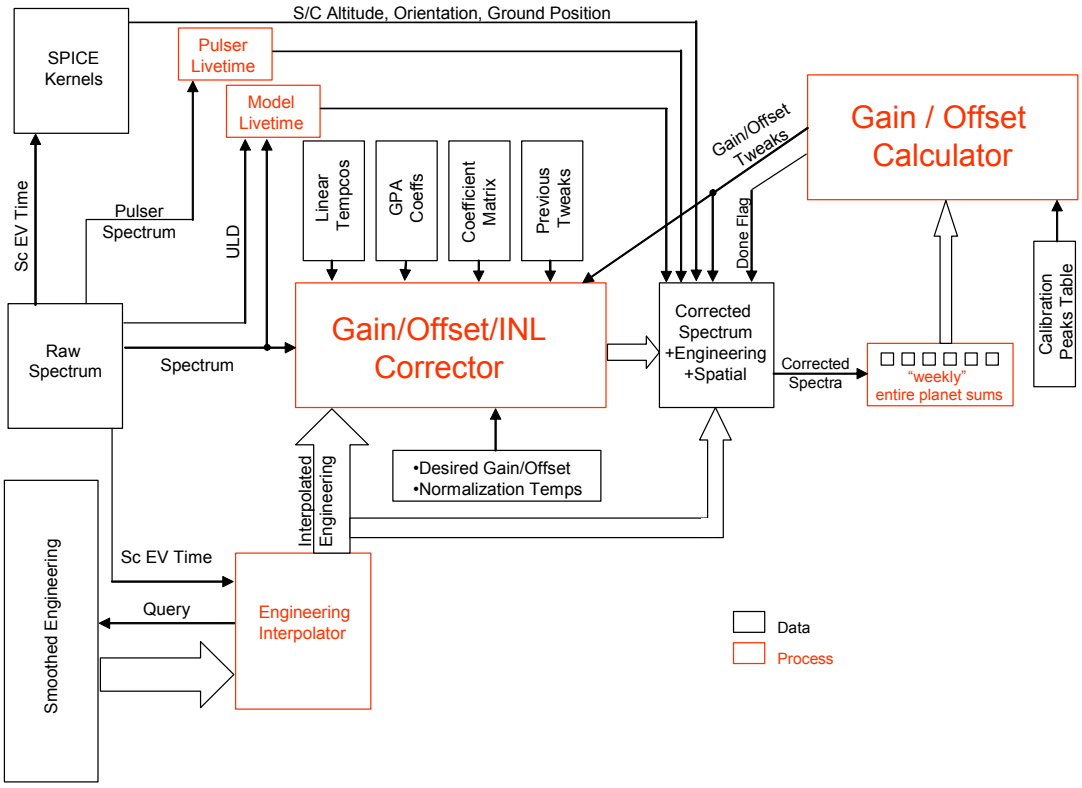
4.7 CORRECTION

The correction portion of the gamma data processing includes gain, offset, and Integral Non-linearity (INL, the deviation of the energy to channel transfer function from a straight line) corrections on the raw gamma spectra. A raw gamma spectrum consists of counts binned by the gamma subsystem into 16384 energy dependent channels. The spectra exhibit gain, offset and INL shape drifts with temperature. These drifts are due to temperature variations in several of the electronic components of the gamma subsystems. The four major components whose

temperatures contribute to these errors are the detector, the 170K board, the gamma preamplifier, and the analog pulse processing system (APPS) board. Each of these components were individually characterized prior to launch and are documented in GRS Calibration Report.

In order to correct gamma spectra to a common energy scale a spectrum shifting process (see Section 4.7.1.1) is used. The temperatures of the subsystems, the gain setting of the APPS board, and a factor based on previous experience are used to shift each individual spectrum to a desired gain (0.625keV/channel) and offset (0). The spectrum shifting process also corrects for the non-linearities in the APPS electronics producing corrected spectra which are suitable for comparison to each other. The spectrum shifting algorithm makes the gain, offset and INL corrections in one pass through the data in order to limit the amount of noise introduced into the corrected spectra.

Figure 5. Data flow to the correction process.



4.7.1 Gain/Offset/INL Corrector

The Gain/Offset/INL correction process begins with a query of the gamma spectrum table for records where either a) the corrected spectrum field is NULL and the required engineering fields are not NULL, or b) the corrected spectrum field is not NULL, and the corrected flag is not yet

set. Case (a) will cause the initial correction, and case (b) will cause the final correction. In both cases, the spectrum is shifted just one time including terms for gain, offset and INL. The input to the spectrum shifting algorithm is a term called "*IN_BIN*" and the output is a term called "*OUT_BIN*".

4.7.1.1 Spectrum Shifting Algorithm

The spectrum shifting algorithm calculates the common energy scale used for each spectrum, and re-bins the counts in the raw spectrum to the proper energy bin in the corrected spectrum. Each channel in the spectrum does not represent a single, unique energy, but rather a range of energies such that any gamma ray where channel_low_limit ≤ gamma_energy < channel_high_limit will be counted in this channel. During calibration 32 precision pulser peaks evenly distributed across the spectrum were used to determine the high and low channel limits of each channel as a function of channel number. A polynomial was then fit to the linear channel versus the measured channel to produce a function that gives the correct channel limits for each channel. Since this relationship is also a function of temperature, a family of polynomials was produced over the operational temperature range of the CEB. A second set of polynomials was fit to the coefficients vs. temperature to produce the INL coefficient matrix ($matrix_{mn}$, see INL Coefficient Matrix Section 7.2).

To determine the center energy of a given channel at a given temperature ***T*** (vref_temperature °C) a set of coefficients **C** are calculated where:

$$C_m = \sum_{n=0}^{order1} \left(matrix_{m,n} \times \left(\frac{T}{100} \right)^n \right) \quad \textbf{eq. 10}$$

Note that ***T*** is divided by 100 to keep the coefficients reasonably large.

The C_m coefficients are used to calculate the lower channel limit (in linearized space) IN_BIN_j for each channel $\boldsymbol{0 \le j \le 16382}$. Channel 16383 is reserved for the overflow counts that will be described later in this section. The lower channel limit calculation is done in two steps to cut the number of polynomial evaluations in half. First the center channel is calculated for each channel from –1 to 16381.

$$IN\_BIN_j = \sum_{m=0}^{order2} \left(C_m \times \left(\frac{j-1}{1000} \right)^m \right) \quad \textbf{eq. 11}$$

Second, the counts in a given channel are assumed to be evenly distributed between channel-0.5 and channel+0.5; the channel is calculated half way between these end points.

Offset errors are primarily due to the Gamma Pulse Amplifier (GPA). The GPA has a very small gain variation (−1.8ppm/°C) but a considerable offset variation with temperature. This offset was characterized both in the lab in pre-launch calibration and again during cruise, and is exponential

over the GPA operating range. From the fit of calibration and flight data a set of coefficients ***G*** is used to calculate this offset as a function of GPA temperature (***T***).

$$Offset_{GPA} = G_0 + G_1 e^{-G_2 T} \quad \textbf{eq. 12}$$

Since the GPA offset occurs before the variable gain stage in the APPS, these coefficients will change with the shaping amp gain setting[2]. The pattern of this change is not yet well understood, but coefficients have been calculated for shaping amp gain 46. Currently, there is a table of the 3 offset coefficients at gain 46 (see Section 7.1 for the table). If it becomes necessary to change the shaping amp gain setting the GPA temperature will be incremented slowly to get a dataset from which a new set of coefficients can be calculated.

The second offset term is a back-calculated value that ensures that the offset correction yields the desired offset (usually 0). The offset is stored in a table, and is associated with the shaping amp gain value (see Section 7.3). Currently there is only one valid value in the gain-offset table, the value associated with shaping amp gain 46.

The offset terms are summed to yield the total offset input into the *IN_BIN* portion of the spectrum shifting algorithm.

$$inputOffset = Offset_{GPA} + Offset_{TABLE} \quad \textbf{eq. 13}$$

If there are any other offsets, they would be included in equation 13.

The results of equation 11 and equation 13 are combined to give:

$$IN\_BIN_j = \frac{\left(IN\_BIN_j + IN\_BIN_{j+1}\right)}{2} + inputOffset \quad \textbf{eq. 14}$$

which is the input spectrum channel limits term for the spectrum shifting algorithm.

Once the input spectrum properties are calculated the output spectrum properties must be calculated. Gain shifts are calculated for the output spectrum channel limits. The adjustments for the gain shifts due to the detector (is_temp_b) and 170K temperatures are linear with temperature and energy. To calculate the gain shift a normalization temperature ($T_{\text{norm,}}$ Section 7.1) is chosen to which each component is normalized. Since the temperature T is known for each component and from the instrument calibration (see GRS Calibration Document for the calibration details) the temperature coefficient C_t is known, a ratio for each of the components can be calculated.

[2] The gain of the system is adjustable from ≈1keV/channel to ≈0.25keV/channel.

$$ratio = 1 + ((T - Tnorm) \times C_t) \quad \textbf{eq. 15}$$

The overall gain ratio then becomes:

$$GainRatio_T = ratio_{DETECTOR} \times ratio_{170K} \times ratio_{GPA} \quad \textbf{eq. 16}$$

In order to properly shift the gain a function of all the corrections must be known. A table is used to store the corrected gain in channels/keV (not keV/channel) for the chosen normalization temperatures of the 170K and the detector. A final gain ratio is then calculated.

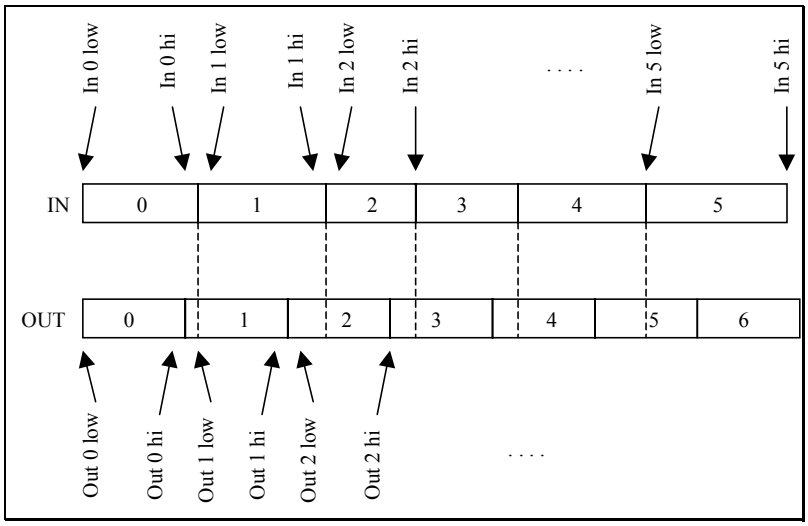
$$FinalGainRatio = desiredGain(keV / channel) \times gain\_table[APPS\_gain] \times GainRatio_T \quad \textbf{eq.17}$$

From the gain ratio the lower channel limit of each channel in the output spectrum for the desired gain and offset can be calculated.

$$OUT\_BIN_j = (j - 0.5) \times FinalGainRatio + desiredOffset \quad \textbf{eq. 18}$$

Figure 6 illustrates the process of mapping the counts from the nonlinear channel space of the instrument to a linear channel space.

Figure 6. Input channels larger than output channels.



As you can see, the counts the instrument puts in channel 0 should really be divided between channel 0 and channel 1. Knowing the high and low limits of the input and output channel, the counts can be divided as follows:

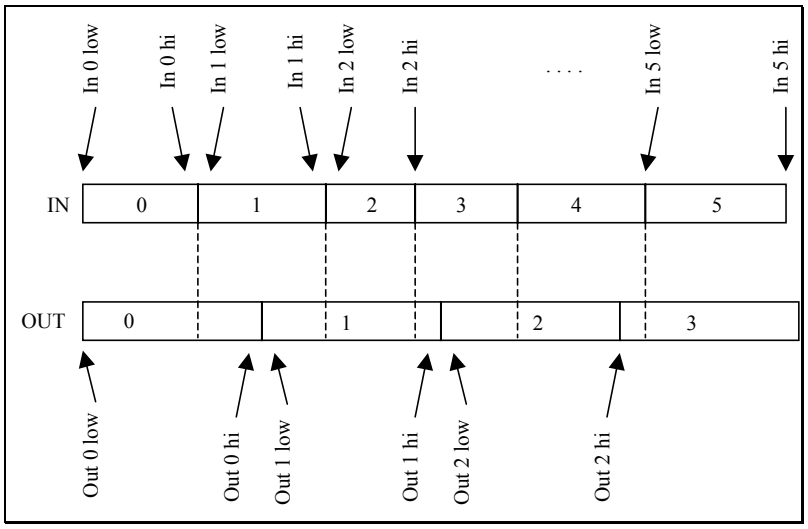
$$out_0 = in_0 \times \frac{out0hi - in0low}{in0hi - in0low} \rightarrow out_1 = in_0 \times \frac{in0hi - out1low}{in0hi - in0low} \quad \textbf{eq. 19}$$

Then for input channel 1 the process is:

$$out_1 = out_1 + in_1 \times \frac{out1hi - in1low}{in1hi - in1low} \rightarrow out_2 = in_1 \times \frac{in1hi - out2low}{in1hi - in1low} \quad \textbf{eq. 20}$$

In Figure 6 the output bins are generally smaller than the input bins. Figure 7 illustrates the case where the output channels are generally wider than the input channels.

Figure 7 Input channels are smaller than output channels.



In this case the process becomes:

$$out_0 = in_0 \rightarrow out_0 = out_0 + in_1 \times \frac{out0hi - in1low}{in1hi - in1low} \quad \textbf{eq. 21}$$

and for channel 1:

$$out_1 = in_1 \times \frac{in1hi - out1low}{in1hi - in1low} \rightarrow out_1 = out_1 + in_2 \rightarrow out_1 = out_1 + in_3 \times \frac{out1hi - in3low}{in3hi - in3low} \quad \textbf{eq. 22}$$

The spectrum shifting process is different for different alignments of the input and output channels. These can be generalized into two cases distinguished by the relative positions of the high limits of the input and output channels. These two cases are illustrated in Figure 8 and Figure 9. Since the instrument gain is set close to the desired gain, the correction will oscillate between these two cases as the channels are inspected and as the temperatures of the various subsystems change.

Figure 8. Algorithm for the case out_hi ≤ in_hi.

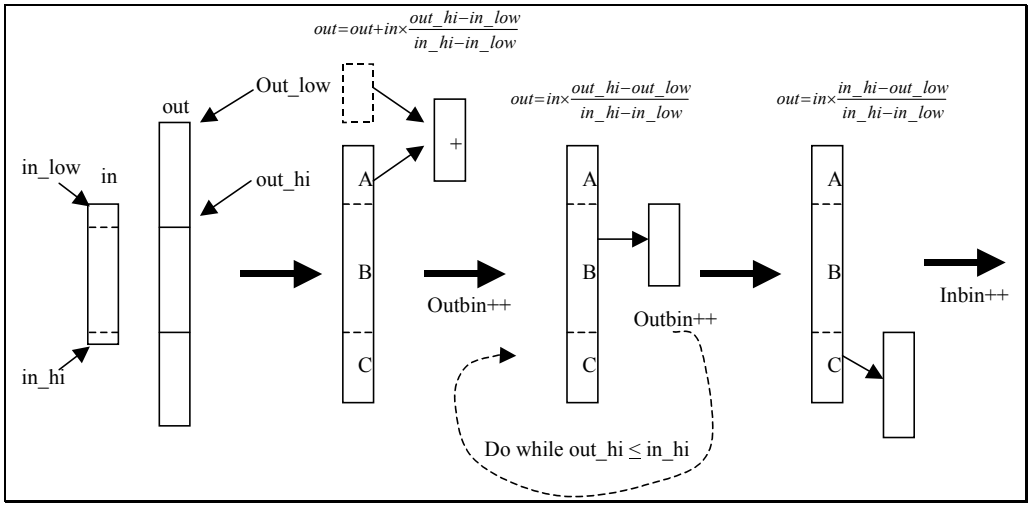
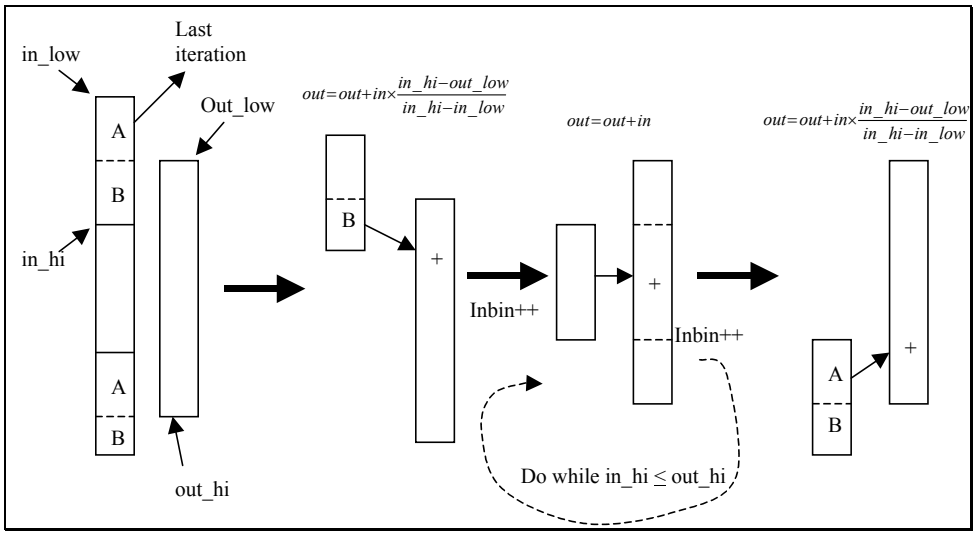


Figure 9. Algorithm for the case out_hi > in_hi. Note: There is a mistake in the labeling of the last (far right) equation; the numerator should read out_hi-in_low.



The spectrum shifting algorithm has several special features. The counts that would be shifted to channels <= 0 are all put in channel 0. Counts that would be shifted into channel 16383 or above are put in channel 16382. Channel 16383 is not shifted since it holds the overflow counts (sum of all counts with energies between channel 16383 and the Upper Level Discriminator setting, which is a record of all counts above the energy level of the ULD setting (usually > 10MeV).

4.7.2 Gain/Offset Calculation

The internal and external environmental conditions the GS experiences change over time. The correction algorithm may or may not do an adequate job of correcting these factors to yield corrected spectra with the desired gain and offset values. A second correction process (re-correction) has been implemented to ensure that the desired gain and offset are met and to force the spectra to be corrected to a common gain scale. The assumptions in the gain_offset calculation are that the INL corrections are fully accounted for in the Gain_Offset_INL correction step (see Section 4.7.1) and that the shape of the summed spectra peaks have not changed since the implementation of the re-correction algorithm.

Table 11. Peaks used for gain offset re-correction.

Peak Number	Channel Number	Energy	Source
0	817	511	Annihilation
1	1574	983.52	Ti48* Sc48 (V48)
2	2040	1274.53	Na22 Ne22*
3	2338	1460.82	K40
4	2845	1778.97	Si28* Al28
5	3558	2223.3	H1ng
6	4406	2754.03	Na24
7	9807	6128.63	O16*
8	12208	7631.2	Fe56ng

The first step in the re-correction process is the creation of summed spectra that have been corrected to the desired gain (0.625) and offset (0). The sums are made up of corrected spectra from the entire planet over a "weekly" temporal interval. The "weekly" interval is typically about six Earth days, but is calculated to be an even division of the Mars Season (15° L_S, see Time Conversion Tables), as the re-correction process is run on a Mars Seasonal basis.

Once sums are created for the 4 (or 5 in the longest Mars Seasons) "weekly" intervals, they are analyzed with a peak fitting algorithm. The peak fitting algorithm uses a Gaussian with an exponential tail fitting method to calculate the centroids of nine strong peaks (see Table 11) distributed across the energy range of the gamma spectrum. A line is fit to the centroids yielding an equation in the form of:

$$y = mx + b$$

where *m* is the gain in channels per KeV and *b* is the offset in channels. The gain and offset numbers are then manipulated to be in the proper form for insertion back into the correction algorithm as per Section 4.7.1Gain/Offset/INL Corrector. The gain factor given to the correction process is the result of 0.625 divided by *m*. The offset factor given to the correction process is **b** divided by *m*. The gain and offset factors are passed to the correction process which is re-run to re-correct the individual spectra to force a gain of 0.625 and an offset of 0.

4.8 VECTOR CHECK

Once each gamma spectrum has been corrected, the vector_check program is run to compare the calculated latitude and longitude associated with each spectrum to the instrument boresite vector and the spacecraft position vector recorded for each collection interval. This is done to ensure that the instrument is collecting data from the spot on the planet's surface that we think it is. The comparison is made by converting the calculated latitude and longitude, the instrument boresite vector, and the spacecraft position vector to be in the same frame of reference.

GRS geometry latitude and longitude are calculated using:

Calc_lat = 90 - grs_ lat **eq. 23**

Calc_lon = grs_lon < 0: grs_lon+360

Else: grs_lon; **eq. 24**

where grs_lat and grs_lon are the latitude and longitude associated with the spectrum of interest.

Spacecraft position and instrument boresite vectors are converted to latitude and longitude by the following equations:

lat = degrees(acos(z/sqrt($x^2 + y^2 + z^2$))); **eq. 25**

lon = degrees(if (x < 0) π+atan(y/x**) eq. 26**

else
if (y < 0) 2π + atan(y/x)
else atan(y/x));

where x, y and z are the x, y, and z coordinates of the vectors.

The standard deviation of the calculated longitude, the instrument boresite, and the spacecraft position latitudes are calculated and compared. In order to pass the latitude test, the standard deviation of the three must be within 0.1. The standard deviation of the three longitudes scaled by the cosine of the grs_lon are then calculated and compared. The standard deviation of the scaled longitudes must be less than 0.1 in order to pass the longitude check. The reason the longitude must be scaled is that longitude varies very quickly at the poles. If either the latitude or longitude or both fail the check, then the bad_code BAD_VECTOR is inserted into the bad_code field for that record. If the bad_code is set the record is not processed further.

5.0 LEVEL 1-C PROCESSING

5.1 SUMMIT_DB

A single gamma collection interval is on the order of 19.7 seconds, which is the equivalent of one degree of motion or 59 km over the surface. Total counts in a spectrum are on the order of 4000 counts over the 16384 channels of the spectrum. The small number of counts per spectrum yields an inadequate signal-to-noise ratio to be statistically significant. In order to improve the signal-to-noise ratio, and therefore acquire a statistically significant count rate, corrected gamma spectra are summed over temporal and spatial regions.

Gamma sums are created by the summit_db process for 5-degree latitude by 5-degree longitude grids over the time span of 15-degrees of areocentic longitude (L_S), where $L_S = 0$ is the vernal (spring) equinox within the northern hemisphere of Mars (see Time Conversion Table in the document directory for this data release for the relationship between Earth time and Mars time). At the completion of a Mars "season"(month), 15-degrees of L_S, summit_db queries the database for all corrected gamma spectra that have met the processing criteria (i.e. no bad flags set) within the temporal and spatial boundary. The spectra and the associated engineering data are then summed together by channels. The resulting spectrum and associated data are identical in format (i.e. 16384 channel spectrum) to the corrected spectra, except that there are now many more counts in each channel, thus yielding a scientifically useful data product.

6.0 APPENDIX A. – ENGINEERING CONVERSIONS

Engineering conversions are made by one of four different database processes. A list of the engineering identification values converted by each function is listed before the algorithm.

6.1 CASE 0:

```
CEB_AGND
CEB_PLUS_5_CRYO
CEB_PS1_TEMP
CEB_CPU_TEMP
CEB_HTR_CNTRL_TEMP
CEB_CPU_PLUS_5
CEB_CPU_PLUS_5_CURR
CEB_PS2_TEMP
CEB_MINUS_12V_CEB_AN
CEB_PLUS_12V_CEB_AN
CEB_PLUS_28_CURR
CEB_ALT_ACT_CURR
CEB_SPARE_CURR_SENSE_2
CEB_SPARE_CURR_SENSE_3
CEB_PC_CURR_REF
CEB_AGND_SPARE1
CEB_AGND_SPARE2
CEB_AGND_SPARE3
CEB_PS_CURR_REF
CEB_MAIN_ACTUATOR_CURR
CEB_AD_TEMP
GAMMA_MUX_OFFSET
PLUS_5V_ANLG
MINUS_5V_ANLG
PLUS_5V_CRNT_ANLG
MINUS_5V_CRNT_ANLG
PLUS_5V_CRNT_DIG
HVPS_MNTR_1
HVPS_MNTR_2
PREAMP_TEMP
```

```
FUNCTION eng_val_case0 (raw_in NUMBER, coefs_in COMMON.coef)
        RETURN NUMBER AS result NUMBER;
BEGIN
   RETURN (raw_in * coefs_in(2)) + coefs_in(1);
END;
```

6.2 CASE 1:

CEB_IS_TEMP_A
CEB_OS_TEMP_A
CEB_MNT_RNG_TEMP_A
CEB_IS_TEMP_B
CEB_OS_TEMP_B
CEB_MNT_RNG_TEMP_B

```
FUNCTION eng_val_case1 (raw_in NUMBER, coefs_in COMMON.coef)
        RETURN NUMBER AS result NUMBER;
BEGIN
   RETURN (((coefs_in(4) * raw_in + coefs_in(3)) * raw_in + coefs_in(2)) * raw_in + coefs_in(1));
END;
```

6.3 CASE 3:

GAMMA_ELEC_FAST
GAMMA_ELEC_SLOW
GAMMA_PLUS_12V_RAIL_VOLT
GAMMA_PLUS_12V_RAIL_CRNT
GAMMA_MINUS_12V_RAIL_VOLT
GAMMA_MINUS_12V_RAIL_CRNT
GAMMA_PLUS_5V_DIG_RAIL_VOLT
GAMMA_PLUS_5V_DIG_RAIL_CRNT
GAMMA_PLUS_12V_HVBS_RAIL_CRNT
GAMMA_MINUS_12V_HVBS_RAIL_CRNT
GAMMA_PLUS_12V_PULSE_AMP_CRNT
GAMMA_MINUS_12V_PULSE_AMP_CRNT
GAMMA_VDD_DIRECT_MNTR
GAMMA_REF_VOLT_166
GAMMA_REF_VOLT_333
GAMMA_REF_VOLT_500
GAMMA_DAC_0
GAMMA_DAC_1
GAMMA_DAC_2
GAMMA_DAC_3
GAMMA_DAC_4
GAMMA_DAC_5
GAMMA_DAC_6
GAMMA_DAC_7
GAMMA_HVBS_MNTR
GAMMA_SPARE_1
GAMMA_HVBS_ENABLE_MNTR
GAMMA_VREF_TEMP
GAMMA_SPARE_2

```
FUNCTION eng_val_case2 (veng_val_in NUMBER, coefs_in COMMON.coef, mux_in NUMBER)
```

```
        RETURN NUMBER AS result NUMBER;
          vmux  NUMBER := mux_in;
BEGIN
    IF vmux > .5 THEN vmux := 0; END IF;
    RETURN (veng_val_in - (coefs_in(2) * vmux));
END;
```

6.4 CASE 4:

```
 GAMMA_GPA_TEMP
 GAMMA_HVBS_TEMP
FUNCTION eng_val_case3 (raw_val_in NUMBER, coefs_in COMMON.coef, mux_in NUMBER,
           mux_coefs_in COMMON.coef)  RETURN NUMBER AS result NUMBER;
          vmux    NUMBER := mux_in;
BEGIN
--dbms_output.put_line('COEFS_IN: '||coefs_in(1)||' '||coefs_in(2)||'.'||coefs_in(3)||' '||coefs_in(4)||'.');
--dbms_output.put_line('MUX_COEF: '||mux_coefs_in(1)||'.');
--dbms_output.put_line('MUX_IN: '||vmux||'.');
    result := (coefs_in(4) * raw_val_in) + coefs_in(3);
    result := (result / 2) + mux_coefs_in(1);
          IF vmux > .5 THEN vmux := 0; END IF;
    result := result - vmux;
    result := (coefs_in(1) - result) / coefs_in(2);
--dbms_output.put_line('Before IF statement, result: '||result||'.');
    IF result < -9.99999 THEN
      result := -9.99999;
    END IF;
--dbms_output.put_line('After IF statement, result: '||result||'.');
    result := -1000 * (10 * result / (10 + result));
--dbms_output.put_line('Before natural log statement, result: '||result||'.');
    result := LN(result);
    result := 1 / ((.0010295) + (.0002391) * result + (.0000001568) * POWER(result,3));
          result := result - 273.15;
    RETURN result;
END;
```

6.5 ENGINEERING CONVERSION COEFFICIENTS TABLE

Engineering ID	Units	MUX Offset	Coef_in (4)	Coef_in (3)	Coef_in (2)	Coef _in (1)
CEB_AD_TEMP	Celsius	26	0	0	150	-273.24
CEB_AGND_SPARE1	Volts	21	0	0	2.9989	0
CEB_AGND_SPARE2	Volts	22	0	0	2.9989	0
CEB_AGND_SPARE3	Volts	23	0	0	2.9989	0
CEB_AGND	Volts	0	0	0	2.9989	0
CEB_ALT_ACT_CURR	Amps	17	0	0	0.869354	2.160449
CEB_CPU_PLUS_5	Volts	8	0	0	2.9989	0
CEB_CPU_PLUS_5_CURR	Amps	9	0	0	0.324643	0.577437
CEB_CPU_TEMP	Celsius	3	0	0	30.933	12.385
CEB_HTR_CNTRL_TEMP	Celsius	7	0	0	30.8995	12.31
CEB_IS_TEMP_A	Celsius	4	-0.274988	-1.394633	-63.846074	-28.2801
CEB_IS_TEMP_B	Celsius	11	-0.273272	-1.408146	-63.92085	-28.1253
CEB_MAIN_ACT_CURR	Amps	25	0	0	1.0596	2.773355
CEB_MINUS_12V_CEB_AN	Volts	13	0	0	6.00398	0
CEB_MNT_RNG_TEMP_A	Celsius	6	-0.277602	-1.378662	-63.820705	-28.8262
CEB_MNT_RNG_TEMP_B	Celsius	14	-0.272865	-1.39027	-64.82907	-27.9916
CEB_OS_TEMP_A	Celsius	5	-0.269665	-1.422079	-64.201152	-27.8551
CEB_OS_TEMP_B	Celsius	12	-0.286831	-1.323396	-63.829516	-29.16
CEB_PLUS_12V_CEB_AN	Volts	15	0	0	5.9985	0
CEB_PLUS_28_CURR	Amps	16	0	0	0.444404	0.620191
CEB_PLUS_5_CRYO	Volts	1	0	0	2.9989	0
CEB_PS1_TEMP	Celsius	2	0	0	30.9635	12.67
CEB_PS2_TEMP	Celsius	10	0	0	30.9765	12.885
CEB_SPARE_CURR_SENSE_2	Amps	18	0	0	1.059789	2.772596
CEB_SPARE_CURR_SENSE_3	Volts	19	0	0	2.9989	0
DOOR_CLOSE1		78	0	0	1	0
DOOR_CLOSE2		79	0	0	1	0
GAMMA_DAC_0	Volts	43	0	0	0.9998	2.4959
GAMMA_DAC_1	Volts	44	0	0	0.9998	2.4949
GAMMA_DAC_2	Volts	45	0	0	0.9998	2.4949
GAMMA_DAC_3	Volts	46	0	0	0.9998	2.4949
GAMMA_DAC_4	Volts	47	0	0	0.9998	2.4959
GAMMA_DAC_5	Volts	48	0	0	0.9998	2.4959
GAMMA_DAC_6	Volts	49	0	0	0.9998	2.4959
GAMMA_DAC_7	Volts	50	0	0	0.9998	2.4949
GAMMA_ELEC_FAST	nanoAmps	27	0	0	8.567	14.838
GAMMA_ELEC_SLOW	nanoAmps	28	0	0	8.567	14.838
GAMMA_GPA_TEMP	Celsius	55	1.9996	-0.0002	0.56958	-0.49601
GAMMA_HK1		80	0	0	1	0
GAMMA_HVBS_ENABLE_MNTR	Volts	53	0	0	0.9998	2.4949
GAMMA_HVBS_MNTR	Volts	51	0	0	999.80004	2495.9
GAMMA_HVBS_TEMP	Celsius	56	1.9996	-0.0002	0.56138	-0.46661

Engineering ID	Units	MUX Offset	Coef_in (4)	Coef_in (3)	Coef_in (2)	Coef_in (1)
GAMMA_MINUS_12V_HVBS_RAIL_CRNT	milliAmps	36	0	0	1.963472	4.416536
GAMMA_MINUS_12V_PULSE_AMP_CRNT	milliAmps	38	0	0	18.331501	40.59223
GAMMA_MINUS_12V_RAIL_CRNT	milliAmps	32	0	0	39.944069	101.7539
GAMMA_MINUS_12V_RAIL_VOLT	Volts	31	0	0	-2.365831	-5.97941
GAMMA_MUX_OFFSET	Volts	58	0	0	0.9998	2.4949
GAMMA_PLUS_12V_HVBS_RAIL_CRNT	milliAmps	35	0	0	9.335201	21.69841
GAMMA_PLUS_12V_PULSE_AMP_CRNT	milliAmps	37	0	0	24.231702	59.28018
GAMMA_PLUS_12V_RAIL_CRNT	milliAmps	30	0	0	39.085224	109.183
GAMMA_PLUS_12V_RAIL_VOLT	Volts	29	0	0	2.393011	6.033748
GAMMA_PLUS_5V_DIG_RAIL_CRNT	milliAmps	34	0	0	17.888711	46.29988
GAMMA_PLUS_5V_DIG_RAIL_VOLT	Volts	33	0	0	0.977322	2.526588
GAMMA_REF_VOLT_166	Volts	40	0	0	0.9998	2.4949
GAMMA_REF_VOLT_333	Volts	41	0	0	0.9998	2.4949
GAMMA_REF_VOLT_500	Volts	42	0	0	0.9998	2.4949
GAMMA_SPARE_1	Volts	52	0	0	0.9998	2.4959
GAMMA_SPARE_2	Celsius	57	0	0	476.095257	1186.143
GAMMA_VDD_DIRECT_MNTR	Volts	39	0	0	0.9998	2.4949
GAMMA_VREF_TEMP	Celsius	54	0	0	79.34921	196.1032
DELAYED_MAX		73	0	0	1	0
ET_HI		69	0	0	1	0
ET_LO		68	0	0	1	0
HVPS		75	0	0	1	0
HVPS_MNTR_1	Volts	64	0	0	999.80004	-0.29994
HVPS_MNTR_2	Volts	65	0	0	999.80004	-0.29994
LT_HI		71	0	0	1	0
LT_LO		70	0	0	1	0
MEMORY1		77	0	0	1	0
MINUS_5V_ANLG	Volts	60	0	0	9.998	-0.003
MINUS_5V_CRNT_ANLG	milliAmps	62	0	0	-999.80004	0.29994
MUX_RUN_SWAP_MEM		76	0	0	1	0
PIX		67	0	0	1	0
PLUS_5V_ANLG	Volts	59	0	0	9.998	-0.003
PLUS_5V_CRNT_ANLG	milliAmps	61	0	0	999.80004	-0.29994
PLUS_5V_CRNT_DIG	milliAmps	63	0	0	999.80004	-0.29994
PREAMP_TEMP	Celsius	66	0	0	99.980004	-0.02999
PRISM_ENABLE		74	0	0	1	0
PROMPT_MAX		72	0	0	1	0
PC_CUR_REF	Volts	20	0	0	2.9989	0
PS_CUR_REF	Volts	24	0	0	2.9989	0

7.0 APPENDIX B. – CORRECTION TABLES

7.1 NORMALIZATION TEMPERATURES AND GPA COEFFICIENTS

The following values are the normalization temperatures used in the correction calculations:

IS_NORMAL_TEMP	-188
IS_TEMPCO	188
B170K_NORM	-103
B170K_TEMPCO	-183

The following GPA coefficients are the three GPA coefficients at shaping amp gain 46 used in equation 12.

GPA0	32.3436
GPA1	3.6914850033
GPA2	0.0202259

7.2 INL COEFFICIENT MATRIX

-20.836241	985.278193	5.558464	-2.304907	0.911276	-0.293042	0.065569	-0.009724	0.000952	-0.000061	0.000002
-0.000000	0.000000	7.024847	-4.595868	-5.300300	-1.163911	3.393806	-1.924299	0.556998	-0.095570	0.010298
-0.000706	0.000030	-0.000001	0.000000	-12.393725	47.816678	-82.358940	76.018917	-38.610366	11.442877	-2.078611
0.239979	-0.017816	0.000835	-0.000023	0.000000	-0.000000	-24.745004	310.094800	-597.998526	573.506101	-315.420764
106.318922	-22.930891	3.255970	-0.307841	0.019196	-0.000759	0.000017	-0.000000	20.197512	183.034512	-462.583013
438.146052	-246.636289	89.527143	-21.245552	3.317817	-0.341889	0.022985	-0.000970	0.000023	-0.000000	-67.560713
-2676.475263	4699.824078	-4218.434418	2201.293015	-712.064505	148.997391	-20.707455	1.928569	-0.118972	0.004664	-0.000105
0.000001	813.338191	-3350.324163	7954.348093	-7434.762538	3864.202417	-1239.839533	258.890413	-36.091512	3.380272	-0.209825
0.008274	-0.000188	0.000002	2200.431796	11294.738534	-15984.945651	12850.850117	-6359.115105	2018.212239	-421.950439	59.159504
-5.586263	0.350241	-0.013969	0.000321	-0.000003	-7217.160206	10327.843892	-32224.577978	31333.252877	-16190.706979	5054.794092
-1017.861243	136.463772	-12.289199	0.733875	-0.027854	0.000608	-0.000006	-6822.427892	-20145.192679	21618.474788	-14475.309002
6570.252207	-2056.227614	439.950707	-64.110363	6.322849	-0.414081	0.017216	-0.000411	0.000004	17814.660649	-5515.611113
45327.655629	-48341.662722	25471.538636	-7891.390892	1556.917269	-203.100402	17.716139	-1.020945	0.037254	-0.000779	0.000

7.3 GAIN OFFSET TABLE

The gain and offset values listed here are the corrected gain and offset values that correspond to a particular shaping amp gain setting. Currently the gain and offset are known well only for shaping amp gain 46.

Shaping Amp Gain	Gain Channels/KeV	Offset
46	1.611579	9.151502

8.0 REFERENCES

Boynton, W.V., W.C. Feldman, I. Mitrofanov, L.G. Evans, R.C. Reedy, S.W. Squyres, R. Starr, I. Trombka, C. d'Uston, J.R. Arnold, P.A.J. Englert, A.E. Metzger, H. Wanke, J. Bruckner, D.M. Drake, C. Shinohara, C. Fellows, D.K. Hamara, K. Harshman, The Mars Odyssey gamma-ray spectrometer instrument suite, Space Sci. Rev., 110 (1-2): 37-83, 2004.

Marchand, P., L. Marmet, Binomial Smoothing Filter - A Way to Avoid Some Pitfalls of Least-Squares Polynomial Smoothing, Review of Scientific Instruments, 54 (8): 1034-1041, 1983.

Navigation and Ancillary Information Facility, Jet Propulsion Laboratory, September 15, 2003, http://pds-naif.jpl.nasa.gov/