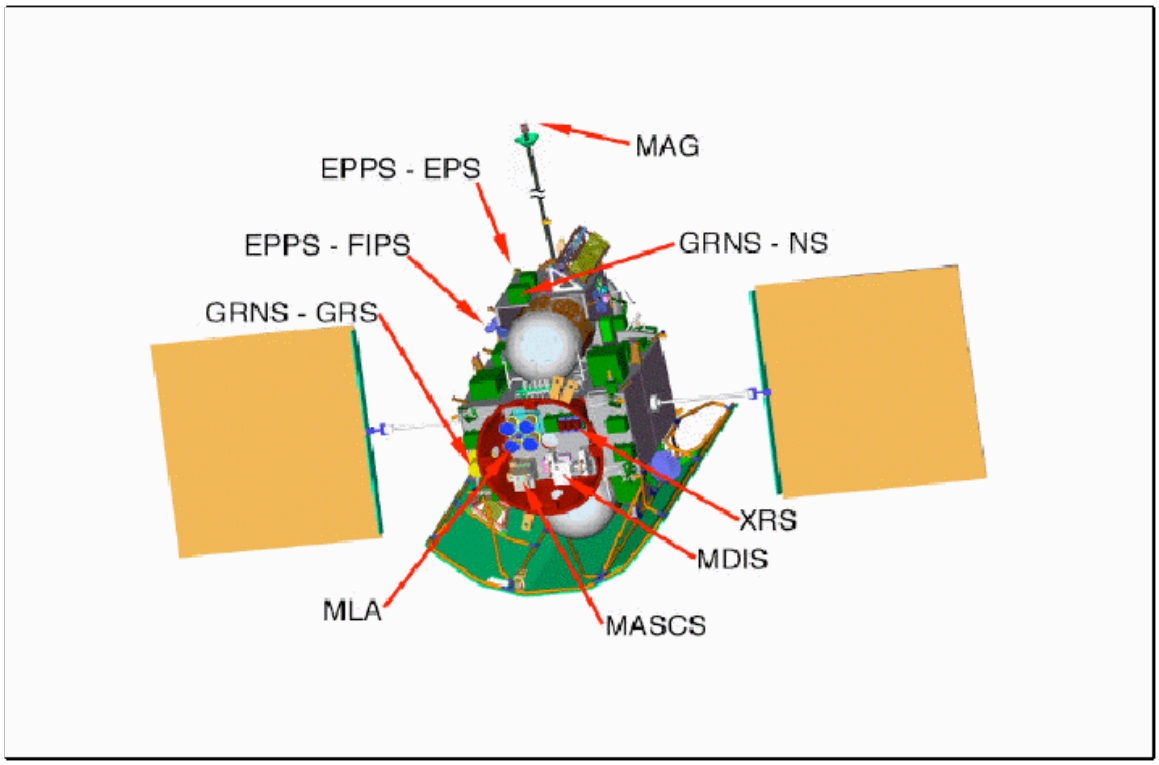
MESSENGER:
Software Interface Specification
for the
Mercury Laser Altimeter
Experiment Data Record

Revision 2j



Prepared by:

Raymond Espiritu

Erick Malaret
Applied Coherent Technology Corporation
112 Elden St, Suite K
Herndon, VA 22304
http://www.actgate.com

Document Review

This document and the archive it describes have been through PDS Peer Review and have been accepted into the PDS archive.

Greg Neumann, MESSENGER MLA Instrument Scientist, has reviewed and approved this document.

Susan Slavney, PDS Geosciences Node Representative, has reviewed and approved this document.

Susan Ensor, MESSENGER Science Operations Center Lead, has reviewed and approved this document.

Document Change History

Revision Number	Revision Date	Author	Section	Remarks
2f	6/17/11	S. Ensor, SOC	Title	1. Add Document Change History table. 2. Add "Experiment Data Record" to document title. 3. Replace signature page with document review information.
2g	5/25/12	S. Ensor, SOC	2, 7	1. Change document title *Data Management and Science Analysis Plan* to *Data Management and Archiving Plan* and update references. 2. Remove PDS delivery schedule, table 1, and reference *Data Management and Archiving Plan.*
2h	12/3/2013	R. McNutt	2	Update references to reflect current versions
2i	12/12/2013	S. Ensor	Title	Change document title from "MESSENGER: Experiment Data Record Software Interface Specification For The Mercury Laser Altimeter Experimental Data Record" to "MESSENGER: Software Interface Specification For The Mercury Laser Altimeter Experiment Data Record"
2j	7/1/2015	S.Ensor	Various, 5.4.2, 6.4	Change "Experimental Data Record" to "Experiment Data Record" in text. Note use of clock partitions in time tags in product labels following January 8, 2013 S/C clock reset. Updated reference to PDS file naming standard (was 27.3 now 36.3).

Table of Contents

1 Purpose and Scope of Document

1.1 Purpose

This document will serve to provide users of the MESSENGER MLA EDR data products with a detailed description of the MLA instrument, EDR product generation, validation, and storage. The MLA EDR data products are deliverables to the Planetary Data System (PDS) and the scientific community that it supports. All data formats are based on the PDS standard. The document is both an EDR data product SIS and an EDR archive volume SIS.

The Committee on Data Management and Computation (CODMAC) has defined levels of data product. The CODMAC Level 2 product is the Experiment Data Record (EDR). The EDR is mainly useful as an input to producing the Calibrated Data Record (CDR) and Reduced Data Record (RDR) products. The CDR and RDR products are calibrated to scientific and engineering units. They provide ranges, laser pulse measurements, and instrument housekeeping for scientific and engineering purposes. Range data are merged with the Spacecraft, Planet, Instrument, C-matrix Events (SPICE) archives of instrument orientation, spacecraft orbit position and pointing, timing, planetary orientation and ephemerides to locate the Mercury Laser Altimeter (MLA) ranges in a planet-fixed, center-of-mass coordinate system. The merged data are aggregated into the CODMAC Level 3 RDR. The RDR is the primary altimeter science data product, from which resampled topographic models may be obtained. Measurements of surface albedo and roughness may be derived from the RDR, as well as more precise models of spacecraft trajectory, planetary gravity, and libration.

1.2 Scope

This Software Interface Specification (SIS) document is of a very limited scope due to the EDRs being of a very low-level. It is not intended for general use by data analysts outside of the MESSENGER project and it is not intended for the typical science user to access the EDRs routinely. Rather, it is mainly useful as an input to producing the CDR and RDR products. There will be a separate PDS data archive for the CDR and RDR products and these products will be described by a separate RDR SIS. The RDRs will be accessible as ASCII tables.

2 Applicable Documents

The MESSENGER MLA SIS is responsive to the following Documents:

- Planetary Data System Standards Reference, August 1, 2003, Version 3.6. JPL D-7669, Part-2
- MESSENGER Data Management and Archiving Plan (Rev. B, 7 May 2013), The Johns Hopkins University, APL. Document ID number 7384-9019, http://pds-geosciences.wustl.edu/missions/messenger/docs/MESSENGER_DMAP_RevB.pdf

- MESSENGER Mercury: Surface, Space Environment, Geochemistry, Ranging; A mission to Orbit and Explore the Planet Mercury, Concept Study, March 1999. Document ID number FG632/99-0479
- Mercury Laser Altimeter instrument design, testing, and performance verification, X. Sun, J. F. Cavanaugh, J. C. Smith, and A. E. Bartels, *Proceedings of the 22nd International Laser Radar Conference (ILRC 2004), Special Publication SP-561*, pp. 961-964, European Space Agency, Noordwijk, The Netherlands, 2004
- The Mercury Laser Altimeter Instrument for the MESSENGER Mission, John F. Cavanaugh, James C. Smith, Xiaoli Sun, Arlin E. Bartels, Luis Ramos-Izquierdo, Danny J. Krebs, Jan F. McGarry, Raymond Trunzo, Anne Marie Novo-Gradac, , Jamie L. Britt, Jerry Karsh, Richard B. Katz, Alan Lukemire, Richard Szymkiewicz, Daniel L. Berry, Joseph P. Swinski, Gregory A. Neumann, Maria T. Zuber, and David E. Smith, *Space Sciences Reviews*, **131**, 451-479, 2007.
- [PLR] Appendix 7 to the Discovery Program Plan; Program Level Requirement for the MESSENGER Discovery Project; June 20, 2001.

3 Relationships with Other Interfaces

The MLA EDR data products are stored on Hard Disk and in an SQL (Structured Query Language) relational database for rapid mission access during mission operations. The data products will be electronically transferred to the PDS Geosciences Node according to the delivery schedule in the *MESSENGER Data Management and Archiving Plan*. The data in the EDR files themselves will be stored in a PDS binary TABLE object.

4 Roles and Responsibilities

The roles and responsibilities of the instrument teams, Applied Physics Lab (APL), Applied Coherent Technology (ACT), and the Planetary Data System (PDS), are defined in the MESSENGER Data Management and Archiving Plan.

5 Data Product Characteristics and Environment

5.1 Overview

MESSENGER stands for MErcury Surface, Space ENvironment, GEochemistry, and Ranging. It is a scientific spacecraft designed to aid in the study of the planet Mercury. It is the first mission to go to Mercury since Mariner 10 in 1975. MESSENGER's orbit about Mercury is highly elliptical, passing 200 km above the surface at its lowest point (periherm) and more than 15,000 km at its highest point (apoherm). The plane of the orbit is inclined 80° to Mercury's rotation axis, and the low point in the orbit is reached at latitude 60° N. MESSENGER's 12 months in orbit cover 2 Mercurian solar days. Because Mercury's rotation period (~58 Earth days) is two thirds of its revolution period around the Sun (~88 Earth days), the solar day, from sunrise to sunrise, is equal to ~176 Earth days. The first solar day of orbital operations is focused on obtaining global map products from the different instruments, and the second focuses on targeted science investigations. At different times during Mercury's solar orbit, the Sun, Mercury, and MESSENGER are in different configurations, necessitating different observing strategies.

Figure 1 shows the MESSENGER orbit and the corresponding MLA instrument operational modes. MESSENGER has a 12-hour orbit of which approximately 25-50 minutes are dedicated to MLA science, depending on solar keep-out constraints. The figure shows a notional orbit in which the planet is within range for 35 minutes. Approximately 15 minutes prior to taking science data the MLA instrument is put into standby mode. The laser is powered on at the start of this mode, and time is provided to allow the analog and laser electronics to warm up. Standby uses less power than science mode because the laser is not firing. The length of the science mode will vary during the course of Mercury's orbit about the sun. For the remaining 670 minutes of the orbit MLA is in Keep Alive mode. This is a low power mode in which the analog and laser electronics are powered off.

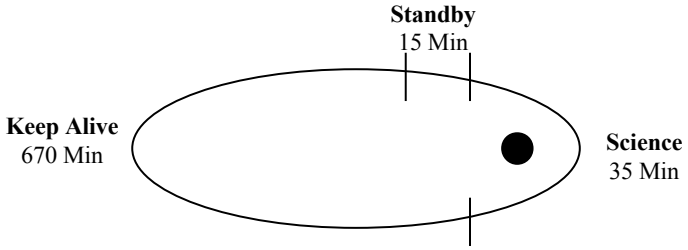


Figure 1: Messenger Orbit

The MLA is used to calculate the range between the spacecraft and Mercury's surface. The transmitter generates a brief laser pulse, and the instrument measures the time required for the light to reach the surface and return. The time-of-flight measurements will be used to make accurate determinations of Mercury's shape and will help determine its rotational dynamics. These shape and kinematics measurements, when combined with analyses of the very precisely tracked spacecraft orbit around Mercury, will yield information on its internal density structure. More information on the MLA instrument is provided in Appendix – MLA Instrument Overview.

5.2 Data Product Overview

The MLA EDR products are grouped together into one data set. Within that data set there are three EDR data products. Each MLA EDR data product consists of two files. One contains the data itself, and is arranged in a PDS compliant binary table file. The other is a PDS label file that describes the content of the table file. The label file defines the start time and end of the observation, product creation time, etc. The label file also describes the different fields within the table.

During the Mercury Orbit mission phase a single data file will contain the observations obtained in one orbit of the spacecraft around Mercury. Prior to the Mercury Orbit mission phase a single data file will aggregate the observations such that all data within the file are taken on the same year, month, day, and hour. This was deemed the most efficient way to archive data resulting from instrument commands which would turn the

instrument on, generate data for upwards of several hours, and then turn the instrument off. The three EDR data products are described as follows:

Science (RAW) EDR
Contains the ranging information as collected by the instrument in "Science" mode. Designated as raw science data as none of the values have been calibrated or converted into engineering units.

Status (STATUS) EDR
Contains the instrument status information, such as voltages, temperatures, and timing parameters. Note that the measurements such as voltage and temperature values are stored in the EDR as the original telemetry counts. The Status CDR will contain the engineering values in the appropriate units.

Hardware Diagnostic Lite (HAD) EDR
Contains diagnostic information about the instrument and background brightness information from the detector at 8 Hz resolution. It is designated as "Lite" because the EDR is obtained from the "lite" version of the hardware diagnostic telemetry packet. This differentiates it from the "full" version of the packet, which contains more diagnostic fields, but will not be utilized during the course of the mission due to bandwidth limitations. As with the other MLA EDRs all values are the original telemetry counts. The HAD CDR will contain the engineering values in the appropriate units.

5.3 Data Processing

5.3.1 Data Processing Level

There is one EDR Data Archive Volume for the MLA instrument. The data volume will contain level 2 CODMAC data products, also known as EDRs. Each product will have a unique file name and conform to the file naming convention in section 6.4. All EDR products will be stored at the Applied Physics Laboratory/Science Operations Center (APL/SOC) during mission operations. Volumes will be electronically transferred to the PDS Geosciences Node following the procedure in section 5.3.3.

5.3.2 Data Product Generation

The MLA EDR files will be produced by the MESSENGER Science Operations Center (SOC), operated jointly by APL and ACT. Inputs to the SOC will consist of telemetry in the form of CCSDS packets. Data downlink is telemetered through NASA's Deep Space Network (DSN) managed by the Jet Propulsion Laboratory in Pasadena, CA, and then forwarded to APL. Level 1 CODMAC data is extracted from the telemetry packets and stored online at the SOC. The 'PIPE-MLA2EDR' software packages the Level 1 CODMAC data to the PDS format defined in this SIS (section 6.2). The EDR data products are made available to the MESSENGER Science Team for initial evaluation and validation. At the end of the evaluation and validation period, the data are organized and stored in the directory structure described in section 6.5 for transmittal to the Geosciences Node. The transmittal process is described in section 5.3.3, Data Flow.

5.3.3 Data Flow

The MESSENGER SOC operates under the auspices of the MESSENGER Project Scientist to plan data acquisition, and to generate and validate data archives. The SOC supports and works with the MOC, the Science Team, instrument scientists, and the PDS. The SOC will be located at the Johns Hopkins University/Applied Physics Lab (JHU/APL). The Data flow diagram in figure 2 shows the general flow of data within the MESSENGER project and data flow to PDS. The MOC handles raw data flow to and from the MESSENGER spacecraft and the SOC converts the raw telemetry into EDRs. The Science Team validates the EDRs and notifies the SOC if corrections are needed. Documentation and EDRs are delivered to the PDS Geosciences Node. SPICE kernels are delivered to the PDS Navigation and Ancillary Information (NAIF) node.

The MESSENGER SOC will deliver data for the MLA EDR data volume to the PDS Geosciences Node in standard product packages. Each package will comprise data and files organized into directory structures consistent with the volume design described in section 6.5.

The following describes the electronic transfer process of releasing data to PDS. This transfer process will be used for the first PDS delivery. Future data deliveries will be assumed to follow the same process unless otherwise noted in an update of this document. Given the long duration of the mission the project is reserving the option of exploring alternate data delivery methods for subsequent deliveries. As such, the method of electronic transfer may change and will be revised accordingly in the SIS. Any changes to the delivery process will be noted in an update to the SIS document and will include the specific dates which will use the new delivery process. The delivery of products to the data volume will follow the schedule in the *MESSENGER Data Management and Archiving Plan*.

In the week prior to the delivery date the directory structure will be compressed into a single “zip archive” file for transmittal to the PDS node. The zip archive preserves the directory structure internally so that it can be recreated after electronic delivery to the PDS node. The zip archive file is transmitted to the PDS node via FTP to an account set up by the receiving node. Also transmitted will be a checksum file created using the MD5 algorithm. This provides an independent method of verifying the integrity of the zip file after it has been sent. Within days of transmittal the PDS node will acknowledge receipt of the archive and checksum file. If acknowledgement is not received, or if problems are reported, the MESSENGER SOC will immediately take corrective action to effect successful transmittal.

After transmittal the PDS node will uncompress the zip archive file and check for data integrity using the checksum file. The node will then perform any additional verification and validation of the data provided and will report any discrepancies or problems to the MESSENGER SOC. It is expected that the node will perform these checks in about two weeks. After inspection has been completed to the satisfaction of the PDS node, the node will issue to the MESSENGER SOC acknowledgement of successful receipt of the data.

Following receipt of a data delivery the PDS node will organize the data into a PDS volume archive structure within its online data system. Newly delivered data will be made available publicly from PDS once accompanying labels and other documentation have been validated.

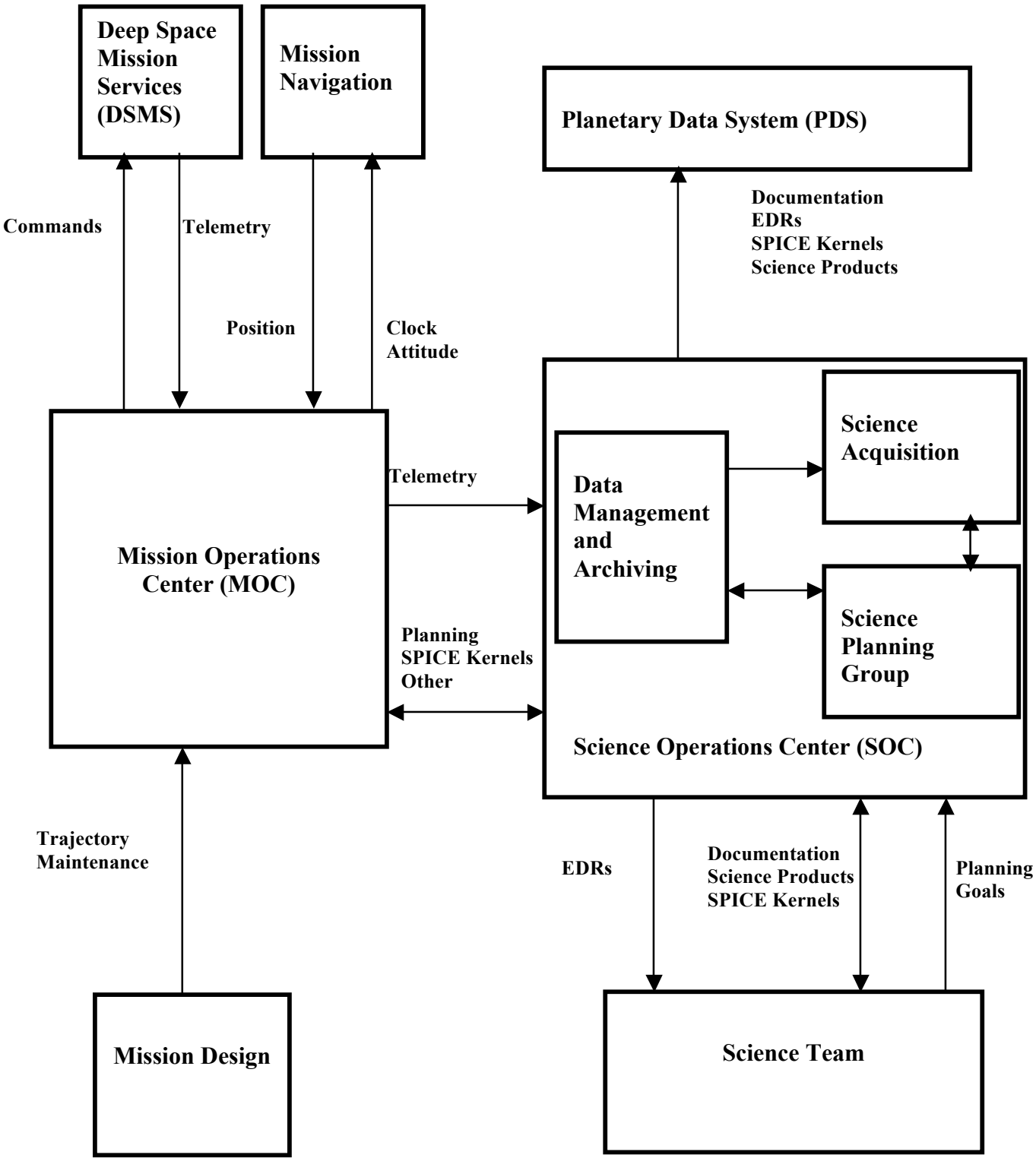


Figure 2. MESSENGER data flow

5.3.4 Labeling and Identification

There is a PDS label file for every MLA EDR data product. Example label files are shown in Appendices 8.1, 8.2, 8.3 for the Raw Science, Status, and Hardware Diagnostic Lite EDR products. The specific table structure for each data file is detailed in section 6.2.

5.4 Standards Used in Generating Data Products

5.4.1 PDS Standards

The MLA EDR data products are constructed according to the data object concepts developed by the PDS. By adopting the PDS format, the MLA EDR data products are consistent in content and organization with other planetary data collections. In the PDS standard, the EDR data file is grouped into objects with PDS labels describing the objects. The EDR data product contains

- A binary table object (the primary data) contained in a data file.
- A detached PDS label file. This contains a pointer to the data file as well as meta-data information and a detailed description of the binary table structure.

5.4.2 Time Standards

The MET field in the MLA EDR binary table matches the spacecraft time in integer seconds that is transmitted to MESSENGER subsystems by the Integrated Electronics Module (IEM). This is referred to by the MESSENGER project as Mission Elapsed Time (MET). MET = 0 is August 3, 2004, at 05:59:16 UTC, which is 1000 seconds prior to the MESSENGER launch. Relativistic effects and circumstances occurring during the mission would result in MET not being a true account of seconds since launch. Following a planned spacecraft clock reset[1] on January 8, 2013, partition numbers (1/, or 2/) were added to product labels to disambiguate MET seconds after the spacecraft clock reset (if partition number is not present, SPICE defaults to partition 1/). For this reason the MESSENGER spacecraft clock coefficients file is archived at the PDS Navigation and Ancillary Information Facility (NAIF) Node. This file is used in conjunction with the leapseconds kernel file in order to calculate the conversion between MET and UTC.

The conversion is easily done through the use of SPICE kernels and the CHRONOS Utility. CHRONOS is a utility included with the SPICE package that is distributed by the PDS NAIF node. The SPICE kernels are files that contain the information needed to perform the conversion. Two SPICE kernels are required. One is the Leapseconds Kernel (LSK) and the other is the MESSENGER Spacecraft Clock Kernel (SCLK). The SCLK file is used by CHRONOS to convert between spacecraft clock time and ephemeris time, while the LSK file is used to convert from ephemeris time to UTC time. The CHRONOS

[1] See instrument host catalog file in MLA EDR volume catalog directory for more information on MESSENGER spacecraft clock reset.

utility is self-documenting and the SPICE package itself contains full documentation on each of the utilities (including CHRONOS) and how they are used.

5.4.3 Data Storage Conventions

The data are organized following PDS standards and stored on hard disk and in an SQL (Structured Query Language) database for rapid access during mission operations. The MESSENGER SOC will transfer data to PDS via electronic transfer and delivery methods detailed in section 5.3.3. After verification of the data transfer PDS will provide public access to MESSENGER science data products through its online data distribution system.

5.5 Data Validation

The MLA EDR data products will be validated by the MLA Instrument scientist for science content and for compliance with PDS archive standards [MESSENGER Data Management and Archiving Plan].

6 Detailed Data Product Specifications

6.1 Data Product Structure and Organization

The MESSENGER MLA data set will be archived at the PDS Geosciences node as a data archive volume. The MLA EDR products in the data archive volume are intended to store the data in a form closest to the raw telemetry data received from the spacecraft. The automated production and release of EDRs will follow the release schedule in the *MESSENGER Data Management and Archiving Plan*. If errors are discovered the data will be replaced with corrected EDRs on the next scheduled delivery date.

6.2 Data Format Description

Data is stored in binary table format . The structure of the binary table is defined by an external format (.FMT) file. The MLA Raw Science EDR is a binary table with all of the packet telemetry points needed to analyze the instrument data, and is only produced when the instrument is in Science Mode and compression is not enabled. It is a Level 2 product, in that units are not resampled or converted. The range data consist of multiple coarse and fine timing counts for transmitted and received pulses that must be combined with instrument configuration data. The Science CDR will contain all ranges and noise returns in scientific units, conforming to CODMAC Data Level 3. The Science RDR will contain edited, geolocated ground returns and ancillary data in scientific units, and conform to CODMAC Data Level 4.

The MLA Hardware Diagnostic Lite EDR is derived from an engineering-mode packet used for calibration experiments and contains a sample of the range measurement data. These packets are generated only in Standby Mode. The MLA Status EDR is a low-rate packet that contains instrument environmental parameters needed to interpret the science data, and may be generated in all modes.

The following tables present the structure of the binary table in a user-friendly format.

6.2.1 Raw Science EDR Binary Table

1. **MET**
   Bytes: 4 Data_Type: MSB_UNSIGNED_INTEGER

Time tag in seconds.

2. **RX_EDGE_MISCOMPARE**
   Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Flag of timestamps comparison between high return pulse leading and trailing edge.
=0 normal, =1 trailing edge is timestamped before leading edge.

3. **RMU_BABBLEBIT**
   Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Check that the bus between RMU and CPU is quiet for 15ms between RMU time zero and RUPT.
=0 no violation, =1 at least one quiet time violation has occurred.

4. **ENDOF_SUPERFRAME**
   Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

A superframe is 10 seconds of science data. This flags the end of the superframe on the 10^{th} second.
=0, seconds 1-9 of superframe; =1 10^{th} second of superframe.

5. **ENDOF_FRAME**
   Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

A frame is two seconds of science data. Some calculations are made across frames.
=0 second one; =1 second two.

6. **CH3_CONFIG**
   Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The detector 3 configuration assumed by the science task.
=0 disabled; =1 enabled.

7. **CH2_CONFIG**
   Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The detector 2 configuration assumed by the science task.
=0 disabled; =1 enabled.

8. **CH1_LO_CONFIG**
   Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The detector 1 low configuration assumed by the science task.
=0 disabled; =1 enabled.

9. **CH1_HI_CONFIG**
   Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The detector 1 high configuration assumed by the science task.
=0 disabled; =1 enabled.

10. **TIMING_VALID**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Check of timing integrity at a rate of once per second. Three checks are performed:

1. Has MET message from S/C been received?
2. Has the 1PPS signal from the S/C been received?
3. Is the phase lock to the 1PPS off by more than 5 microseconds?

If any of these checks fail then the timing integrity is in question.
=0 timing integrity fine; =1 timing integrity under question

11. CURR_SC_FLAG
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Reserved spare value.

12. MODE_2_SUBMODE
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Reserved spare value.

13. SC_RANGE_MODE
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

=0 latest range message received is valid; =1 latest range message received is invalid.

14. ALGORITH_MODE
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The actual algorithm mode being run.
=0 Fixed Mode, =1 Spacecraft Range Mode.

15. LSR_PULSE_WID_MAX
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Maximum width of laser pulse, measured by the HSI as microseconds between rising and falling edge on the laser diode pulse signal. Maximum value out of 8 shots fired in one second.

16. LSR_PULSE_WID_MIN
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Minimum width of laser pulse, measured by the HSI as microseconds between rising and falling edge on the laser diode pulse signal. Minimum value out of 8 shots fired in one second.

17. LSR_PULSE_WID_MEAN
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Average width of laser pulse, measured by the HSI as microseconds between rising and falling edge on the laser diode pulse signal. Average of 8 shots fired in one second.

18. STARTPLS_LEAD_COARSE
Bytes:16 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes: 2 Items: 8
Coarse time counts of the leading edge start pulse for each of the eight shots fired in one second.

19. STARTPLS_LEAD_FINE
Bytes:16 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes: 2 Items: 8

Fine time counts of the leading edge start pulse for each of the eight shots fired in one second.

20. **STARTPLS_TRAIL_COARSE**
    Bytes:8 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes: 1 Items: 8

Coarse time counts of the trailing edge start pulse for each of the eight shots fired in one second.

21. **STARTPLS_TRAIL_FINE**
    Bytes:16 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes: 2 Items: 8

Fine time counts of the trailing edge start pulse for each of the eight shots fired in one second.

22. **DIODE_CURR_MAX**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Raw A/D counts of maximum measured laser diode current.

23. **DIODE_CURR_MIN**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Raw A/D counts of minimum measured laser diode current.

24. **DIODE_CURR_MEAN**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Raw A/D counts of average measured laser diode current.

25. **TX_PLS_ENERGY**
    Bytes:8 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes: 1 Items: 8

Raw A/D counts of the measured transmitted pulse energy for each of eight shots fired in one second.

26. **CH1HI_PLS_ID**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Flag noting the status of the each of eight returns detected by Channel 1 High.
=0 invalid pulse; =1 valid pulse.

27. **RANGE_GATE_START**
    Bytes:2 Data_Type: MSB_UNSIGNED_INTEGER

Range gate start read back from RMU; units of 30 meters.

28. **RANGE_GATE_STOP**
    Bytes:2 Data_Type: MSB_UNSIGNED_INTEGER

Range gate stop read back from RMU; units of 30 meters.

29. **CH1_HI_THLD_SHOT1**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The commanded value of the threshold for channel 1 high, shots 1-4. Calculated every 4 shots by science task.

30. **CH1_HI_THLD_SHOT5**

Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The commanded value of the threshold for channel 1 high, shots 5-8. Calculated every 4 shots by science task.

31. **CH1_LO_THLD_SHOT1**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The commanded value of the threshold for channel 1 low, shots 1-4. Calculated every 4 shots by science task.

32. **CH1_LO_THLD_SHOT5**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The commanded value of the threshold for channel 1 low, shots 5-8. Calculated every 4 shots by science task.

33. **CH2_THLD_SHOT1**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The commanded value of the threshold for channel 2, shots 1-4. Calculated every 4 shots by science task.

34. **CH2_THLD_SHOT5**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The commanded value of the threshold for channel 2, shots 5-8. Calculated every 4 shots by science task.

35. **CH3_THLD_SHOT1**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The commanded value of the threshold for channel 3, shots 1-4. Calculated every 4 shots by science task.

36. **CH3_THLD_SHOT5**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The commanded value of the threshold for channel 3, shots 5-8. Calculated every 4 shots by science task.

37. **VGA_SETTING**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Commanded value for VGA gain, sampled for first shot.

38. **NOISE_CH1_HI_1_4**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of noise hits detected on Channel 1 high, summed over shots 1-4.

39. **NOISE_CH1_HI_5_8**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of noise hits detected on Channel 1 high, summed over shots 5-8.

40. **NOISE_CH1_LO_1_4**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of noise hits detected on Channel 1 low, summed over shots 1-4.

41. **NOISE_CH1_LO_5_8**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of noise hits detected on Channel 1 low, summed over shots 5-8.

42. **NOISE_CH2_1_4**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of noise hits detected on Channel 2, summed over shots 1-4.

43. **NOISE_CH2_5_8**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of noise hits detected on Channel 2, summed over shots 5-8.

44. **NOISE_CH3_1_4**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of noise hits detected on Channel 3, summed over shots 1-4.

45. **NOISE_CH3_5_8**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of noise hits detected on Channel 3, summed over shots 5-8.

46. **RANGE_RATE**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Current spacecraft range - last known good range divided by the number of seconds between them. Expressed in units of 30 meters per second. Calculated once per second.

47. **SIGNAL_BINS**
    Bytes:2 Data_Type: MSB_UNSIGNED_INTEGER

For odd numbered seconds it gives the signal bin number for the appropriate lower histogram. For even numbered seconds it gives the signal bin number for the upper histogram.

48. **BIN_THLD**
    Bytes:2 Data_Type: MSB_UNSIGNED_INTEGER

For odd numbered seconds it is the minimum count needed in a bin of a lower histogram in order to establish signal for that histogram. For even numbered seconds it gives the minimum count of the number of returns summed over all of the bins in the upper histogram that are needed to establish a signal.

49. **FRAME_RX_PROC_CNT**
    Bytes:2 Data_Type: MSB_UNSIGNED_INTEGER

Number of returns tabulated in a lower histogram during a frame (2 seconds).

50. **SC_RANGE**
    Bytes:2 Data_Type: MSB_UNSIGNED_INTEGER

The spacecraft range value used by the science task for algorithm calculations. Note: as soon as the science task receives a range message it begins using the new range in its calculations. Thus the spacecraft range in telemetry may NOT be the one used throughout the entire second of algorithm calculations.

51. **SPARE**

Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Spare column.

52. OUT_OF_SYNC
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The first shot of every second the science task checks itself to determine if it is in synch with the flight software.
=0 in synch with 1PPS. =1 out of synch with 1PPS.

53. STARTPLS_INVALID
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

=0 all start pulses for the second were valid.
=1 at least one start pulse during the second was invalid.

54. RMU_WRITE_ERR
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Verification of range gate values written to the RMU.
=0 no error. =1 error

55. INVALID_CONFIG
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Flags an invalid configuration for science task and spacecraft range mode.
=0 valid config, =1 invalid config.

56. CORRUPT_MEM
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Science task check on variables that should never go outside a certain range. This should never be 1.
=0 variables okay. =1 variables not okay.

57. SIGNAL_FOUND
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether upper or lower histograms have been found by science task when processing 10 seconds worth of science data.
=0 no signal.
=1 upper histogram (high returns) found.
=2 lower histogram (low returns) found
=3 both upper and lower histogram found.

58. SIG_FRAM_PER_SUPER
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

The number of frames in the superframe showing a signal. There are 5 frames per superframe. NOTE: There must be at least 4 out of 5 frames showing signal in order to have a signal based on the lower histograms.

59. RDOT_FIT_ERR
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Reserved spare value.

60. SECS_TO_CMP_RDOT
Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Number of seconds used when computing range rate. Nominally one. If value reaches 10 then science task will stop trying to use spacecraft range.

61. **TIME_1HZ_TO_RUPT_0_15**
    Bytes:2 Data_Type: MSB_UNSIGNED_INTEGER

The time in from the 1PPS signal to the first RUPT of the second. This telemetry point is the least significant 16 bits of the time.

62. **TIME_1HZ_TO_RUPT_16_23**
    Bytes:2 Data_Type: MSB_UNSIGNED_INTEGER

The time in from the 1PPS signal to the first RUPT of the second. This is the most significant 8 bits of the time.

63. **CH1_HI_RX_LEAD_COARSE**
    Bytes:16 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes: 2 Items: 8

Coarse time counts of the leading edge channel 1 high returns. One timestamp per shot, 8 shots total.

64. **CH1_HI_RX_LEAD_FINE**
    Bytes:16 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes: 2 Items: 8

Fine time counts of the leading edge channel 1 high returns. One timestamp per shot, 8 shots total.

65. **CH1_HI_RX_TRAIL_COARSE**
    Bytes:8 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes: 1 Items: 8

Coarse time counts of the trailing edge channel 1 high returns. One timestamp per shot, 8 shots total.

66. **CH1_HI_RX_TRAIL_FINE**
    Bytes:16 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes: 2 Items: 8

Fine time counts of the trailing edge channel 1 high returns. One timestamp per shot, 8 shots total.

67. **WIDE_FILT_RX_CNT_1**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Total number of wide filter low returns for shot 1, maximum is 10.

68. **WIDE_FILT_RX_CNT_2**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Total number of wide filter low returns for shot 2, maximum is 10.

69. **WIDE_FILT_RX_CNT_3**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Total number of wide filter low returns for shot 3, maximum is 10.

70. **WIDE_FILT_RX_CNT_4**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Total number of wide filter low returns for shot 4, maximum is 10.

71. **WIDE_FILT_RX_CNT_5**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Total number of wide filter low returns for shot 5, maximum is 10.

72. **WIDE_FILT_RX_CNT_6**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Total number of wide filter low returns for shot 6, maximum is 10.

73. **WIDE_FILT_RX_CNT_7**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Total number of wide filter low returns for shot 7, maximum is 10.

74. **WIDE_FILT_RX_CNT_8**
    Bytes:1 Data_Type: MSB_UNSIGNED_INTEGER

Total number of wide filter low returns for shot 8, maximum is 10.

75. **SHOT1_RX_LEAD_ID**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

IDs for leading edge of shot 1, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

76. **SHOT1_RX_LEAD_TIME_COARSE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Coarse time counts of the leading edge shot 1, low return pulses. If less than 10 returns collected then unused item slots are 0.

77. **SHOT1_RX_LEAD_TIME_FINE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Fine time counts of the leading edge shot 1, low return pulses. If less than 10 returns collected then unused item slots are 0.

78. **SHOT1_RX_TRAIL_ID**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

IDs for trailing edge of shot 1, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

79. **SHOT1_RX_TRAIL_TIME_COARSE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Coarse time counts of the trailing edge shot 1, low return pulses. If less than 10 returns collected then unused item slots are 0.

80. **SHOT1_RX_TRAIL_TIME_FINE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Fine time counts of the trailing edge shot 1, low return pulses. If less than 10 returns collected then unused item slots are 0.

81. **SHOT1_RX_VALID_RETURN**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

Flag whether the shot 1 returns are valid. Valid returns are defined as having the same non-zero lead and trail ID value. =1 valid; =0 invalid

82. **SHOT2_RX_LEAD_ID**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

IDs for leading edge of shot 2, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

83. **SHOT2_RX_LEAD_TIME_COARSE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Coarse time counts of the leading edge shot 2, low return pulses. If less than 10 returns collected then unused item slots are 0.

84. **SHOT2_RX_LEAD_TIME_FINE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Fine time counts of the leading edge shot 2, low return pulses. If less than 10 returns collected then unused item slots are 0.

85. **SHOT2_RX_TRAIL_ID**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

IDs for trailing edge of shot 2, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

86. **SHOT2_RX_TRAIL_TIME_COARSE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Coarse time counts of the trailing edge shot 2, low return pulses. If less than 10 returns collected then unused item slots are 0.

87. **SHOT2_RX_TRAIL_TIME_FINE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Fine time counts of the trailing edge shot 2, low return pulses. If less than 10 returns collected then unused item slots are 0.

88. **SHOT2_RX_VALID_RETURN**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

Flag whether the shot 2 returns are valid. Valid returns are defined as having the same non-zero lead and trail ID value. =1 valid; =0 invalid

89. **SHOT3_RX_LEAD_ID**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

IDs for leading edge of shot 3, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

90. **SHOT3_RX_LEAD_TIME_COARSE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Coarse time counts of the leading edge shot 3, low return pulses. If less than 10 returns collected then unused item slots are 0.

91. **SHOT3_RX_LEAD_TIME_FINE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Fine time counts of the leading edge shot 3, low return pulses. If less than 10 returns collected then unused item slots are 0.

92. **SHOT3_RX_TRAIL_ID**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

IDs for trailing edge of shot 3, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

93. **SHOT3_RX_TRAIL_TIME_COARSE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Coarse time counts of the trailing edge shot 3, low return pulses. If less than 10 returns collected then unused item slots are 0.

94. **SHOT3_RX_TRAIL_TIME_FINE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Fine time counts of the trailing edge shot 3, low return pulses. If less than 10 returns collected then unused item slots are 0.

95. **SHOT3_RX_VALID_RETURN**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

Flag whether the shot 3 returns are valid. Valid returns are defined as having the same non-zero lead and trail ID value. =1 valid; =0 invalid

96. **SHOT4_RX_LEAD_ID**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

IDs for leading edge of shot 4, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

97. **SHOT4_RX_LEAD_TIME_COARSE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Coarse time counts of the leading edge shot 4, low return pulses. If less than 10 returns collected then unused item slots are 0.

98. **SHOT4_RX_LEAD_TIME_FINE**
    Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:2 Items: 10

Fine time counts of the leading edge shot 4, low return pulses. If less than 10 returns collected then unused item slots are 0.

99. **SHOT4_RX_TRAIL_ID**
    Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
    Item Bytes:1 Items: 10

IDs for trailing edge of shot 4, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

100. SHOT4_RX_TRAIL_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the trailing edge shot 4, low return pulses. If less than 10 returns collected then unused item slots are 0.

101. SHOT4_RX_TRAIL_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the trailing edge shot 4, low return pulses. If less than 10 returns collected then unused item slots are 0.

102. SHOT4_RX_VALID_RETURN
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

Flag whether the shot 4 returns are valid. Valid returns are defined as having the same non-zero lead and trail ID value. =1 valid; =0 invalid

103. SHOT5_RX_LEAD_ID
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

IDs for leading edge of shot 5, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

104. SHOT5_RX_LEAD_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the leading edge shot 5, low return pulses. If less than 10 returns collected then unused item slots are 0.

105. SHOT5_RX_LEAD_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the leading edge shot 5, low return pulses. If less than 10 returns collected then unused item slots are 0.

106. SHOT5_RX_TRAIL_ID
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

IDs for trailing edge of shot 5, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

107.SHOT5_RX_TRAIL_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the trailing edge shot 5, low return pulses. If less than 10 returns collected then unused item slots are 0.

108.SHOT5_RX_TRAIL_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the trailing edge shot 5, low return pulses. If less than 10 returns collected then unused item slots are 0.

109.SHOT5_RX_VALID_RETURN
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

Flag whether the shot 5 returns are valid. Valid returns are defined as having the same non-zero lead and trail ID value. =1 valid; =0 invalid

110.SHOT6_RX_LEAD_ID
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

IDs for leading edge of shot 6, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

111.SHOT6_RX_LEAD_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the leading edge shot 6, low return pulses. If less than 10 returns collected then unused item slots are 0.

112.SHOT6_RX_LEAD_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the leading edge shot 6, low return pulses. If less than 10 returns collected then unused item slots are 0.

113.SHOT6_RX_TRAIL_ID
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

IDs for trailing edge of shot 6, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

114.SHOT6_RX_TRAIL_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the trailing edge shot 6, low return pulses. If less than 10 returns collected then unused item slots are 0.

115.SHOT6_RX_TRAIL_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the trailing edge shot 6, low return pulses. If less than 10 returns collected then unused item slots are 0.

116.SHOT6_RX_VALID_RETURN
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

Flag whether the shot 6 returns are valid. Valid returns are defined as having the same non-zero lead and trail ID value. =1 valid; =0 invalid

117.SHOT7_RX_LEAD_ID
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

IDs for leading edge of shot 7, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

118.SHOT7_RX_LEAD_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the leading edge shot 7, low return pulses. If less than 10 returns collected then unused item slots are 0.

119.SHOT7_RX_LEAD_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the leading edge shot 7, low return pulses. If less than 10 returns collected then unused item slots are 0.

120.SHOT7_RX_TRAIL_ID
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

IDs for trailing edge of shot 7, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

121. SHOT7_RX_TRAIL_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the trailing edge shot 7, low return pulses. If less than 10 returns collected then unused item slots are 0.

122. SHOT7_RX_TRAIL_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the trailing edge shot 7, low return pulses. If less than 10 returns collected then unused item slots are 0.

123. SHOT7_RX_VALID_RETURN
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

Flag whether the shot 7 returns are valid. Valid returns are defined as having the same non-zero lead and trail ID value. =1 valid; =0 invalid

124. SHOT8_RX_LEAD_ID
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

IDs for leading edge of shot 8, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

125. SHOT8_RX_LEAD_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the leading edge shot 8, low return pulses. If less than 10 returns collected then unused item slots are 0.

126. SHOT8_RX_LEAD_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the leading edge shot 8, low return pulses. If less than 10 returns collected then unused item slots are 0.

127. SHOT8_RX_TRAIL_ID
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

IDs for trailing edge of shot 8, low return pulses, if less than 10 returns then unused item slots are 0.
=0 no id (used when less than 10 returns exist); =1 channel 1; =2 channel 2; =3 channel 3.

128. SHOT8_RX_TRAIL_TIME_COARSE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Coarse time counts of the trailing edge shot 8, low return pulses. If less than 10 returns collected then unused item slots are 0.

129. SHOT8_RX_TRAIL_TIME_FINE
Bytes:20 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:2 Items: 10

Fine time counts of the trailing edge shot 8, low return pulses. If less than 10 returns collected then unused item slots are 0.

130. SHOT8_RX_VALID_RETURN
Bytes:10 Data_Type: MSB_UNSIGNED_INTEGER
Item Bytes:1 Items: 10

Flag whether the shot 8 returns are valid. Valid returns are defined as having the same non-zero lead and trail ID value. =1 valid; =0 invalid

6.2.2 MLA Status EDR Binary Table

1. **MET**
Bytes: 4 Data_Type: MSB_UNSIGNED_INTEGER

Mission elapsed time in seconds.

2. **RMU_DATA_SIZE**
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU supports two interfaces in reading the RMU shot data after every RUPT: byte mode and nibble mode. Nibble mode not supported by flight software, therefore this telemetry point should always read 0 (byte mode).

3. **RMU_TEST_PATTERN**
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Test pattern used to fill RMU shot data.
=0,6,7: real data, =1: LFSR
=2: 0xAA, =3: 0x55
=4: Counter, =5: ~Counter

4. **RMU_DATA_TYPE**
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Identifies shot data as real or test data.
=0 real data, =1 test data.

5. **RMU_RATE_SELECT**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU Rate selected. =0, 1Hz; =1, 6Hz; =2, 8Hz; =3, 10Hz

6. **CANNED_DATA**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Designates whether flight software is using collection of RMU shot data or fake data as input to the science algorithms. =0 Real Data; =1 Fake Data

7. **SPARE**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Spare column.

8. **RMU_1PPS_SYNC**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU synchronized with 1PPS flag. =0 not synchronized; =1 synchronized

9. **RMU_TRANSFER_MODE**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Data Transfer mode commanded to RMU. Does not reflect the state of the RMU but rather the state of the software and how the software is reading the RMU. It is possible after an RMU reset to have this telemetry point not coincide with RMU_DATA_SIZE.
=0 byte mode; =1 low nibble; =2 high nibble

10. **DETECTOR_CH3**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Status of detector channel 3. =0 enabled; =1 disabled

11. **DETECTOR_CH2**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Status of detector channel 2. =0 enabled; =1 disabled

12. **DETECTOR_CH1LO**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Status of detector channel 1 LOW. =0 enabled; =1 disabled

13. **DETECTOR_CH1HI**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Status of detector channel 1 HIGH. =0 enabled; =1 disabled

14. **DUMP_ACTIVE**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Designates whether software is performing a memory dump.
=0 not performing memory dump
=1 currently performing memory dump

15. **RMU_CLOCK_SELECT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Clock configuration for the RMU. This drives the entire operation of the RMU. RMU has 4 different 5MHz clocks available to it, two internal to the RMU board and two external from the spacecraft.
=0 Oscillator A; =1 Internal 40%; =2 Internal 50%; =3 Oscillator B

16. **RMU_CYCLE_RESET_TOF**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU uses TOF chips to measure in fine detail the return pulses coming back from a laser fire. The chips are prone to lockup if flooded with input. If the flight software detects that they are locked up, it will automatically configure the RMU to reset the TOF chips for one shot. Unfortunately the rate at which this telemetry point is monitored means that it is very unlikely that this event will ever be reflected in this telemetry point.
=0 TOF chips are not reset after each shot.
=1 TOF chips are reset after each shot.

17. **RMU_CAL_SELECT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates which calibration setting was applied to the RMU during ground testing, if enabled:
=0 0ns, =1 200ns, =2 400ns, =3 200ns

18. **RMU_CAL_ENABLE**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether RMU will output a fixed test pattern during ground testing as selected by RMU_CAL_SELECT:
=0,1,2 Real Data from instrument; =3 TOF-A calibration data

19. **OVR_THRESHOLD_CH3**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether the science task has overridden (via command) the channel 3 threshold value written as a DAC to the hardware. =0 not overridden, =1 overridden

20. **OVR_THRESHOLD_CH2**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether the science task has overridden (via command) the channel 2 threshold value written as a DAC to the hardware. =0 not overridden, =1 overridden

21. **OVR_THRESHOLD_CH1LO**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether the science task has overridden (via command) the channel 1 LOW threshold value written as a DAC to the hardware. =0 not overridden, =1 overridden

22. **OVR_THRESHOLD_CH1HI**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether the science task has overridden (via command) the channel 1 HIGH threshold value written as a DAC to the hardware. =0 not overridden, =1 overridden

23. **OVR_CHAN_DISABLES**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether channel disables mask used by science algorithms has been overridden. This mask is used to enable and disable the return channels (1hi, 1low, 2, 3).
=0 not overridden, =1 overridden.

24. **OVR_GAIN**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether the science task has overridden (via command) the gain value written as a DAC to the hardware.
=0 not overridden, =1 overridden.

25. **OVR_RANGE_WINDOW**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether the science task has overridden (via command) the range window value used to write the RMU range gate start and stop.
=0 not overridden, =1 overridden.

26. **OVR_RANGE_DELAY**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether the science task has overridden (via command) the range delay value used to write the RMU range gate start and stop.
=0 not overridden, =1 overridden.

27. **V2DOT5_MON**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the +2.5 volt monitor.

28. **V5_MON**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the +5 volt monitor.

29. **V5NEG_MON**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the -5 volt monitor.

30. **V12_MON**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the +12 volt monitor.

31. **V32DOT5_MON**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the +32.5 volt monitor.

32. **V550_MON**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the +550 volt monitor.

33. **LASER_TX_THRESHOLD**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the laser Tx threshold that is commanded by the science task automatically every shot, via a DAC write, using a table value.

34. **AEM_LATCHUP_CTR**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Count of the number of times the flight software responds to an AEM latch-up. It is important to note that this counter does not reflect the number of AEM latch-ups, but rather the number of times the software responds to a latch-up. Because the AEM is not able to remove the latch-up condition fast enough, the software will attempt to acknowledge the same latch-up multiple times. Because of this, the software limits the number of latch-up acknowledgements a second to 2. In practice, for every AEM latch-up this counter will increment by two.

35. **DETECTOR_BOARD_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the detector board temperature.

36. **ALTIMETER_DET_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the altimeter detector temperature.

37. **ANALOG_BOARD_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the analog board temperature.

38. **RMU_BOARD_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the RMU board temperature.

39. **XTAL_OSCILLATOR_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the crystal oscillator temperature.

40. **CPU_BOARD_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the CPU board temperature.

41. **LASER_ELEC_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the laser electronics temperature.

42. **LASER_AMP_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the laser amplifier temperature.

43. **LASER_OSCILLATOR_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the laser oscillator temperature.

44. **PCA_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the PCA temperature.

45. **BEAM_XPAND_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the beam expander temperature.

46. **RX_TUBELENS_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the receiver tube lens temperature.

47. **RX_TUBE_BASE_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the Rx tube base temperature.

48. **MLA_HOUSING_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the MLA housing temperature.

49. **CAL_LO_TEMP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the low calibration temperature.

50. CAL_HI_TEMP
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ADC of the high calibration temperature.

51. TELEMETRY_VOLUME
Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Telemetry volume counter. Increments by 1 for every 1KB of data output by the telemetry framer task since the last software boot.

52. CHECKSUM_ENABLED
Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

An integer representation of the 16-bit mask that shows which tables in memory are being checksummed. Each bit in the mask corresponds to the table ID of the same number (bit 4 in the mask corresponds to table 4). Convert back to binary notation before evaluating the checksumenabled mask:
=0 table is not being checksummed, =1 table is being checksummed.

53. CHECKSUM_STATUS
Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

An integer representation of the 16-bit mask that shows whether the checksum of a table in memory is correct or incorrect. Each bit in the mask corresponds to the table ID of the same number (bit 4 in the mask corresponds to table 4). Convert back to binary notation before evaluating the checksumstatus mask:
=0 checksum is correct, =1 checksum is incorrect

54. TOF_LOCKUP_CNTR
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

There are 6 TOF chips in the RMU. The RMU uses them to timestamp the Tx pulse, the high return, and the low returns. Every 16 shots the software analyzes the state of the TOFs. If during the past 16 shots the fine time (least significant 10 bits) on a RMU time stamp has not changed at all, and the coarse time has changed at least once, then a TOF lockup is detected and this telemetry point is incremented, and for the next shot, the TOF chips are configured to be reset. The software will immediately begin monitoring for the next 16 shots. The software only monitors the TOF lockups in Standby and Science modes.

55. TELEM_RATE_CONFIG
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Indicates whether the TELEMETRY_VOLUME counter is derived from the FSW or from a test source:
=0 Oper, =1 Test

56. SDHWDIAG_LITESWITCH
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

=0 The MLA_HWDiagLite packet is off.
=1 The MLA_HwDiagLite packet is on.

57. SI_TIMING_VALID
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Check of timing integrity at rate of once per second. Three checks are performed:
1. Has MET message from S/C been received?
2. Has the 1PPS signal from the S/C been received?
3. Is the phase lock to the 1PPS off by more than 5 microseconds?

If any of the checks fail then the timing integrity is in question.
=0 Timing integrity fine, =1 Timing integrity under question.

58. CHECKSUM_EN_FLAG
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

=0 checksumming of tables is disabled.
=1 checksumming of tables is enabled.

59. ONE_PPS_OCCURED
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Each second the spacecraft sends a 1PPS signal to the MLA instrument. The flight software uses this signal to synchronize.
=0 1PPS did not occur, =1 1PPS occurred.

60. SDHWDIAG_FULLSWITCH
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

=0 MLA_HwDiagnostic packet is off.
=1 MLA_HwDiagnostic packet is on.

61. OS_EPROM_VERSION
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

There are two banks of EEPROM that the software is stored in. When the instrument is turned on, or any time the CPU board is rebooted, the code stored in EEPROM is copied out of one of the two banks, into RAM and executed there. This telemetry point says which bank of EEPROM the code was copied out of.
=0 EEPROM Bank 0 - Non Write-Protected
=1 EEPROM Bank 1 - Write-Protected

62. SIDIRECT_THRES_OVR
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The science task allows the thresholds applied to each of the channels to be overridden. The science algorithms specify that the value commanded for them to be overridden with must be scaled by a function of the gain, and range window. For testing purposes, it was necessary to have the ability to directly apply the threshold given in the threshold override command.
=0 thresholds are scaled as prescribed in science algorithm.
=1 thresholds are directly applied

63. IDLE_CHECKIN
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

By default, the flight software requires certain critical tasks to 'check in' in order for the watchdog to be kicked. In diagnosing a problem it might be necessary to disable that behavior.
=0 critical tasks do NOT have to check in.
=1 critical tasks DO have to check in.

64. IDLE_MET_CHECKIN
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

By default, the software checks that the MET is being received while in Science mode. If the MET is missing for a preset amount of time, then the software will execute the Mla_Safe script that will transition the software to Keep Alive mode. This telemetry point reflects whether the software is actively checking if the MET is being received.
=0 MET not being checked.
=1 MET being checked.

65. SISC_TIME_BIAS
Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

The difference between the s/c ranging data alignment to the value of the MET, used in Science algorithm mode 1. Value is a B8 fraction.

66. SISC_RANGE_BIAS
Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

The difference between the s/c ranging data alignment to the range value, used in Science algorithm mode 1 to account for orbital error and other biases. Value is a signed integer number of counts.

67. SI_TRANSMIT_THRESHOLD
Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

The transmit start detector threshold value (currently defaults to 15 counts).

68. SD_AEM_ERROR_COUNT
Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Once a second, the flight software performs a series of ADCs on the temperatures and voltages. After every RUPT, the flight software performs a series of ADCs on all the items that can be written to via a DAC. If any ADC fails, due to a time out (the software depends on the AEM asserting that it is finished the ADC), then this counter is incremented.

69. SI_RMU_ERROR_COUNT
Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Every shot when the software is in Science mode, the science task writes the values of the range gates (start and stop). At that time it also reads the values back and compares them to the values it wrote. If those two values do not agree then the science task increments this counter.

70. **CMD_ACCEPT_CTR**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Counter increments every time a command is sent and the software executes that command to completion without any errors.

71. **CMD_OPCODE**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The opcode of the last command that was processed.

72. **CMD_RESULTCODE**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The error code generated by the last command executed. If the last command executed did not generate an error, then this value is zero.

73. **ALARM_ID**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The ID of the last alarm issued by the flight software.

74. **ALARM_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Counter is incremented every time an alarm is issued by the flight software. The flight software treats the reset message as an alarm, therefore value is one at boot up. However the actual integer range of values will be from 129 to 255 because the most significant bit of the telemetry point is used as a flag that is always set to 1.

75. **CMD_REJECT_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Counter increments every time the flight software fails to execute a command.

76. **ITF_REJECT_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Counter is incremented every time the flight software receives an ITF and the ITF is either invalid or unable to be processed. The four cases for which this counter increments are:

Not enough memory available to allocate a software bus packet;
Invalid command format;
Incorrect checksum;
Synch pattern incorrect.

77. **WATCHDOG_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Increments on every reset of the flight software watchdog.

78. CPURESET_COUNT
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Increments on every reset of the flight software CPU. A CPU reset can occur one of two ways:
Software was directly commanded to reset;
Software erroneously branched to an unused memory address and executed a 0xFF instruction.

79. RESET_CAUSE
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Every time the flight software reboots, registers in the CPU board FPGA are examined in order to determine the cause of the reboot:
=0 Power On, =1 CPU Reset, =2 Watchdog, =3 Invalid.

80. MLA_MODE
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Flight software mode: =0 Keep Alive, =1 Standby, =2 Science

81. DPU_SELECTION
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The S/C has two DPUs that can interface to the MLA instrument. Only one of them can be active at a time.
=0 DPU A, =1 DPU B

82. WATCHDOG_ENABLED
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

=1 Watchdog enabled.
=0 Watchdog disabled. Flight software will still continue to service the watchdog but it will have no effect, nor will there be any effect if the software fails to service the watchdog.

83. PCA_POWER_MODE
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The PCA controls the power on the MLA instrument. There are two power modes. Under normal operation, Keep Alive software mode configures the PCA to low power mode. Standby and Science software modes configure the PCA to high power mode.
=0 low power mode, =1 high power mode.

84. PCA_LASER_ENABLED
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The laser on the MLA instrument can be directly turned on and off. Under normal operations the laser is automatically configured to be on in Science mode and automatically configured to be off in all other modes.
=0 laser is OFF, =1 laser is ON.

85. ALGORITHM_MODE
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The currently designated science algorithm mode. Only modes 0 and 1 were implemented in FSW at launch.
=0 Mode 0 , =1 Mode 1 , =2 Closed Loop.

86. RANGE_MODE
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

The currently designated range tracking mode:
0= Don't use the s/c range data
1= Use s/c range in tracking algorithm
2= Reserved for future use in Algorithm Mode 2.

87. HWDIAG_TLM_CONFIG
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

=0 neither MLA_HwDiagnostic nor MLA_HwDiagLite is on.
=1, one of the hardware diagnostic packets is on.

88. SWDIAG_TLM_CONFIG
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

=0 The MLA_SwDiagnostic packet is OFF
=1 The MLA_SwDiagnostic packet is ON

89. MET_FREE_RUN
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Every second the flight software checks to see if it has received a MET message from the spacecraft. If it did not receive the MET message it will free run off an internal clock until the MET message is received again. When using GSEOS build 4 or lower, the MET message is not consistently issued, and therefore this telemetry point will show the software free running often; this is fine. There is one known error condition for this telemetry point, and that is when the science task overruns and delays the telemetry framer task for a considerable period of time (over 50ms). It is possible in that case that the MET was received, but because the task that monitors the MET was delayed, the software believes it is free running and will flag this
telemetry point.
=0 the flight software did receive a MET message
=1 the flight software did NOT receive an MET message

90. MEM_WRITE_ENABLE
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

=0 memory is not write enabled
=1 memory is write enabled.

91. SD_PARITY_ERROR
Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Every time the RUPT is generated, the flight software reads 166 bytes from the RMU (the shot data). The last byte of the data read from the RMU is a parity byte (XOR over entire block). The flight software also computes a parity for the data read from the RMU. If the two parities do not equal each other this telemetry point increments. This value should never be anything but 0.

6.2.3 MLA Hardware Diagnostic Lite EDR Binary Table

1. **MET**
   Bytes: 4 Data_Type: MSB_UNSIGNED_INTEGER

Mission elapsed time in seconds.

2. **DATA_XFER_MODE**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Science Data Transfer Mode. =0 byte, =1 nibble.

3. **TEST_PATTERN**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Test Pattern Selection
=1 LFSR, =2 AA, =3 55, =4 Counter, =5 Counter Inverse

4. **TEST_DATA_SELECT**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Test data select.
=0 Real data, =1 Test Data pattern. Defaults to 1.

5. **RATE_SELECT**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Rate select flag.
=0 One Hz, =1 Six Hz, =2 Eight Hz, =3 Ten Hz.

6. **MODEWORD1_SPARE**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Spare column reserved for future use.

7. **HWLIGHT_SPAREBYTE**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Spare column reserved for future use.

8. **STATUS3SPARE6_7**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Spare column reserved for future use.

9. **CYCLE_RESET_TOF**
   Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Reset Time of Flight timing chips. With reset enabled the RMU will automatically reset the TOF after every RUPT until configured to do otherwise.

10. **CAL_SELECT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Calibration mode selected.
=0 0ns, =1 200ns, =2 400ns, =3 200ns.

11. **CAL_ENABLE**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Enable calibration.
=0,1,2 Real Data. =3 TOF-A calibration data.

12. **CLOCK_SELECT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Clock selected.
=0 Internal (40%), =1 OSC_B (external clock), =2 OSC_A (external clock), =3 Internal (50%)

13. **STATUS2SPARE2_4**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Spare column reserved for future use.

14. **SYNCH_TO_PPS**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Synchronize the RMU to the 1PPS. =0 inhibited, =1 enabled

15. **CYCLE_SLIP**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

CPU Cycle slip. =0 no slip, =1 slip

16. **CH1_NOISE_HI**
    Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Background noise counter for channel 1 HIGH.

17. **CH1_NOISE_LO**
    Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Background noise counter for channel 1 LOW.

18. **CH2_NOISE**
    Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Background noise counter for channel 2.

19. **CH3_NOISE**
    Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Background noise counter for channel 3.

20. **START_PULSE_BEGIN**
    Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the beginning edge of the start pulse.

Start Bit	Num Bits	Bit Column Name	Description
1	1	START_PULSE_BEGIN_VALID	=1 if pulse is valid, =0 if invalid
2	3	START_PULSE_BEGIN_PID	Pulse ID for start pulse is always 0.

5	18	START_PULSE_BEGIN_COARSE	RMU coarse counter, 200 ns ticks
23	10	START_PULSE_BEGIN_FINE	RMU fine counter, ~400 ps ticks

21. START_PULSE_END
Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the trailing edge of the start pulse.

Start Bit	Num Bits	Bit Column Name	Description
1	1	START_PULSE_END_VALID	=1 if pulse is valid, =0 if invalid
2	3	START_PULSE_END_PID	Pulse ID for start pulse is always 0.
5	18	START_PULSE_END_COARSE	RMU coarse counter, 200 ns ticks
23	10	START_PULSE_END_FINE	RMU fine counter, ~400 ps ticks

22. CH1_HI_PULSE_BEGIN
Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the beginning edge of the channel 1 high pulse.

Start Bit	Num Bits	Bit Column Name	Description
1	1	CH1_HI_PULSE_BEGIN_VALID	=1 if pulse is valid, =0 if invalid
2	3	CH1_HI_PULSE_BEGIN_PID	Identifies pulse channel. =1 Channel 1, =2 channel 2, =3 Channel 3.
5	18	CH1_HI_PULSE_BEGIN_COARSE	RMU coarse counter, 200 ns ticks
23	10	CH1_HI_PULSE_BEGIN_FINE	RMU fine counter, ~400 ps ticks

23. CH1_HI_PULSE_END
Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the trailing edge of the channel 1 high pulse.

Start Bit	Num Bits	Bit Column Name	Description
1	1	CH1_HI_PULSE_END_VALID	=1 if pulse is valid, =0 if invalid
2	3	CH1_HI_PULSE_END_PID	Identifies pulse channel. =1 Channel 1, =2 channel 2, =3 Channel 3.
5	18	CH1_HI_PULSE_END_COARSE	RMU coarse counter, 200 ns ticks
23	10	CH1_HI_PULSE_END_FINE	RMU fine counter, ~400 ps ticks

24. LOWPULSE_1_BEGIN
Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the beginning edge of the low pulse 1.

Start Bit	Num Bits	Bit Column Name	Description
1	1	LOWPULSE_1_BEGIN_VALID	=1 if pulse is valid, =0 if invalid
2	3	LOWPULSE_1_BEGIN_PID	Identifies pulse channel. =1 Channel 1, =2 channel 2, =3 Channel 3.
5	18	LOWPULSE_1_BEGIN_COARSE	RMU coarse counter, 200 ns ticks
23	10	LOWPULSE_1_BEGIN_FINE	RMU fine counter, ~400 ps ticks

25. LOWPULSE_1_END
Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the trailing edge of the low pulse 1.

Start Bit	Num Bits	Bit Column Name	Description
1	1	LOWPULSE_1_END_VALID	=1 if pulse is valid, =0 if invalid
2	3	LOWPULSE_1_END_PID	Identifies pulse channel. =1 Channel 1, =2 channel 2, =3 Channel 3.
5	18	LOWPULSE_1_END_COARSE	RMU coarse counter, 200 ns ticks
23	10	LOWPULSE_1_END_FINE	RMU fine counter, ~400 ps ticks

26. LOWPULSE_2_BEGIN
Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the beginning edge of the low pulse 2.

Start Bit	Num Bits	Bit Column Name	Description
1	1	LOWPULSE_2_BEGIN_VALID	=1 if pulse is valid, =0 if invalid
2	3	LOWPULSE_2_BEGIN_PID	Identifies pulse channel. =1 Channel 1, =2 channel 2, =3 Channel 3.
5	18	LOWPULSE_2_BEGIN_COARSE	RMU coarse counter, 200 ns ticks
23	10	LOWPULSE_2_BEGIN_FINE	RMU fine counter, ~400 ps ticks

27. LOWPULSE_2_END
Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the trailing edge of the low pulse 2.

Start Bit	Num Bits	Bit Column Name	Description
1	1	LOWPULSE_2_END_VALID	=1 if pulse is valid, =0 if invalid
2	3	LOWPULSE_2_END_PID	Identifies pulse channel. =1 Channel 1, =2 channel 2, =3 Channel 3.
5	18	LOWPULSE_2_END_COARSE	RMU coarse counter, 200 ns ticks
23	10	LOWPULSE_2_END_FINE	RMU fine counter, ~400 ps ticks

28. LOWPULSE_3_BEGIN
Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the beginning edge of the low pulse 3.

Start Bit	Num Bits	Bit Column Name	Description
1	1	LOWPULSE_3_BEGIN_VALID	=1 if pulse is valid, =0 if invalid
2	3	LOWPULSE_3_BEGIN_PID	Identifies pulse channel. =1 Channel 1, =2 channel 2, =3 Channel 3.
5	18	LOWPULSE_3_BEGIN_COARSE	RMU coarse counter, 200 ns ticks
23	10	LOWPULSE_3_BEGIN_FINE	RMU fine counter, ~400 ps ticks

29. **LOWPULSE_3_END**
    Bytes: 4 Data_Type: MSB_BIT_STRING

Contains the timing and validity bits for the trailing edge of the low pulse 3.

Start Bit	Num Bits	Bit Column Name	Description
1	1	LOWPULSE_3_END_VALID	=1 if pulse is valid, =0 if invalid
2	3	LOWPULSE_3_END_PID	Identifies pulse channel. =1 Channel 1, =2 channel 2, =4 Channel 3.
5	18	LOWPULSE_3_END_COARSE	RMU coarse counter, 200 ns ticks
23	10	LOWPULSE_3_END_FINE	RMU fine counter, ~400 ps ticks

30. **RANGE_GATE_START**
    Bytes: 4 Data_Type: MSB_UNSIGNED_INTEGER

Range Gate start time in units of 200 ns.

31. **RANGE_GATE_STOP**
    Bytes: 4 Data_Type: MSB_UNSIGNED_INTEGER

Range Gate stop time in units of 200 ns.

32. **CH1_BEGIN_EVENT_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU pulse begin triggers on Ch. 1

33. **CH1_END_EVENT_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU pulse end triggers on Ch. 1

34. **CH2_BEGIN_EVENT_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU pulse begin triggers on Ch. 2

35. **CH2_END_EVENT_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU pulse end triggers on Ch. 2

36. **CH3_BEGIN_EVENT_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU pulse begin triggers on Ch. 3

37. **CH3_END_EVENT_COUNT**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

RMU pulse end triggers on Ch. 3

38. **LASER_TX_THRESHOLD**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Laser start detector threshold (counts).

39. **DETECTOR_GAIN**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Detector VGA setting (counts).

40. **DET_CH1_HI_THRESHOLD**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

AEM readback Ch. 0 threshold setting.

41. **DET_CH1_LO_THRESHOLD**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

AEM readback Ch. 1 threshold setting.

42. **DET_CH2_THRESHOLD**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

AEM readback Ch. 2 threshold setting.

43. **DET_CH3_THRESHOLD**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

AEM readback Ch. 3 threshold setting.

44. **TX_PULSE_ENERGY**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Laser pickoff energy monitor.

45. **LASER_DIODE_CURR**
    Bytes: 1 Data_Type: MSB_UNSIGNED_INTEGER

Laser diode pump current.

46. **LASER_PULSE_WIDTH**
    Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Laser pulse width (counts).

47. **HZ_TO_RUPT**
    Bytes: 4 Data_Type: MSB_UNSIGNED_INTEGER

8 MHz Ticks from 1PPS signal to T_0.

48. **SUBSECONDS**
    Bytes: 2 Data_Type: MSB_UNSIGNED_INTEGER

Fraction of second from 1PPS to Packet, 0-65536.

6.3 Label and Header Descriptions

The following are the keyword definitions for the detached PDS label file accompanying the binary data file. The detached PDS label file has the same name as the data file it describes, except for the extension .LBL to distinguish it as a label file.

PDS_VERSION_ID
Represents the version number of the PDS standards documents that is valid when a data product label is created. PDS3 is used for the MESSENGER data products.

FILE_RECORDS
Indicates the number of physical file records, including both label records and data records.

RECORD_TYPE
Indicates the record format of a file. Storing the data in a binary table object necessitates using a FIXED LENGTH record type.

RECORD_BYTES
Indicates the number of bytes in a physical file record, including record terminators and separators.

PRODUCT_ID
Represents a permanent, unique identifier assigned to a data product by its producer. Details of the PRODUCT_ID naming convention still have yet to be worked out by the science team in conjunction with the PDS.

PRODUCT_VERSION_ID
Identifies the version of the product, represented initially by "V1". The version number will be incremented if the product needs to be regenerated as a result of a correction of the data or as a result of a change in the method used to create the product.

PRODUCT_CREATION_TIME
Defines the UTC system format time when a product was created.

PRODUCT_TYPE
Identifies whether the EDR contains science data ("DATA") or ancillary engineering data ("ANCILLARY").

STANDARD_DATA_PRODUCT_ID
The MLA EDR files are identified as belonging to the same standard data product via this ID.

SOFTWARE_NAME
Identifies the data processing software used to convert from spacecraft telemetry into EDR products. For MLA it is the "PIPE-MLA2EDR" software engine.

SOFTWARE_VERSION_ID
Indicates the version (development level) of the program defined in SOFTWARE_NAME. It will be incremented if the program is modified during the course of the mission.

INSTRUMENT_HOST_NAME
The full name of the host on which an instrument is based. For MESSENGER this is the "MERCURY SURFACE, SPACE ENVIRONMENT, GEOCHEMISTRY, AND RANGING" satellite. However, since the project commonly uses the acronym for the satellite, it is simply MESSENGER.

INSTRUMENT_NAME
The "MERCURY LASER ALTIMETER".

INSTRUMENT_ID
Provides an abbreviated name or acronym that identifies an instrument. In this case "MLA".

DATA_SET_ID
The data_set_id for the MLA data products. There is only 1 data set id for the entire MLA EDR archive.

MISSION_PHASE_NAME
Identifies the project designation of the mission phase associated with the data product.

TARGET_NAME
Identifies a target. The possible targets for the MLA are Earth, Venus, Mercury, and Calibration. Although this may change during the course of the mission, and as mission planning refines their target list.

START_TIME
Provides the date and time corresponding to the MET of the first record in the binary table.

STOP_TIME
Provides the date and time corresponding to the MET of the last record in the binary table.

SPACECRAFT_CLOCK_START_COUNT
The value of the spacecraft clock corresponding to the MET of the first record in the binary table.

SPACECRAFT_CLOCK_STOP_COUNT
The value of the spacecraft clock corresponding to the MET of the last record in the binary table.

^TABLE
This is a pointer to the external data file containing the binary table. The TABLE object is a uniform collection of rows containing binary values stored in columns.
Note: The MLA data is stored in binary format, hence the value of the keyword interchange_format is BINARY, and the table uses FIXED_LENGTH records.

6.4 File Naming Conventions

The file names developed for PDS data volumes are restricted to a maximum 36 character file name and a 3 character extension name with a period separating the file and extension names. The general form of the MLA data product files is "MLAXXXYYMMDDHHNN". All binary table data files will end in ".DAT" and PDS label files will end in ".LBL". The components of the base filename are described below. Note that references to time units in the filename refer to UTC time. The UTC time used in the filename corresponds to the first record in the binary table. Recall that the data files group observations in two different methods, depending on whether the data is gathered during the Mercury Orbit phase or prior to Mercury orbit (section 5.2).

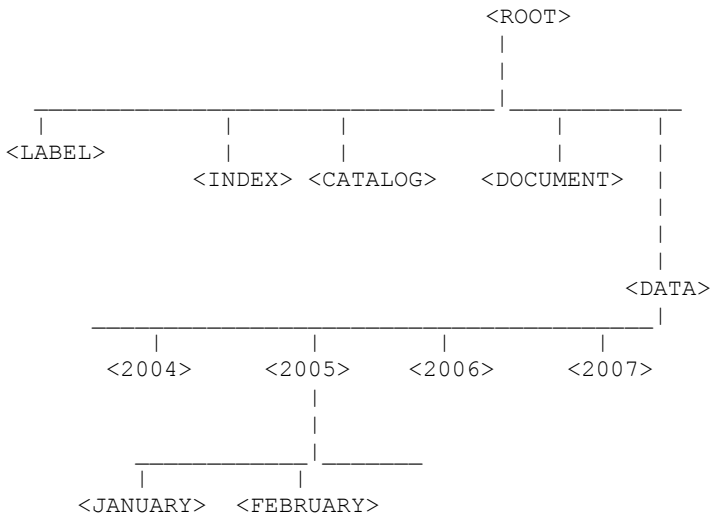
MLA : Instrument name – Mercury Laser Altimeter
XXX : Data product type:
RAW – Raw science data
STA – Status data
HAD – Hardware Diagnostic Lite.
YY : The last two digits of the year in which the first data record was acquired.
MM : The two digit month in which the first data record was acquired.
DD : The two digit day in which the first data record was acquired.
HH : The two digit hour corresponding to the first record in the data.
NN : The two digit minute corresponding to the first record in the data.

Each PDS label file contains a ^STRUCTURE pointer to an external file. This external file will contain the definition of the binary table object. Due to the large number of columns in the table, and since the structure of the binary table does not change, it makes sense to define the structure once in an external file and include a pointer to it in every PDS detached label file.

6.5 Directory Structure and Contents for MLA EDR Data Archive Volume

The following illustration shows the directory structure overview for the MLA EDR Data Archive Volume. This volume contains the MLA EDR data products, supporting documentation, and any additional files required for the volume to be compliant with PDS standards. The content of the volume is expected to be updated with periodic releases according to the schedule in the *MESSENGER Data Management and Archiving Plan*. Revised EDRs (if needed) will also be delivered according to the same schedule. Revised EDRs will have an incremented version number in the PDS label.

Directory Structure Overview



6.5.1 Directory Contents

<ROOT> Directory

This is the top-level directory of the data volume. The following are files contained in the root directory.

AAREADME.TXT - General information file. Provides users with information about the MESSENGER MLA data products. Directs user to other documents on the volume containing more detailed information.

VOLDESC.CAT - PDS file containing the VOLUME object. This gives a high level description of the contents of the volume. Information includes: production date, producer name and institution, volume ID, etc.

ERRATA.TXT - Text file for identifying and describing errors and/or anomalies found in the current volume. Any known errors for the associated volume will be documented in this file.

<DOCUMENT> Directory

This subdirectory contains the documentation that will be needed in order to understand and analyze the EDR data products. Multiple copies of each document will be stored, each one in a different format. Files will be stored in PDF and ASCII format.

DOCINFO.TXT – Identifies and describes the function of each file in the DOCUMENT directory

<CATALOG> Directory

This subdirectory contains the catalog object files for the entire volume. The following files are included in the catalog subdirectory.

CATINFO.TXT: Identifies and describes of function of each file in the catalog directory**.**

DATASET.CAT: Describes the general content of the dataset and includes information about the duration of the mission and the person or group responsible for producing the data.

INSTRUMENT.CAT: Describes physical attributes of the MLA instrument and provides relevant references to published literature.

INSTRUMENT_HOST.CAT: Describes the MESSENGER spacecraft.

MISSION.CAT: Describes the scientific goals and objectives of the MESSENGER program. It also identifies key people and institutions.

REF.CAT: Contains the reference objects. These reference additional documents that may be useful to the person using the MLA EDR.

<INDEX> Directory

This subdirectory contains the indices for the MLA EDR data products. The following files are contained in the index subdirectory.

INDEXINFO.TXT – Identifies and describes the function of each file in the index subdirectory. This includes a description of the structure and contents of each index table in the subdirectory AND usage notes.

INDEX.TAB - The EDR index file is organized as a table; there is one entry for each of the data files included in the MLA data set; the columns contain parameters that describe the data as well as instrument and spacecraft parameters.

INDEX.LBL - Detached PDS label for INDEX.TAB.

<LABEL> - Label Directory

This subdirectory contains the format files for each of the different EDR product types. The format file describes the format of the Binary Table object in the EDR data file.

<DATA> - Data Directory

This is the top level of the directory in which the EDRs are contained. The data directories are organized by year (YYYY) followed by month in UTC. The EDRs reside in the month folders.

7 Archive Release Schedule to PDS

The MESSENGER MLA EDR archive will be transferred from the SOC to the PDS Geosciences Node using the electronic transfer process detailed in section 5.3.3. The transfer will take place according to the release schedule in the *MESSENGER Data Management and Archiving Plan*.

8 Appendices

8.1 Appendix - MLA Science Raw PDS Label

```
PDS_VERSION_ID                = "PDS3"

/*** FILE FORMAT ***/
FILE_RECORDS                  = 1178
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 1076

/*** GENERAL DATA DESCRIPTION PARAMETERS ***/
PRODUCT_ID                    = "MLASCI0505111310_DAT"
PRODUCT_VERSION_ID            = "V1"
PRODUCT_CREATION_TIME         = 2006-09-28T20:32:12
PRODUCT_TYPE                  = "DATA"
STANDARD_DATA_PRODUCT_ID      = "MLASCIENCERAW"
SOFTWARE_NAME                 = "PIPE-MLA2EDR"
SOFTWARE_VERSION_ID           = "1.0"
INSTRUMENT_HOST_NAME          = "MESSENGER"
INSTRUMENT_NAME               = "MERCURY LASER ALTIMETER"
INSTRUMENT_ID                 = "MLA"
DATA_SET_ID                   = "MESS-E/V/H-MLA-2-EDR-RAWDATA-V1.0"
MISSION_PHASE_NAME            = "EARTH CRUISE"
TARGET_NAME                   = "CALIBRATION"
START_TIME                    = 2005-05-11 13:10:15.000
STOP_TIME                     = 2005-05-11 13:29:52.000
SPACECRAFT_CLOCK_START_COUNT  = 24304159
SPACECRAFT_CLOCK_STOP_COUNT   = 24305336
^TABLE                        = "MLASCI0505111310.DAT"

OBJECT                        = TABLE
COLUMNS                       = 130
INTERCHANGE_FORMAT            = BINARY
ROW_BYTES                     = 1076
ROWS                          = 1178
DESCRIPTION                   = "
    This table contains one set of MLA Raw Science values, as reported
    by the MESSENGER Mercury Laser Altimeter (MLA).
    The complete column definitions are contained in an structure file
   found in the LABEL directory of the archive disk. Additional details are
```

```
    contained in the EDR SIS document."
^STRUCTURE  = "MLARAW.FMT"
END_OBJECT                    = TABLE
END
```

8.2 Appendix - MLA Status Raw PDS Label

```
PDS_VERSION_ID                = "PDS3"

/*** FILE FORMAT ***/
FILE_RECORDS                  = 6
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 102

/*** GENERAL DATA DESCRIPTION PARAMETERS ***/
PRODUCT_ID                    = "MLASTA0505110001_DAT"
PRODUCT_VERSION_ID            = "V1"
PRODUCT_CREATION_TIME         = 2006-09-28T20:28:21
PRODUCT_TYPE                  = "ANCILLARY"
STANDARD_DATA_PRODUCT_ID      = "MLASTATUS"
SOFTWARE_NAME                 = "PIPE-MLA2EDR"
SOFTWARE_VERSION_ID           = "1.0"
INSTRUMENT_HOST_NAME          = "MESSENGER"
INSTRUMENT_NAME               = "MERCURY LASER ALTIMETER"
INSTRUMENT_ID                 = "MLA"
DATA_SET_ID                   = "MESS-E/V/H-MLA-2-EDR-RAWDATA-V1.0"
MISSION_PHASE_NAME            = "EARTH CRUISE"
TARGET_NAME                   = "CALIBRATION"
START_TIME                    = 2005-05-11 00:01:11.000
STOP_TIME                     = 2005-05-11 00:51:11.000
SPACECRAFT_CLOCK_START_COUNT  = 24256815
SPACECRAFT_CLOCK_STOP_COUNT   = 24259815
^TABLE                        = "MLASTA0505110001.DAT"

OBJECT                        = TABLE
COLUMNS                       = 91
INTERCHANGE_FORMAT            = BINARY
ROW_BYTES                     = 102
ROWS                          = 6
DESCRIPTION                   = "
    This table contains one set of instrument engineering data as reported by
    the MESSENGER Mercury Laser Altimeter (MLA) status packets.
    The complete column definitions are contained in an structure file
   found in the LABEL directory of the archive disk. Additional details are
    contained in the EDR SIS document."
^STRUCTURE  = "MLASTA.FMT"
END_OBJECT                    = TABLE
END
```

8.3 Appendix - MLA Hardware Diagnostic Lite PDS Label

```
PDS_VERSION_ID                = "PDS3"

/*** FILE FORMAT ***/
FILE_RECORDS                  = 45000
RECORD_TYPE                   = FIXED_LENGTH
RECORD_BYTES                  = 96

/*** GENERAL DATA DESCRIPTION PARAMETERS ***/
PRODUCT_ID                    = "MLAHAD0408191912_DAT"
PRODUCT_VERSION_ID            = "V1"
PRODUCT_CREATION_TIME         = 2006-09-29T01:19:13
PRODUCT_TYPE                  = "ANCILLARY"
STANDARD_DATA_PRODUCT_ID      = "MLAHDIAGNOSTIC"
SOFTWARE_NAME                 = "PIPE-MLA2EDR"
SOFTWARE_VERSION_ID           = "1.0"
INSTRUMENT_HOST_NAME          = "MESSENGER"
```

```
INSTRUMENT_NAME                = "MERCURY LASER ALTIMETER"
INSTRUMENT_ID                  = "MLA"
DATA_SET_ID                    = "MESS-E/V/H-MLA-2-EDR-RAWDATA-V1.0"
MISSION_PHASE_NAME             = "LAUNCH"
TARGET_NAME                    = "CALIBRATION"
START_TIME                     = 2004-08-19 19:12:56.000
STOP_TIME                      = 2004-08-19 19:59:59.000
SPACECRAFT_CLOCK_START_COUNT   = 1430009
SPACECRAFT_CLOCK_STOP_COUNT    = 1432832
^TABLE                         = "MLAHAD0408191912.DAT"

OBJECT                         = TABLE
COLUMNS                        = 48
INTERCHANGE_FORMAT             = BINARY
ROW_BYTES                      = 96
ROWS                           = 45000
DESCRIPTION                    = "
    This table contains one set of instrument engineering data as reported by
    the MESSENGER Mercury Laser Altimeter (MLA) hardware diagnostic lite
    packets.
    The complete column definitions are contained in an structure file
   found in the LABEL directory of the archive disk. Additional details are
    contained in the EDR SIS document. "
^STRUCTURE  = "MLAHAD.FMT"
END_OBJECT                     = TABLE
END
```

8.4 Appendix - SPICE Kernel Files Used in MESSENGER Data Products

The following SPICE kernel files will be used to compute the UTC time and any geometric quantities found in the PDS labels. Kernel files will be generated throughout the mission with a file-naming convention specified by the MESSENGER project.

***.bsp:**
MESSENGER spacecraft ephemeris file. Also known as the Planetary Spacecraft Ephemeris Kernel (SPK) file.

***.bc:**
MESSENGER spacecraft orientation file. Also known as the Attitude C-Kernel (CK) file.

***.tf:**
MESSENGER reference frame file. Also known as the Frames Kernel. Contains the MESSENGER spacecraft, science instrument, and communications antennae frame definitions.

***.ti:**
MESSENGER instrument kernel (I-kernel). Contains references to mounting alignment, operation modes, and timing as well as internal and field of view geometry for the MLA instrument.

***.tsc:**
MESSENGER spacecraft clock coefficients file. Also known as the Spacecraft Clock Kernel (SCLK) file.

***.tpc:**
Planetary constants file. Also known as the Planetary Constants Kernel (PcK) file.

***.tls:**
NAIF leapseconds kernel file. Used in conjunction with the SCLK kernel to convert between Universal Time Coordinated (UTC) and MESSENGER Mission Elapsed Time (MET). Also called the Leap Seconds Kernel (LSK) file.

8.5 Appendix - Data Archive Terms

Definition of Terms:

Archive	An archive consists of one or more data sets along with all the documentation and ancillary information needed to understand and use the data. An archive is a logical construct independent of the medium on which it is stored.
Archive Volume, Archive Volume Set	A volume is a unit of medium on which data products are stored; for example, one DVD. An archive volume is a volume containing all or part of an archive, that is, data products plus documentation and ancillary files. When an archive spans multiple volumes, they are called an archive volume set. Usually the documentation and some ancillary files are repeated on each volume of the set, so that a single volume can be used alone.
Data Product	A labeled grouping of data resulting from a scientific observation, usually stored in one file. A product label identifies, describes, and defines the structure of the data. An example of a data product is a planetary image, a spectrum table, or a time series table.
Data Set	An accumulation of data products. A data set together with supporting documentation and ancillary files is an archive.
Experiment Data Records	NASA Level 0 data for a given instrument; raw data. Same as CODMAC level2.
Reduced Data Records	Science data that have been processed from raw data to NASA Level 1 or higher. See section 8.6 for definitions of processing levels.
Standard Data Product	A data product that has been defined during the proposal and selection process and that is contractually promised by the PI as part of the investigation. Standard data products are generated in a predefined way, using well-understood procedures, and processed in "pipeline" fashion.

8.6 Appendix - CODMAC and NASA Data Levels

CODMAC/NASA Definition of processing levels for science data sets

CODMAC Level	Proc. Type	Data Processing Level Description
1	Raw Data	Telemetry data stream as received at the ground station, with science and engineering data embedded. Corresponds to NASA packet data.
2	Edited Data	Instrument science data (e.g., raw voltages, counts) at full resolution, time ordered, with duplicates and transmission errors removed. Referred to in the MESSENGER program as Experiment Data Records (EDRs). Corresponds to NASA Level 0 data.
3	Calibrated Data	Edited data that are still in units produced by the instrument, but have been transformed (e.g., calibrated, rearranged) in a reversible manner and packaged with needed ancillary and auxiliary data (e.g., radiances with calibration equations applied). Referred to in the MESSENGER Program as Calibrated Data Records (CDRs). In some cases these also qualify as derived data products (DDRs). Corresponds to NASA Level 1A.
4	Resampled data	Irreversibly transformed (e.g., resampled, remapped, calibrated) values of the instrument measurements (e.g., radiances, magnetic field strength). Referred to in the MESSENGER program as either derived data products (DDPs) or derived analysis products (DAPs). Corresponds to NASA Level 1B.
5	Derived Data	Derived results such as maps, reports, graphics, etc. Corresponds to NASA Levels 2 through 5
6	Ancillary Data	Non-Science data needed to generate calibrated or resampled data sets. Consists of instrument gains, offsets, pointing information for scan platforms, etc.
7	Corrective Data	Other science data needed to interpret space-borne data sets. May include ground based data observations such as soil type or ocean buoy measurements of wind drift.
8	User Description	Description of why the data were required, any peculiarities associated with the data sets, and enough documentation to allow a secondary user to extract information from the data.

The above is based on the National Research Council Committee on Data Management and Computation (CODMAC) data levels.

8.7 Appendix - Acronyms

ACT	Applied Coherent Technology Corporation
ADC	Analog-to-Digital Converter
AEM	Analog Electronics Module
AM	Atmosphere and Magnetosphere Group
APL	The Johns Hopkins University Applied Physics Laboratory
ASCII	American Standard Code for Information Interchange
CCD	Charged-Coupled Device
CCSDS	Consultative Committee for Space Data Systems
CDR	Calibrated Data Record
CK	C-Kernels (SPICE)
CODMAC	Committee on Data Management and Computation
Co-I	Co-Investigator
DAC	Digital-to-Analog Converter
DAP	Derived Analysis Product
DDP	Derived Data Product
DPU	Data Processing Unit
DSN	Deep Space Network
EDR	Experiment Data Records
EK	Event Kernel
EPPS	Energetic Particle and Plasma Spectrometer
ET	Ephemeris Time
FIPS	Fast Imaging Plasma Spectrometer
FITS	Flexible Image Transport System
FOV	Field-of-View
FPA	Focal Plane Assembly
FTP	File Transfer Protocol
GC	Geochemistry Group
GP	Geophysics Group
GRNS	Gamma-Ray and Neutron Spectrometer
GSFC	Goddard Space Flight Center
HSI	High Speed Input; measures laser pump pulse duration
I&T	Integration and Test
IK	Instrument Measurement Kernel (SPICE)
ILRC	International Laser Radar Conference
LFSR	Linear Feedback Shift Register
LSK	Leapseconds Kernel (SPICE)
MAG	Magnetometer
MASCS	Mercury Atmospheric and Surface Composition Spectrometer
MDIS	Mercury Dual Imaging System
MESSENGER	MErcury, Surface, Space ENvironment, Geochemistry, and Ranging
MET	Mission Elapsed Time
MLA	Mercury Laser Altimeter
NAIF	Navigation and Ancillary Information Facility
NASA	Navigation Aeronautics and Space Administration
PCK	Planetary Constant Kernel (SPICE)
PDS	Planetary Data System
PPS	Pulse Per Second
RDR	Reduced Data Record
RMU	Range Measurement Unit
RUPT	A timing pulse that generates an interrupt (see calibration document)
SCLK	Space Clock Kernel (SPICE)
SIS	Software Interface Specification
SOC	MESSENGER Science Operations Center
SPICE	Spacecraft, Planet, Instrument, C-matrix Events
SPK	Spacecraft and Planets Kernel (SPICE)
SQL	Structured Query Language
TOF	Time-of-flight ASIC
UTC	Coordinated Universal Time
VGA	Variable Gain Amplifier

8.8 Appendix - MLA Instrument Overview

The Mercury Surface, Space Environment, Geochemistry, and Ranging (MESSENGER) mission is to orbit Mercury following two flybys each of Venus and Mercury. It launched in August 2004 and will use two flybys of Venus and Mercury to achieve an orbit insertion around Mercury in March 2011. Initial data collection will begin during the two flybys of Mercury, and will primarily consist of global mapping and measurements of the surface, atmosphere, and magnetosphere composition. MESSENGER will then remain in orbit for the rest of the nominal mission, which is scheduled to end in March 2012 [see the *MESSENGER Data Management and Archiving Plan* for extended mission updates]. Once in orbit around Mercury it will begin a series of observations using multiple instruments. These observations will provide data to answer questions about the nature and composition of Mercury's crust, tectonic history, the structure of the atmosphere and magnetosphere, and the nature of the polar regions.

Most of the instruments are fixed-mounted, so that coverage of Mercury is obtained by spacecraft motion over the planet. The instruments are co-located on a science deck facing Mercury while the spacecraft is shielded from direct sunlight by a lightweight sunshade, (Figure A-1) below.

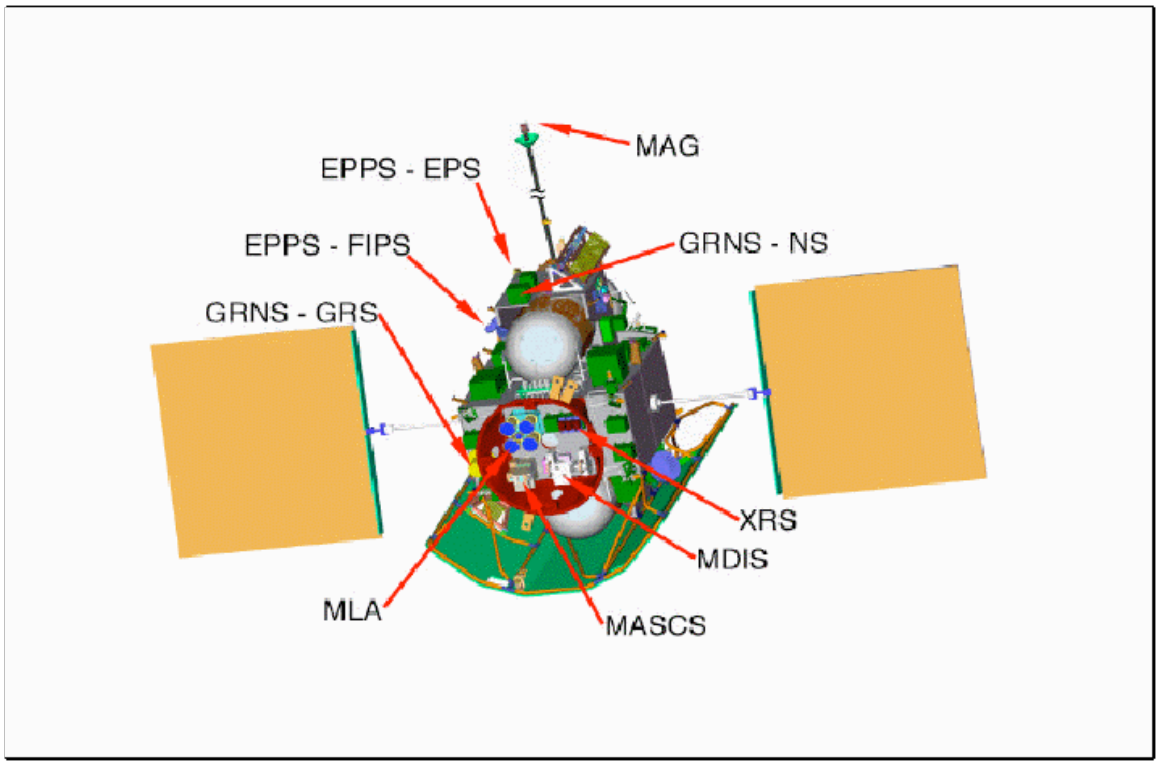


Figure A-1 Spacecraft with Instrument Locations.

The Mercury Laser Altimeter (MLA) uses a solid-state pulsed laser to measure the distance between the spacecraft and the surface of Mercury. This will allow the science team to take detailed measurements of Mercury's shape and surface structure. The MLA is a bi-static system, meaning that it consists of separate transmitter and receiver systems.

General Specifications:

Mass : 7.25kg (Allocated)
Power: 25.6 W Science. Mode (Allocated).
20W Orbit average.
Volume ~ 0.027m^3 (1ft^3); 33cm height limit (13")

The transmitter generates a brief laser pulse, and the instrument measures the time required for the light to reach the surface and return. The MLA data complements the visible and near infrared imaging that will also be performed on Mercury. Unlike the imager, the MLA does not rely on solar illumination and can make measurements over the entire surface of Mercury including the dark side. The MLA complements imaging because the direct range measurements enable unambiguous determinations of topography that will improve the interpretation of images.

More details about the instrument can be obtained at the following site: http://mla.gsfc.nasa.gov/ has been reserved.