W. W. Hansen Experimental Physics Laboratory

STANFORD UNIVERSITY

STANFORD, CALIFORNIA 94305-4085

Gravity Probe-B (GP-B) Relativity Mission

TDP to Level 0 Data Processing Verification Test Report

S0572, Rev. B

June 20, 2003

[illegible] 20 June 2003
Prepared by: Jennifer L. Spencer — Date
Data Processing Lead

[illegible] 20 June 2003
Prepared by: Paul McGown — Date
Binary Software Developer

[illegible] 23 June 03
Signature: Ron Sharbaugh — Date
Software Manager

[illegible] 23 June '03
Signature: Marcie Smith — Date
Mission Operations Manager

[illegible] 6/23/03
Signature: Kelly Burlingham — Date
Software Quality Assurance

[illegible] 6/20/03
Edited by: John Doiron — Date
Senior Technical Communicator

[illegible] 6·23·03
Tom Langenstein ITAR Assessment Performed, ITAR Control Req'd? Yes [✓ No]

Table of Contents

Terminology, Abbreviations, and Acronyms

TERMINOLOGY, ABBREVIATION, or ACRONYM	DESCRIPTION
APID	**A**synchronous **P**acket **ID**: Stanford GP-B's Sybase database data packet identification system
EDR	**E**ngineering **D**ata **R**eduction software: LM proprietary software, developed and used by LM to process raw binary data of the kind GP-B will be receiving from the SC during in-flight operations; used in the TDP to Level 0 Test Verification process as a known good data packets processing software.
GN	**G**round **N**etwork: controlled by Wallops Flight Facility (WFF); includes stations at Wallops Island, VA; Poker Flats, AK; McMurdo, Antarctica; and Scalbard, Norway
GP-B	**G**ravity **P**robe **B**
IPDU	**I**nternet **P**rotocol **D**ata **U**nit: the protocol used by the GN to transfer space telemetry from one point to another
LASP	**Laboratory** for **A**tmospheric and **S**pace **P**hysics; developer of the source code for TDP
lasp.xx	version .xx of software developed by LASP and purchased by GP-B for use in GP-B SC operations to process raw binary data that has been transmitted via telemetry; further developed by SU GP-B into TDP
LM	**L**ockheed **M**artin Corporation: GP-B mission SV development company
LV	**L**aunch **V**ehicle: the GP-B mission Boeing DELTA II rocket
MOC	GP-B **M**ission **O**perations **C**ontrol
Payload	All GP-B science mission electronics and other hardware housed in the SV
SC	**S**pace **C**raft: the underlying functionality of the mission contained in the SV, e.g., power system; attitude control; communications; etc.
SU	**S**tanford **U**niversity
SV	**S**pace **V**ehicle: the GP-B satellite
TDP	The non-real-time **T**elemetry **D**ata **P**rocessing software application suite developed by Stanford University from lasp.xx; used to process the raw binary spacecraft telemetry data files into mission-usable, Level 0 Sybase database data packets.
TLM	**T**e**l**e**m**etry
VC	**V**irtual **C**hannel: The Consultatative Committee on Space Data Systems' (CCSDS) TLM binary data stream delivery channel standard. The GP-B mission uses VC 0 (for real-time data), and VC 0, 1, and 2 (for non-real-time data stored in the GP-B Solid State Recorder [SSR]) channels and the binary packets created from these. Currently, a Front End Processor (FEP) at the LM Integrated Test Facility (ITF) attached to a GP-B SV emulator, and an FEP at the Sunnyvale LM Vehicle Test Facility attached to the GP-B SV, provide the `vc1and2` and `vc0` files referenced in the test case verifications described in this document.

Related Documents[1]

Number and Rev.	Date of Pub.	Title
S0331, Rev. A	6/30/00	*Data Management Plan*
S0401, Rev. A	6/3/02	*Stanford Post-Processing Operations for Science Mission Data*
S0503, Rev. E	10/18/02	*MOC Version Description Document (VDD) for TDP/TCAD*
S0567, Rev. –	8/23/02	*Mission Operations Demonstration Phase 1 Verification Requirement Compliance Matrix (VRCM)*
S0613, Rev. C	5/20/03	*Software Design TDP/TCAD*
P0949, Rev. A	5/30/03	*Operational Procedure for Regression Testing of TDP*
SCSE-16	unk	Section 9, "Flight Software Design Specification"
LMSSC/P480456-01, test case dta001	unk	*GP-B Mission Operations Center Ground Software Verification Test Reports and Procedures – EDRS v. 1.6*

MOC Specifications Verified by this Document

Number	Title
MOC-073	Data processing is required to process APID types: 100; 2xx; 300; 301; and 400 to the Level 0 database.
MOC-077	Data processing is required to archive the IPDU header time for each real-time transfer frame.
MOC-078	Data processing subsystems are required to uniquely identify each snapshot, including vehicle time and snapshot type.

[1] Please note that all document revisions and their publication dates listed in this table are the most current as of this document's, S0572, *TDP to Level 0 Data Processing Verification Report*, publication date. Users of this document need to consult with GP-B documentation control to ensure the documents listed in this table have not ben updated since this revision of S0572's publication.

1 Introduction

During the GP-B mission flight phase, mission-critical, non-real time telemetry (TLM) binary data packets will be streamed from the space vehicle (SV) to the GP-B Mission Operations Center (MOC). These binary data packets need to be processed into a human-readable format before they can be used by GP-B personnel. The first step in this processing is to the Level 0 state, as defined in S0331, *Data Management Plan.*

To process GP-B SV TLM data to Level 0, Stanford University (SU) GP-B Data Processing has developed the Telemetry Data Processing (TDP) application from the Laboratory for Atmospheric and Space Physics' `lasp.x-x` TLM data processing software. TDP consists of a number of programs used to process specific TLM data (see section 3.1, TDP Programs that were Verification Tested, page 6 for more information).

Before software deliveries, TDP to Level 0 TLM data processing verification testing is required. Document Purpose

This document describes the verification process used to verify versions 1.4, 1.6 and 1.7 of the LASP TDP code. Each delivery of the code was tested and results documented in previous revisions of this document as shown in the table below. Revision A did not include the results from the original version, but for this version, we are showing results for all three releases. Since these tests were complete, a formal test procedure was generated which documents the tests used here as well as others. P0949, Rev. A, *Operational Procedure for Baseline and Regression Testing of TDP* is now the standard procedure for verifying new releases of the code. The original drafts of these test reports are presented in Appendices A through E.

TDP Version	Notes	Detailed Test Documentation
v1.4	First release from SU of `gpb_tdp_L0.pro`	S0572, — (see Appendix A&B)
v1.6	Improved `gpb_tdp_L0.pro`; added `process_tcor.pro`	S0572, Rev A (see Appendix C&D)
v1.7	Improved `ssbuild.pro`, `gpb_tdp_L0.pro`	Appendix E

2 TDP Verification Testing Pass/Fail Criteria

The TDP to Level 0 test cases listed in section 0, Document Purpose above, PASSED when *all* of the test case steps *correctly* processed *all* of the data required per P0949, Rev. A, *Operational Procedure for Regression Testing of TDP.*

The TDP to Level 0 test cases listed in section 0, Document Purpose above, FAILED when *any* of the test case steps *incorrectly* processed *any* of the data required per P0949, Rev. A, *Operational Procedure for Regression Testing of TDP.*

3 TDP Verification Testing Software Background

3.1 TDP Programs that were Verification Tested

This section lists the TDP programs verified by the verification tests reports in this document.

3.1.1 GPB_TDP_L0.PRO

`gpb_tdp_L0.pro` is the TDP program that processes all types of raw telemetry data into the Level 0 database. There are five data types, APID 100 (programmable telemetry), APID 2xx (memory readouts), APID 300 (database readouts), APID 301 (event data) and APID 400 (snapshots).

3.1.2 PROCESS_TCOR.PRO

`process_tcor.pro` processes timing information from the IPDU headers that are packaged with the raw spacecraft telemetry. This program was first delivered with TDP v1.6.

3.1.3 SSBUILD.PRO

`ssbuild.pro` is a post processing tool that combines the raw packets of snapshot data into complete snapshots for the Level 0 database. This program was first delivered by LASP and improved over time to accommodate new snapshot types.

3.2 Software that was Used to Verification Test TDP Programs

Several software applications and programs were used in the verification testing of TDP to Level 0. This section provides information on these applications and their programs.

3.2.1 EDR

Engineering Data Reduction (EDR) is a verified, non-real-time, TLM data processing software developed by Lockheed Martin (LM). It was used in this validation testing as a known-good comparison source in all TDP2 verification test cases, as well as in TDP4. The application is limited as a comparison by its end format. TDP reduces data much further and more specifically than EDR, so EDR cannot be used in all verifications.

3.2.2 TERLXXXX.PL

The `TERLxxxx.pl`[2] application consists of a PERL-based set of software programs developed by SU GP-B data processing's Paul McGown to independently emulate TDP and EDR file outputs. The `TERLxxxx.pl` versions used in the TDP validation test cases described in this document are as follows:

- `TERLout6.pl`
- `TERLout8.pl`
- `TERLoutX.pl`
- `TERLTcor.pl`[3]

[2] `TERLxxxx.pl` and all of its successive versions, with the exception of `TERLoutX.pl`, were verification tested separately from their related TDP verification test cases. `TERLoutX.pl` was verification tested in TDP4; see section 5.3.1, Summary of TDP2: TDP v1.7 processing Apid 100/2xx/300/301 to Level 0, page 5, and Appendix E: Original Documentation of TDP4: TDP v1.7 Processing APID 400 SRE and TRE Snapshot Data to Level 0, page 37.

[3] In this test case, EDR files were not used as comparison files because EDR does not recognize IPDU Time Correlation Headers; see section 5.2.2, Summary of TDP3: TDP v1.6 Processing APID 100 IPDU Time Correlation Header Data to Level 0, page 16 for further details.

In the TPD verification testing reported in this document, these programs' TDP- and EDR-emulated outputs were compared against their respective TDP- and EDR-processed outputs using the UNIX `diff` function to verify TDP's processing using the following logic:

- **If** the `TERLxxx.pl` EDR-emulated output is the same as the known-good EDR-processed output (verified with `diff` compare)
- **And** the `TERLxxx.pl` TDP-emulated output is the same as the TDP-processed output (verified with `diff` compare)
- **And** the `TERLxxx.pl` EDR-emulated output is the same as the TERLxxx.pl TDP-emulated output (by definition, since both come from the same VC source file)
- **Then** the TDP-processed output is verified as good
- **Or**
- **If any** statement above is **False**
- **Then** the TDP-processed output is verified as not good

`TERLTcor.pl`, which was only used in the TDP3: TDP v1.6 Processing APID 100 IPDU Time Correlation Headers data to Level 0 test case, further processes its TDP-emulated output into ASCII format. The TDP v1.6's `process_tcor.pro` processing produced a binary output file which was, as it would be in the normal course of data processing, bulk copied into Sybase copied (using the UNIX `bcp` function). Because `process_tcor.pro`'s input of this vc0 file was the only input into the database thus far, an ASCII export from Sybase was done on the database so that data can be `diff` compared with the ASCII output of `TERLtcor.pl`

3.2.3 SSBUILD.PL

`ssbuild.pl` is an SU-developed software program designed to independently emulate TDP v1.7 `ssbuild.pro` output. Since EDR does not build assembed snapshots, `ssbuild.pl` was verified against `TERLoutX.pl` and then used to verify `ssbuild.pro.`

3.2.4 STEXPLODE.PL

`STExplode.pl` is an SU-developed software tool, designed to sort snapshot packet files and convert them to ASCII, allowing comparison between files.

3.2.5 SSEXPLODE.PL

`SSExplode.pl` is an SU-developed software tool, designed to sort assembed snapshot files and convert them to ASCII, allowing comparison between files.

4 Synopsis of TDP Verification Testing Results

This section provides the synoptic results of all TDP test verification cases included in this document listed by the revision of S0572 in which they were first documented.

4.1 TDP v1.4

4.1.1 SYNOPSIS OF TDP2 RESULTS: TDP V1.4 PROCESSING APID 100/2XX/300/301 DATA TO LEVEL 0

TDP v1.4 PASSED the TDP2 test case for processing APID 100/200/300/301 data to Level 0.

NOTE: See section 5.1.1, Summary of TDP2: TDP v1.4 Processing APID 100/2xx/300/301 Data to Level 0, page 10 for more information.

4.1.2 SYNOPSIS OF TDP2 RESULTS: TDP V1.4 PROCESSING APID 400 DATA TO LEVEL 0

TDP v1.4 PASSED the TDP2 test case for processing APID 400 data to Level 0.

NOTE: See section 5.1.2, Summary of TDP2: TDP v1.4 Processing APID 400 Data to Level 0, page 13 for more information.

4.2 TDP v1.6

4.2.1 SYNOPSIS OF TDP2 RESULTS: TDP V1.6 PROCESSING APID 100/2XX/300/301 DATA TO LEVEL 0

TDP v1.6 PASSED the TDP2 test case for processing 100/2xx/300/301 data to Level 0.

NOTE: See section 5.2.1, Summary of TDP2: TDP v1.6 Processing APID 100/2xx/300/301 Data to Level 0, page 14 for more information.

4.2.2 SYNOPSIS OF TDP3 RESULTS: TDP V1.6 PROCESSING APID 100 IPDU TIME CORRELATION HEADERS DATA TO LEVEL 0

TDP v1.6 PASSED the TDP3 test case for processing APID 100 IPDU Time Correlation Headers data to Level 0.

NOTE: See section 5.2.2, Summary of TDP3: TDP v1.6 Processing APID 100 IPDU Time Correlation Header Data to Level 0, page 16 for more information.

4.3 TDP v1.7

4.3.1 SYNOPSIS OF TDP2 RESULTS: TDP V1.7 PROCESSING APID 100/2XX/300/301 DATA TO LEVEL 0

TDP v1.7 PASSED the TDP2 test case for processing 100/2xx/300/301 data to Level 0.

NOTE: See page 1 of Appendix E: Original Documentation of TDP4: TDP v1.7 Processing APID 400 SRE and TRE Snapshot Data to Level 0, page 37 for more information.

4.3.2 SYNOPSIS OF TDP4 RESULTS: TDP V1.7 PROCESSING APID 400 SRE AND TRE SNAPSHOTS DATA TO LEVEL 0

TDP v1.7 PASSED all the TDP4 test cases for processing APID 400 SRE and TRE Snapshots data to Level 0.

NOTE: See section 5.3.1, Summary of TDP2: TDP v1.7 processing Apid 100/2xx/300/301 to Level 0, page 5, and Appendix E: Original Documentation of TDP4: TDP v1.7 Processing APID 400 SRE and TRE Snapshot Data to Level 0, page 37 for more information.

5 Summary of TDP Verification Testing Reports

This section provides summaries of each TDP verification test case report by the revision of S0572 in which these verification reports were first documented. It includes which functions required by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP* were tested and not tested, as well as what the approach, steps, and results of each test case were.

5.1 TDP v1.4

5.1.1 SUMMARY OF TDP2: TDP V1.4 PROCESSING APID 100/2XX/300/301 DATA TO LEVEL 0

5.1.1.1 Introduction

This section summarizes this TDP2 verification test case's report from S0572, Rev. –. That report detailed how TDP v1.4 `gpb_tdp_L0.pro` correctly decomposed APID 100/2xx/300/301 data from a VC file to Level 0, as required by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.2.

For the original documentation of this test case, see Appendix A: Original Documentation of TDP2: TDP v1.4 Processing APID 100, 2xx, 300 and 301 Data to Level 0, page 22.

5.1.1.2 Required Functions Tested

The following functions required for this TDP2 verification test case by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.2 were tested:

- The proper parsing of APID 100/2xx/300/301 data to Level 0
- The proper packet length processing of APID 100/200/300/301 data to Level 0
- The proper packet timestamp and byte content processing of APID 100/2xx/300/301 data to Level 0

5.1.1.3 Required Functions Not Tested

The following functions required for this TDP2 verification test case by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.2 were not tested:

- TDP v1.4 processing of APID 400 data to Level 0 was not tested in this TDP2 test case, as none of these data were present in the `vc1and2_107232001.bin` file.

  TDP v1.4 processing of APID 400 data to Level 0 was performed in a separate TDP2 test case using `vc1and2_105110221.bin` which did have APID 400 data; see section 5.1.2, Summary of TDP2: TDP v1.4 Processing APID 400 Data to Level 0, page 13.

5.1.1.4 Approach

Prior to testing, it was agreed upon that the following approach would be taken in this TDP2 verification test case:

- A VC file that included APID 100/2xx/300/301 data would be obtained for use in this test case. The file obtained was `vc1and2_107232001.bin`.

- `TERLout8.pl` would be verified by comparing its output with EDR (an independently verified software package.)
- TDP v1.4's `gpb_tdp_L0.pro` would then be verified against `TERLout8.pl`.

5.1.1.5 Test Case Steps

After deciding upon the approach above, the following steps were then performed to process the APID 100/2xx/300/301 data from `vcland2_107232001.bin` to Level 0 in this TDP2 verification test case:

1. `vcland2_107232001.bin` Processed by TDP v1.4 to Output TDP-Processed (`.tmp`) "A" Files:

```
/apps/supported/lasp/src/tdp/p100.tmp

/apps/supported/lasp/src/tdp/p200.tmp

/apps/supported/lasp/src/tdp/event.tmp

/apps/supported/lasp/src/tdp/p300.tmp
```

2. `vcland2_107232001.bin` Processed by EDR to Output EDR-Processed (`.bin`) "B" Files:

```
/home/jennifer/timing_test/alpha_test_07232001/progtlm.bin

/home/jennifer/timing_test/alpha_test_07232001/db_readout_
dr.bin

/home/jennifer/timing_test/alpha_test_07232001/eventdata.
bin

/home/jennifer/timing_test/alpha_test_07232001/memdump.bin
```

3. `vcland2_107232001.bin` Processed by TERLout8.pl to Output TDP-Emulated (`.binCO.xxx`[4]) "C" Files

```
88425   /home/pmcgown/TLMrpt/vcland2_107232001.binCO.100

14610   /home/pmcgown/TLMrpt/vcland2_107232001.binCO.2xx

11      /home/pmcgown/TLMrpt/vcland2_107232001.binCO.300

26517   /home/pmcgown/TLMrpt/vcland2_107232001.binCO.301
```

4. `vcland2_107232001.bin` Processed by TERLout8.pl to Output EDR-Emulated (`.binLM.xxx`[8]) "D" Files

```
88425   /home/pmcgown/TLMrpt/vcland2_107232001.binLM.100

14610   /home/pmcgown/TLMrpt/vcland2_107232001.binLM.2xx

11      /home/pmcgown/TLMrpt/vcland2_107232001.binLM.300

26517   /home/pmcgown/TLMrpt/vcland2_107232001.binLM.301
```

[4] `xxx` here stands for three numeric characters.

5. Verification of `TERLout8.pl` ("B" Files Compared to "D" Files)

   The following `diff` screen capture with its null response, copied from Appendix A: Original Documentation of TDP2: TDP v1.4 Processing APID 100, 2xx, 300 and 301 Data to Level 0, page 22, shows that the EDR-processed (`.bin`) "B" files output by EDR processing `vc1and2_107232001.bin` were identical to the EDR-emulated (`.binLM.xxx`) "D" files output by `TERLout8.pl` processing `vc1and2_107232001.bin`:

```
diff
/home/jennifer/timing_test/alpha_test_07232001/progtlm.bin
vc1and2_107232001.binLM.100

diff
/home/jennifer/timing_test/alpha_test_07232001/memdump.bin
vc1and2_107232001.binLM.2xx

diff
/home/jennifer/timing_test/alpha_test_07232001/db_readout_
dr.bin vc1and2_107232001.binLM.300

diff
/home/jennifer/timing_test/alpha_test_07232001/eventdata.b
in vc1and2_107232001.binLM.301
```

6. Verification of TDP v1.4 `gpb_tdp_L0.pro` ("A" Files Compared to "C" Files)

   The following `diff` screen capture with its null response, copied from Appendix A: Original Documentation of TDP2: TDP v1.4 Processing APID 100, 2xx, 300 and 301 Data to Level 0, page 22, shows that the TDP v1.4-processed (`.tmp`) "A" files output by `gpb_tdp_L0.pro` processing `vc1and2_107232001.bin` were identical to the TDP v1.4-emulated (`.binCO.xxx`) "C" files output by `TERLout8.pl` processing `vc1and2_107232001.bin`:

```
diff /apps/supported/lasp/src/tdp/p100.tmp
vc1and2_107232001.binCO.100

diff /apps/supported/lasp/src/tdp/p200.tmp
vc1and2_107232001.binCO.2xx

diff /apps/supported/lasp/src/tdp/p300.tmp
vc1and2_107232001.binCO.300

diff /apps/supported/lasp/src/tdp/events.tmp
vc1and2_107232001.binCO.301
```

5.1.1.6 Test Case Results

TDP2: TDP v1.4 processing APID 100/2xx/300/301 data to Level 0 – PASSED.

Each `diff` comparison performed for this verification test case rendered null results, showing all compared files to be identical.

5.1.2 SUMMARY OF TDP2: TDP V1.4 PROCESSING APID 400 DATA TO LEVEL 0

5.1.2.1 Introduction

This section summarizes this TDP2 verification test case's report from S0572, Rev. –. That report detailed how TDP v1.4 `gpb_tdp_L0.pro` correctly decomposed APID 400 data from a VC file to Level 0, as required by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.2.

5.1.2.2 Required Functions Tested

The following functions required for this TDP2 verification test case by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.2 were tested:

- The proper parsing of APID 400 data to Level 0
- The proper packet length processing of APID 400 data to Level 0
- The proper packet timestamp and byte content processing of APID 400 data to Level 0

5.1.2.3 Required Functions Not Tested

The following functions required for this TDP2 verification test case by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.2 were not tested:

- TDP v1.4 processing of APID 100/2xx/300/301 data to Level 0 was not tested in this TDP2 test case, as none of these data were present in the `vcland2_105110221.bin` file.

  TDP v1.4 processing of APID 100/2xx/300/301 data to Level 0 was performed in a separate TDP2 test case using `vcland2_107232001.bin` which did have APID 100/2xx/300/301 data; see section 5.1.1, Summary of TDP2: TDP v1.4 Processing APID 100/2xx/300/301 Data to Level 0, page 10.

5.1.2.4 Approach

Prior to testing, it was agreed upon that the following approach would be taken in this TDP2 verification test case:

- A VC file that included APID 400 data would be obtained for use in this test case. The file obtained was `vcland2_105110221.bin`.
- `TERLout6.pl` would be verified by comparing its output with EDR.
- TDP v1.4's `gpb_tdp_L0.pro` would then be verified against `TERLout6.pl`.

5.1.2.5 Test Case Steps

The following steps were performed to process APID 400 data from `vcland2_105110221.bin` to Level 0 in this TDP2 verification test case:

1. `vcland2_105110221.bin` Processed by TDP v1.4 to Output a TDP-Processed (`.tmp`) "A" File:

   ```
   ss.tmp
   ```

2. `vcland2_105110221.bin` Processed by EDR to Output an EDR-Processed (`.bin`) "B" File:

```
SREsnapshot.bin
```

3. `vc1and2_105110221.bin` Processed by `TERLout6.pl` to Output a TDP-Emulated (`.binCO.400`) "C" File:

   ```
   /home/pmcgown/TLMrpt/vc1and2_105110221.binCO.400
   ```

4. `vc1and2_105110221.bin` Processed by TERLout6.pl to Output an EDR-Emulated (`.binLM.400.SRE`) "D" File:

   ```
   /home/pmcgown/TLMrpt/vc1and2_105110221.binLM.400.SRE
   ```

5. Verification of TERLout6.pl ("B" File Compared to "D" File)

   The following `diff` screen capture with its null response, copied from Appendix B:Original Documentation of TDP2: TDP v1.4 Processing APID 400 Data to Level 0, page 28, and shows that the EDR-processed (`.bin`) "B" files output by EDR `vc1and2_105110221.bin` were identical to the EDR-emulated (`.binLM.400.SRE`) "D" output files output by `TERLout6.pl` processing `vc1and2_105110221.bin`:

   ```
   diff SREsnapshot.bin    vc1and2_105110221.binLM.400.SRE
   ```

6. Verification of TDP v1.4 `gpb_tdp_L0.pro` ("A" File Compared to "C" File)

   The following `diff` screen capture with its null response, copied from Appendix B:Original Documentation of TDP2: TDP v1.4 Processing APID 400 Data to Level 0, page 28, shows that the TDP-processed (`.tmp`) "A" output files output from `gpb_tdp_L0.pro` processing `vc1and2_105110221.bin` were identical to the TDP-emulated (`.binCO.400`) "C" files output by `TERLout6.pl` processing `vc1and2_105110221.bin`:

   ```
   diff ss.tmp  vc1and2_105110221.binCO.400
   ```

5.1.2.6 Test Case Results

TDP2: TDP v1.4 processing APID 400 data to Level 0 – PASSED.

Each `diff` comparison performed for this test case rendered null results, showing all compared files to be identical.

5.2 TDP v1.6

5.2.1 SUMMARY OF TDP2: TDP V1.6 PROCESSING APID 100/2XX/300/301 DATA TO LEVEL 0

5.2.1.1 Introduction

This section summarizes this TDP2 test case's verification report from S0572, Rev. A. That report detailed how TDP v1.6 `gpb_tdp_L0.pro` correctly decomposed APID 400 data from a VC file to Level 0, as required by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.2.

NOTE: It became necessary to re-test, re-verify, and re-document this TDP2 processing of APID 100/2xx/300/301 data to Level 0 verification test case after the publication of S0572, Rev. –, because TDP was revised from v1.4 to v1.6.

5.2.1.2 Required Functions Tested

The following functions required for this TDP2 test case by P0949, Rev. A *Operational Procedure for Regression Testing of TDP*, section 10.2 were tested:

- The proper parsing of APID 100/2xx/300/301 data to Level 0
- The proper packet length processing of APID 100/2xx/300/301 data to Level 0
- The proper packet timestamp and byte content processing of APID 100/2xx/300/301 data to Level 0

5.2.1.3 Required Functions Not Tested

The following functions required for this TDP2 verification test case by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.2 were not tested:

- TDP v1.6 processing of APID 400 data to Level 0 was not tested in this TDP2 test case. No changes were made to TDP that affected the processing of APID 400 data.

5.2.1.4 Approach

Prior to testing, it was agreed upon that the following approach would be taken in this TDP2 verification test case:

- A VC file that included APID 100/2xx/300/301 data would be obtained for use in this test case. The file obtained was `vc1and2_111132137.bin`.
- TDP v1.6's `gpb_tdp_L0.pro` would then be verified against `TERLout8.pl` (which was verifed with TDP v1.4)

5.2.1.5 Test Case Steps

The following steps were used to process APID 400 data to Level 0 from `vc1and2_111132137.bin` in this TDP2 verification test case:

1. `vc1and2_111132137.bin` Processed by TDP v1.6 to Output TDP-Processed (`.tmp`) "A" Files:

```
p100.tmp

p300.tmp

events.tmp

p200.tmp
```

2. `vc1and2_111132137.bin` Processed by `TERLout8.pl` to Output TDP-Emulated (`.binCO.xxx`) "B" Files:

```
226990      /home/pmcgown/VC/vc1and2_111132137.binCO.100

6001        /home/pmcgown/VC/vc1and2_111132137.binCO.2xx

4           /home/pmcgown/VC/vc1and2_111132137.binCO.300

68092       /home/pmcgown/VC/vc1and2_111132137.binCO.301
```

3. Verification of TDP v1.6 `gpb_tdp_L0.pro` ("A" Files Compared to "B" Files)

   The following `diff` screen capture with its null response, copied from Appendix C: Original Documentation of TDP2: TDP v1.6 Processing APID 100, 2xx, 300 and 301 Data to Level 0, page 32, shows that the (`.tmp`) "A" TDP v1.6-processed files output by `gpb_tdp_L0.pro` processing `vcland2_111132137.bin` were identical to the (`.binCO.xxx`) "B" TDP v1.6-emulated files output by `TERLout8.pl` processing `vcland2_111132137.bin`:

```
diff p100.cycle-4001.tmp vcland2_111132137.binCO.100

diff p200.cycle-4001.tmp vcland2_111132137.binCO.2xx

diff p300.cycle-4001.tmp vcland2_111132137.binCO.300

diff events.cycle-4001.tmp vcland2_111132137.binCO.301
```

5.2.1.6 *Test Case Results*

TDP2: TDP v1.6 processing APID 100/2xx/300/301 data to Level 0 – PASSED.

Each `diff` comparison performed for this test case rendered null results, showing all compared files to be identical.

5.2.2 SUMMARY OF TDP3: TDP V1.6 PROCESSING APID 100 IPDU TIME CORRELATION HEADER DATA TO LEVEL 0

5.2.2.1 *Introduction*

This section summarizes this TDP3 test case's verification report from S0572, Rev. A. That report detailed how TDP v1.6 `process_tcor.pro` correctly decomposed APID 100 IPDU Time Correlation Header data from a VC file to Level 0, as required by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.3.

5.2.2.2 *Required Functions Tested*

The following functions required for this TDP3 test case by P0949, Rev. A *Operational Procedure for Regression Testing of TDP*, section 10.3 were tested:

- The proper IPDU parsing of `vc0_111132137.bin` file Time Correlation Headers by `process_tcor.pro` to Level 0

- The proper least-squares fit generated by `process_tcor.pro` to Level 0

5.2.2.3 *Required Functions Not Tested*

The following functions required for this TDP3 test case by P0949, Rev. A *Operational Procedure for Regression Testing of TDP*, section 10.3 were not tested:

- All required functions were tested

5.2.2.4 *Approach*

Prior to testing, it was agreed upon that the following approach would be taken in this TDP3 verification test case:

- A VC file that included IPDU Header data would be obtained for use in this test case. The file obtained was `vc1and2_111132137.bin`.
- TDP v1.6's `process_tcor.pro` would be verified by comparing to an independent program `TERLtcor.pl`.

5.2.2.5 Test Case Steps

The following steps were performed to process IPDU Time Correlation Headers data to Level 0 from `vc0_111132137.bin` in this TDP3 verification test case:

1. `vc0_111132137.bin` Processed by TDP v1.6 `process_tcor.pro` to Output an TDP-Processed binary file, `tcor_raw.tmp,` and `tcor_fit.tmp`, which are bulk-copied to Sybase. Their data was exported out of the database to ASCII file "A" (name unknown) and ASCII file "B". File "A" contained the IPDU header data and file "B" contained the least-squares fit information:

    - Name of file was not recorded. `process_tcor.pro` immediately copies the data into the Sybase database. It was exported out as an ASCII file, the name of which was not recorded.

2. `vc0_111132137.bin` Processed by `TERLtcor.pl` to Output a "C" TDP-Emulated (`tcor.raw`) ASCII file, files "C which contained both raw and least-squares fit data:

3. `vc0_111132137.rpt.100.tcor.raw`Verification of TDP v1.6 `process_tcor.pro` ("A" File Compared to "C" File, and file "B" compared to file "C")

    The user visually compared the `TERLTcor.pl` TDP-emulated output file, `vc0_111132137.rpt.100.tcor.raw`, with the `process_tcor.pro` TDP-processed ASCII Sybase database files. All 300 entries were exactly the same, and the least squares fit number only differed by a factor of 10^{-6} for the 300 data points. This difference is accounted for by the fact that the `TERLtcor.pl` routine uses a slightly different least squares fit algorithm than the process_tcor routine. All data files can be found on the Level 0 verification tape in the GP-B MOC Manager's tape library.

5.2.2.6 Test Case Results

TDP3: TDP v1.6 processing APID 100 IPDU Time Correlation Headers data to Level 0 – PASSED.

There were no critical differences between the files reviewed.

5.3 TDP v1.7

5.3.1 SUMMARY OF TDP2: TDP V1.7 PROCESSING APID 100/2XX/300/301 TO LEVEL 0

Please see Appendix E: Original Documentation of TDP4: TDP v1.7 Processing APID 400 SRE and TRE Snapshot Data to Level 0, page 37 for a summary of this test case.

5.3.2 SUMMARY OF TDP4: TDP V1.7 PROCESSING APID 400 SRE AND TRE SNAPSHOT DATA TO LEVEL 0

5.3.2.1 Introduction

This section summarizes this TDP4 verification test case's report for S0572, Rev. B. It details how TDP v1.7 `ssbuild.pro` correctly decomposed APID 400 SRE and TRE Snapshot data from a VC file to Level 0, as required by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.4.

For the original documentation of this test case, see Appendix E: Original Documentation of TDP4: TDP v1.7 Processing APID 400 SRE and TRE Snapshot Data to Level 0, page 37.

5.3.2.2 Required Functions Tested

The following functions required for this TDP4 verification test case by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.4 were tested:

- The proper parsing of APID 400 SRE and TRE Snapshot data to Level 0
- The proper packet length processing of APID 400 SRE and TRE Snapshot data to Level 0
- The proper packet timestamp and byte content processing of APID 400 SRE and TRE Snapshot data to Level 0

5.3.2.3 Required Functions Not Tested

The following functions required for this TDP4 verification test case by P0949, Rev. A, *Operational Procedure for Regression Testing of TDP*, section 10.4 were not tested:

- Processing APID 400 GSS Snapshot data to Level 0 since this data was not available in `vc1and2_207121829.bin`.

5.3.2.4 Approach

Prior to testing, it was agreed upon that the following approach would be taken in this TDP4 verification test case:

- A VC file that included APID 400 SRE and TRE Snapshot data would be obtained for use in this test case. The file obtained was `vc1and2_207121829.bin`.
- `TERLoutX.pl`, would be verified using EDR.
- TDP v1.7 `gpb_tdp_L0.pro` would be verified for packet processing with TERLoutX.pl.
- `ssbuild.pl` would be verified using `TERLout.pl`.
- TDP v1.7 `ssbuild.pro` would be verified for full snapshot assembly with `ssbuild.pl`.

5.3.2.5 *Test Case Steps*

After deciding upon the approach above, the following steps were performed to process the APID 400 SRE and TRE Snapshot data in `vcland2_207121829.bin` to Level 0 in this TDP4 verification test case:

1. `vcland2_207121829.bin` Processed by EDR to Output EDR-Processed "A1" and "A2" Files:

   `SREsnapshot.bin`

   `TREsnapshot.bin`

2. `vcland2_107232001.bin` Processed by `TERLoutX.pl` to Output EDR-Emulated "B1" and "B2" Files::

   `vcland2_207121829.binLM.SRE`

   `vcland2_207121829.binLM.TRE`

3. Verification of TERLoutX.pl ("A" Files Compared to "B" Files)

   While there is no screen capture to confirm this TDP4 step, it was reported that the "A1" `SREsnapshot.bin` output file was Unix `diff` compared to the "B2" `vcland2_207121829.binLM.SRE` output file with a null result, verifying that `TERLoutX.pl` was processing the APID 400 SRE Snapshot data in `vcland2_207121829.bin` properly.

   Additionally, while there is no screen capture to confirm this TDP4 step, it was reported that the EDR-processed "A2" `TREsnapshot.bin` output file was Unix `diff` compared to the `TERLoutX.pl` "B2" `vcland2_207121829.binLM.TRE` output file with a null result, verifying that `TERLoutX.pl` was processing the APID 400 TRE Snapshot data in `vcland2_207121829.bin` properly.

4. `vcland2_207121829.bin` Processed by TDP `gpb_tdp_L0.pro` to Output TDP-Processed "C" File:

   `ss4200.snaptmp.111662`

5. Snaptmp "C" File Processed by `STExplode.pl` to Output an ASCII file for comparison, File "D":

   `ss4200.snaptmp.111662.STExplode`

6. `vcland2_207121829.bin` Processed by `TERLoutX.pl` to Output a TDP v1.6-Emulated File, file "E":

   `vcland2_207121829.binCO.400.snaptemp.109005`

7. Snaptemp "E" File Processed by `STExplode.pl` to Output an ASCII File for comparion, file "F"

   `vcland2_207121829.binCO.400.snaptemp.109005.STexplode`

8. Verification of `gpb_tdp_L0.pro` snapshot packets ("D" Files Compared to "F" Files):

   While there is no screen capture to confirm this TDP4 step, it was reported that the ASCII "D" `ss4200.snaptmp.111662.STExplode` output file was visually compared to the ASCII "F" `vcland2_207121829.binCO.400 .snaptemp.109005.STexplode` output file. The TDP files include all data from its snapshot area, not just the data from the original file. All data from the orginal input file matched exactly, verifying that `gpb_tdp_L0.pro` was processing the APID 400 Snapshot data in `vcland2_207121829.bin` properly.

9. File "E" (`TERLoutX.pl` file) Processed by `SSbuild.pl` to Output an `SSbuild.pro`-Emulated File, file "G":

   `vcland2_207121829.binCO.snaptemp.400.109005SNAPSHOT.bin`

10. Snapshot file "G" Processed by `SSExplode.pl` to Output an ASCII Comparison File, file "H":

    `vcland2_207121829.binCO.snaptemp.400.109005SNAPSHOT.bin.STExplode`

11. `vcland2_207121829.bin` Processed by `TERLoutX.pl` to Output a TDP-emulated File, file "I":

    `vcland2_207121829.binCO.400`

12. Snapshot file "I" Processed by `SSExplode.pl` to Output an ASCII Comparison "J" File:

    `vcland2_207121829.binCO.400.SSExplode`

13. Verification of `ssbuild.pl` ("H" Files Compared to "J" Files)

    While there is no screen capture to confirm this TDP4 step, it was reported that the ASCII "H" `vcland2_207121829.binCO.snaptemp.400 .109005/SNAPSHOT.bin.SSexplode` ASCII comparison file was UNIX `diff` compared to the ASCII "J" `vcland2_207121829.binCO.400 .SSexplode` ASCII comparison file with a null result, verifying that `SSbuild.pl` was processing the APID 400 Snapshot data in `vcland2_207121829.bin` properly.

14. File "C" Processed by `SSbuild.pro` to Output File "K":

    `snapshots.tmp`

15. File "E" Processed by `ssbuild.pl` to Output file "L":

    `SNAPSHOT.bin`

16. Verification of `ssbuild.pro` ("K" files Compared to "L" Files)

    While there is no screen capture to confirm this TDP4 step, it was reported that the ASCII "K" `snapshots.tmp` ASCII comparison file was UNIX `diff` compared to the ASCII file "K," `SNAPSHOT.bin`, with a null result, verifying that TDP v1.7 `gbp_tdp_L0.pro` and `SSbuild.pro` were processing the APID 400 Snapshot data in `vcland2_207121829.bin` to Level 0 properly.

5.3.2.6 Test Case Results

TDP4: TDP v1.7 processing APID 400 Snapshot data to Level 0 – PASSED.

Each `diff` comparison performed for this verification test case rendered null results (or identical for the data in the input file), showing all compared files to be correct.

Appendix A: Original Documentation of TDP2: TDP v1.4 Processing APID 100, 2xx, 300 and 301 Data to Level 0

A GP-B spacecraft binary file of the same format as expected in flight, vc1and2_107232001.bin, was independently processed twice, first using TDP, version 1.4, and then using the Lockheed Engineering Data Reduction System (EDRS). The processing produced binary output files from both TDP and EDRS. The TDP output files are, in the normal course of data processing, copied directly into Sybase using the thoroughly-verified Sybase tool "bcp". This analysis examines the binary files prepared by TDP for use by Sybase (before they would be copied into Sybase). EDRS binary output files are the end product of the system and are not further processed under normal circumstances. This analysis uses these files as well.

A third-party custom-written PERL program, TERLout8.pl, was used to process the same binary spacecraft data. APID types 100, 2xx, 300 and 301 were verified against files of LM's EDRS software and compared to TDP output files through Level 0 processing. Using the binary spacecraft file vc1and2_107232001.bin, TERLout8.pl compared the outputs of EDRS and TDP against its own output.

The program TERLout8.pl was run as follows:

At the UNIX prompt, the following string was entered:

```
/home/pmcgown/PERL/TERLout8.pl

*****************************************************************

  CYCLE NUMBER ? (signed integer)

               like   -99 ... 1 ... 99   [ -99 is default ]

*****************************************************************

        -98
```

The response from the program was:

```
*****************************************************************

HYDROACTIVE:  Which DEFAULT INPUT DIRECTORY do you want to use ?

                A     /home/pmcgown/TLMdata/

                B     /home/ftp/

                C     /home/davis/gpb/tdp/

                P     /home/pmcgown/TLMrpt/

                N     (other)

*****************************************************************

        n
```

The option chosen by the user was n, use another directory than those shown when looking for the source telemetry file given above (vc1and2_107232001.bin), as shown below:

```
**************************************************************

  HYDROACTIVE:  Input Directory ?

**************************************************************
```

The user then entered the full path of the input directory:

```
/home/jennifer/timing_test/official_test/
```

The system found any .bin files in the input directory and listed them for the user's convenience, then asked for a filename:

```
* * * * * * * * * * * * * *

  vc1and2_107232001.bin

* * * * * * * * * * * * * *

INTERACTIVE:  FILENAME ?
```

The user entered in the filename:

```
vc1and2_107232001.bin
```

The program found the file and then asked user if he or she wanted to check the file by reading the bottom of it.

```
INTERACTIVE:  Do you want to TAIL the TLM file { Y or [ N ] } ?

N
```

The program requested an APID type for parsing (and later comparison):

```
INTERACTIVE:  Enter SubType, integer or patterns separated by pipes ("|") ?

100 200 202 211 212 213 214 220 300 301 400 2047 ALL XXX 10X 1XX 20X 21X
22X 2XX 30X 3XX 40X 4XX
```

The user chose "xxx", which requests that all packet types be processed except APID 2047, the idle packets. [This particular data set contains no APID 400 packets; that verification is addressed in the next section.]

```
    xxx
```

The outputs of the TDP data were compared first, so when an output type was requested, the user chose "3", the "CO" or Colorado (since TDP was created by LASP at the University of Colorado) option:

```
INTERACTIVE:  Output Type, enter an integer ?

          0 = HDR information

          1 = PKT data

          2 = RPT output file

          3 = CO BIN output file

          4 = LM BIN output file


    3

          3
```

The system confirmed that the user had selected option 3 above, and that the user wanted all APID types processed (below):

```
SubType =  100|200|202|211|212|213|214|220|300|301|400
Including ID=
                     100 = TLM
                     200 = MRO - CCCA
                     202 = MRO - SSR
                     211 = MRO - GSS1
                     212 = MRO - GSS2
                     213 = MRO - GSS3
                     214 = MRO - GSS4
                     220 = MRO - Squid/ST
                     300 = DBRO
                     301 = EVENT
                               400 = SNAPSHOTS
```

The next output marked the beginning of formal binary processing by this program. It shows synchronization pattern and start bytes:

```
74C2472C at    2
74C2472C at 1298
1296
```

The system takes the difference of the two numbers above to compute the transfer frame length. The repeat word count in this case is 1298-2=1296 (the length of the transfer frame above in bytes).

In this particular case, no Snapshot data (APID 400) was available. Snapshot verification is addressed below using another data file:

```
          ## of SNAPSHOTs =    0
```

After about 30 minutes, the program informed the user that the following files have been created for comparison against the data files TDP creates. The number on the left refers to the number of packets found during parsing (e.g. 88425 APID 100 data packets were found for the first file).

```
88425   /home/pmcgown/TLMrpt/vc1and2_107232001.binCO.100
14610   /home/pmcgown/TLMrpt/vc1and2_107232001.binCO.2xx
11      /home/pmcgown/TLMrpt/vc1and2_107232001.binCO.300
26517   /home/pmcgown/TLMrpt/vc1and2_107232001.binCO.301
```

If TDP is correctly parsing its data, a binary "diff" should yield zero against the above files when compared with TDP output files for the same APIDs. When a binary "diff" against Lockheed data files (from the already-verified EDRS) and files in "LM" format created by TERLout8.pl was performed (shown below), a zero was also yielded. In this way, we are comparing TDP against an already-verified data reduction system using the same data files. If all differences are zero between processed files, TDP processed its Level 0 data successfully.

After running TERLout8.pl for the LASP/Colorado/TDP file format, differences were performed against the TDP files themselves: p100.tmp, p200.tmp, p300.tmp and events.tmp (the APID type 301 file). The screen results were:

```
diff /apps/supported/lasp/src/tdp/p100.tmp vc1and2_107232001.binCO.100

diff /apps/supported/lasp/src/tdp/p300.tmp vc1and2_107232001.binCO.300

diff /apps/supported/lasp/src/tdp/events.tmp vc1and2_107232001.binCO.301

diff /apps/supported/lasp/src/tdp/p200.tmp vc1and2_107232001.binCO.2xx
```

(i.e. a null response)

These results indicate that there are no differences between TERLout8.pl outputs and the TDP output files.

TERLout8.pl was then run again, but this time requesting the LM (EDRS) file format (option 4 of `INTERACTIVE:  Output Type, enter an integer ?` query), receiving the following screen output:

```
/home/pmcgown/PERL/TERLout8.pl
*****************************************************************
  CYCLE NUMBER ? (signed integer)
               like   -99 ... 1 ... 99   [ -99 is default ]
*****************************************************************
       -98
*****************************************************************
HYDROACTIVE:  Which DEFAULT INPUT DIRECTORY do you want to use ?
                A     /home/pmcgown/TLMdata/
                B     /home/ftp/
                C     /home/davis/gpb/tdp/
                P     /home/pmcgown/TLMrpt/
                N     (other)
*****************************************************************
       n
**************************************************************
  HYDROACTIVE:  Input Directory ?
**************************************************************
       /home/jennifer/timing_test/official_test/
 * * * * * * * * * * * * * *
  vc1and2_107232001.bin
 * * * * * * * * * * * * * *
  INTERACTIVE:  FILENAME ?
       vc1and2_107232001.bin
```

```
INTERACTIVE:  Do you want to TAIL the TLM file { Y or [ N ] } ?
N
INTERACTIVE:  Enter SubType, integer or patterns separated by pipes ("|") ?
100 200 202 211 212 213 214 220 300 301 400 2047 ALL XXX 10X 1XX 20X 21X
22X 2XX 30X 3XX 40X 4XX
        xxx
          INTERACTIVE:  Output Type, enter an integer ?
                   0 = HDR information
                   1 = PKT data
                   2 = RPT output file
                   3 = CO BIN output file
                   4 = LM BIN output file
             4
                 4
          SubType =  100|200|202|211|212|213|214|220|300|301|400
             Including ID=
                        100 = TLM
                        200 = MRO - CCCA
                        202 = MRO - SSR
                        211 = MRO - GSS1
                        212 = MRO - GSS2
                        213 = MRO - GSS3
                        214 = MRO - GSS4
                        220 = MRO - Squid/ST
                        300 = DBRO
                        301 = EVENT
                        400 = SNAPSHOTS
74C2472C at    2
74C2472C at 1298
1296
```

The following data files created were:

```
88425  /home/pmcgown/TLMrpt/vc1and2_107232001.binLM.100
14610  /home/pmcgown/TLMrpt/vc1and2_107232001.binLM.2xx
11     /home/pmcgown/TLMrpt/vc1and2_107232001.binLM.300
26517  /home/pmcgown/TLMrpt/vc1and2_107232001.binLM.301
```

After running TERLout8.pl for the LM EDRS (verified) file format, we performed differences against the LM EDRS files themselves: progtlm.bin (APID type 100 file), db_readout_dr.bin (APID type 300 file), eventdata.bin (APID type 301 file) and memdump.bin (the APID type 2xx file). Screen results were:

```
diff /home/jennifer/timing_test/alpha_test_07232001/progtlm.bin
vc1and2_107232001.binLM.100

diff /home/jennifer/timing_test/alpha_test_07232001/db_readout_dr.bin
vc1and2_107232001.binLM.300

diff /home/jennifer/timing_test/alpha_test_07232001/eventdata.bin
vc1and2_107232001.binLM.301

diff /home/jennifer/timing_test/alpha_test_07232001/memdump.bin
vc1and2_107232001.binLM.2xx
```

(i.e. a null response)

Appendix B: Original Documentation of TDP2: TDP v1.4 Processing APID 400 Data to Level 0

TERLout6.pl data (written by Paul McGown) was the program used to process binary spacecraft. This program was used to verify TDP snapshot processing in this case against the LM SREsnapshot.bin file produced by EDRS (already a validated piece of software). In this case, the file being processed by all three softwares was vc1and2_105110221.bin, a 7.2-hour telemetry dump file taken at the ITF on May 11, 2001.

At the UNIX prompt, the following string was entered:

```
/home/pmcgown/PERL/TERLout6.pl vc1and2_105110221.bin
```

Response from the program was:

```
*****************************************************************

HYDROACTIVE:  Which DEFAULT INPUT DIRECTORY do you want to use ?
                A    /home/pmcgown/data/
                B    /home/ftp/
                N    (other)

*****************************************************************
```

The option chosen by the user was b, use the /home/ftp directory when looking for the source telemetry file given above (vc1and2_105110221.bin), as shown below:

```
b
```

The program found the file and then asked user if he or she wanted to check the file by reading the bottom of it.

```
INTERACTIVE:  Do you want to TAIL the TLM file { Y or [ N ] } ?
```

The user answered no:

```
n
```

The system confirmed no:

```
N
```

The program asked the user which telemetry packet type(s) to process. In this case, comparing snapshot data was beng compared, so the user chose APID 400 as follows:

```
INTERACTIVE:  Enter SubType, integer or patterns separated by pipes ("|") ?

100 200 202 211 212 213 214 220 300 301 400 2047 ALL XXX 10X 1XX 20X 21X
22X 2XX 30X 3XX 40X 4XX
```

The user answered 400:

```
400
```

The user was asked to enter the format of the output file desired. This program emulates Lockheed ***snapshot.bin (where *** is a member of {SRE, TRE, GSS}), by snapshot type, or TDP ss.tmp format (options 3 and 4). Options 3 and 4 below create binary files. Options 0, 1 & 2 are diagnostic tools that give screen output.

```
INTERACTIVE:  Output Type, enter an integer ?
          0 = HDR information
          1 = PKT data
          2 = RPT output file
          3 = CO BIN output file
          4 = LM BIN output file
```

The user requested type 3 as a comparison mechanism for the TDP ss.tmp file:

```
3
```

The system confirmed type 3:

```
3
```

The system re-confirmed choices:

```
SubType =  400
Including ID=  400
```

The next output marks the beginning of formal binary processing by this program. It shows synchronization pattern and start bytes:

```
74C2472C at    2
74C2472C at 1298
```

The system takes the difference of the two above numbers to figure out the transfer frame length. Repeat word count in this case is 1298-2=1296 (the length of the transfer frame in bytes):

```
1296
```

Thirty or forty minutes later, the user received the following screen output:

```
## of SNAPSHOTs =   1201
## type=   1    =    301
## type=   2    =    300
## type=   3    =    300
## type=   4    =    300
```

All output files for output type 3, CO BIN output files, were marked as "*.bin.CO":

```
Output file = "vc1and2_105110221.binCO.400"
```

ss.tmp is the file created by TDP that is copied exactly as is into Sybase. Comparing the ss.tmp file against the SREsnapshot.bin file via this program, TERLout6.pl, verified correct snapshot processing by TDP. If there is no difference between the ss.tmp file and the output of this program, vc1and2_105110221.binCO.400, and no difference between SREsnapshot.bin and vc1and2_105110221.binLM.400.SRE (the output of this program using option 4, the LM bin output file format), we know that TDP is successfully processing SRE snapshot data. (The LM EDRS, Engineering Data Reduction System, has already been verified. SREsnapshot.bin is its output.)

The user issued the following UNIX directive:

```
diff ss.tmp  vc1and2_105110221.binCO.400
```

This yielded a null response, indicating no differences in the binary files between the Colorado format and the output of TERLout6.pl.

Next, the Lockheed format of the file vc1and2_105110221.bin was processed as follows:

At the UNIX prompt, the following string was entered:

```
/home/pmcgown/PERL/TERLout6.pl vc1and2_105110221.bin
```

The program responded:

```
*****************************************************************
 HYDROACTIVE:  Which DEFAULT INPUT DIRECTORY do you want to use ?
                A    /home/pmcgown/data/
                B    /home/ftp/
                N    (other)
*****************************************************************

        b


INTERACTIVE:  Do you want to TAIL the TLM file { Y or [ N ] } ?

n
N
INTERACTIVE:  Enter SubType, integer or patterns separated by pipes ("|") ?

100 200 202 211 212 213 214 220 300 301 400 2047 ALL XXX 10X 1XX 20X 21X
22X 2XX 30X 3XX 40X 4XX

        400

  INTERACTIVE:  Output Type, enter an integer ?
         0 = HDR information
         1 = PKT data
         2 = RPT output file
         3 = CO BIN output file
         4 = LM BIN output file
```

The user chose option 4 to output the data from this program in the same format as SREsnapshot.bin:

```
4
```

The system confirmed option 4

```
4
```

Processing began and showed the following screen display:

```
SubType = 400
Including ID= 400
74C2472C at 2
74C2472C at 1298
1296

Output file = "vc1and2_105110221.binLM.400.SRE"
```

The final line of the screen display validated that the output file was successfully created.

The user issued the following UNIX directive:

```
diff SREsnapshot.bin    vc1and2_105110221.binLM.400.SRE
```

This yielded a null response, validating no differences in the binary files.

Appendix C: Original Documentation of TDP2: TDP v1.6 Processing APID 100, 2xx, 300 and 301 Data to Level 0

A GP-B SC binary file of the same format as expected in flight, vc1and2_111132137.bin, was independently processed, first using TDP, version 1.6, and then using TERLout8.pl. The processing produced two individual binary output files. The TDP output files are in the normal course of data processing, copied directly into the GP-B Sybase database using the verified Sybase tool bcp.

The verification in Rev. A examines the binary files prepared by TDP for use by Sybase before they would be copied into Sybase.

The third-party custom-written PERL program, TERLout8.pl, was used to process the same binary SC data. APID types 100, 2xx, 300 and 301 were compared to TDP output files through Level 0 processing. Using the binary spacecraft file vc1and2_111132137.bin, TERLout8.pl compared the output of TDP against its own output.

The program TERLout8.pl was run as follows:

At the UNIX prompt, the user entered the following string:

```
{pmcgown@moc-server:63} /home/pmcgown/PERL/TERLout8.pl

*************************************************************************
  CYCLE NUMBER ? (signed integer)
               like   -9999 ... 1 ... 99   [ -99 is default ]
*************************************************************************

        -4001
```

The program responded:

```
*************************************************************************

  HYDROACTIVE:  Which DEFAULT INPUT DIRECTORY do you want to use ?
                A     /home/pmcgown/TLMdata/
                B     /home/ftp/
                C     /home/davis/gpb/tdp/
                P     /home/pmcgown/VC/
                N     (other)
*************************************************************************


        p
```

The option chosen by the user was p, use the /home/pmcgown/VC directory when looking for the source telemetry file given above (vc1and2_111132137.bin), as shown below. The system found any .bin files in the input directory and listed them for the user's convenience, then asked for a filename:

```
*  *  *  *  *  *  *  *  *  *  *  *  *  *

  vc1and2_111132137.bin

 *  *  *  *  *  *  *  *  *  *  *  *  *  *


  INTERACTIVE:  FILENAME ?
```

The user entered in the filename:

```
vc1and2_111132137.bin
```

The program found the file, then asked user if he or she wanted to check the file by reading the bottom of it.

```
INTERACTIVE:  Do you want to TAIL ( T ) or JUMP ( J ) into the TLM file or
[ N ] ?

n

N
```

The program requested an APID type for parsing (and later comparison):

```
INTERACTIVE:  Enter SubType, integer or patterns separated by pipes ("|") ?

100 200 202 211 212 213 214 220 300 301 400 2047 ALL XXX 10X 1XX 20X 21X
22X 2XX 30X 3XX 40X 4XX
```

The user chose "xxx", which requests that all packet types be processed except APID 2047, the idle packets. [This particular data set contains no APID 400 packets; that verification is addressed in the previous version of this document.]

```
xxx
```

The outputs of the TDP data were compared, so when an output type was requested, the user chose "3", the "CO" or Colorado (since TDP was created by LASP at the University of Colorado):

```
INTERACTIVE:  Output Type, enter an integer ?
          0 = HDR information
          1 = PKT data
          2 = RPT output file
          3 = CO BIN output file
          4 = LM BIN output file
    3
```

The system confirmed that we selected option 3 above, and that we wanted all APID types processed. The next output marked the beginning of formal binary processing by this program. It shows synchronization pattern and start bytes:

```
...
74C2472C at    2
74C2472C at 1298
1296
```

The system takes the difference of the two above numbers to figure out the transfer frame length. Repeat word count in this case is 1298-2=1296 (the length of the transfer frame in bytes above).

In this particular case, no Snapshot data (APID 400) was available. Snapshot verification is addressed below using another data file.

```
## of SNAPSHOTs =       0
  VC= 0  =    68096
  HC= 1  =    68096
```

After about 30 minutes, the program informed the user that the following files have been created for comparison against the data files TDP creates. The number on the left refers to the number of packets found during parsing (e.g. 226990 APID 100 data packets were found for the first file):

```
226990    /home/pmcgown/VC/vcland2_111132137.binCO.100
6001      /home/pmcgown/VC/vcland2_111132137.binCO.2xx
4         /home/pmcgown/VC/vcland2_111132137.binCO.300
68092     /home/pmcgown/VC/vcland2_111132137.binCO.301
```

If TDP is correctly parsing its data, a binary "diff" should yield zero against the above files when compared with TDP output files for the same APIDs. In the Rev. – of this document, when a binary "diff" against Lockheed data files (from the already-verified EDRS) and files in "LM" format created by TERLout8.pl was performed, a zero was also yielded. In this way, the present version of TDP is compared against an already-verified data reduction system (TERLout8.pl) using the same data files. If all differences are zero between processed files, TDP processed its Level 0 data successfully.

After running TERLout8.pl for the LASP/Colorado/TDP file format, we performed differences against the TDP files themselves: p100.tmp, p200.tmp, p300.tmp and events.tmp (the APID type 301 file). Screen results were:

```
diff p100.cycle-4001.tmp vcland2_111132137.binCO.100
diff p200.cycle-4001.tmp vcland2_111132137.binCO.2xx
diff p300.cycle-4001.tmp vcland2_111132137.binCO.300
diff events.cycle-4001.tmp vcland2_111132137.binCO.301
```

(i.e. a null response)

Appendix D: Original Documentation of TDP3: TDP v1.6 Processing APID 100 IPDU Headers Data to Level 0

Time correlation between the earth receipt time and the SC vehicle time may be required if the on-board GPS receivers fail. The process_tcor routine was appended to the .lasp software package as a contingency against this case. To verify the process_tcor routine, TERLtcor.pl was written independently to do the same process with the same data.

NOTE: Both TERLtcor and process_tcor are only to be used with VC0 (real time) files; a time correlation between earth receipt time and spacecraft time is not logical because old, previously recorded spacecraft time stamps (in VC 1 and 2 files) are not properly associated with earth receipt time.

To verify process_tcor, a real-time (VC0) GP-B spacecraft binary file of the same format as expected in flight, vc0_111132137.bin, was independently processed, first using TDP's process_tcor routine, version 1.6, and then using TERLtcor.pl. The process_tcor processing produced a binary output file which was, as it would be in the normal course of data processing, copied into Sybase using the thoroughly verified "bcp" tool. Because process_tcor's input of this vc0 file was the only input into the database thus far, an ASCII export from Sybase was done on the database.

The software-run and its analysis examines the ASCII file from Sybase (data originally entered by process_tcor) and the ASCII output from TERLtcor.pl.

The program TERLtcor.pl was run as follows.

```
{pmcgown@moc-server:137} TERLtcor.pl
```

The program responded:

```
******************************************************************
  CYCLE NUMBER ? (signed integer)
                like   -9999 ... 1 ... 99   [ -99 is default ]
******************************************************************
        -4000
```

The user chose Cycle –4000, the cycle number for the data in question:

```
******************************************************************
HYDROACTIVE:  Which DEFAULT INPUT DIRECTORY do you want to use ?
                A     /home/pmcgown/TLMdata/
                B     /home/ftp/
                C     /home/davis/gpb/tdp/
                P     /home/pmcgown/VC-TCOR/
                N     (other)
******************************************************************
        p
```

The option chosen by the user was p, use the /home/pmcgown/VC-TCOR directory when looking for the source telemetry file given above (vc0_111132137.bin). The system found any .bin files in the input directory and listed them for the user's convenience, then asked for a filename:

```
* * * * * * * * * * * * * *
  vc0_111132137.bin
* * * * * * * * * * * * * *
  INTERACTIVE:  FILENAME ?
```

The user entered the filename:

```
vc0_111132137.bin
```

The program found the file and then asked user if he or she wanted to check the file by reading the bottom of it.

```
INTERACTIVE:  Do you want to TAIL ( T ) or JUMP ( J ) into the TLM file or
[ N ] ?

n
N
```

The program requested an output type for the file (and later comparison). User requested an ASCII file:

```
INTERACTIVE:  Output Type, enter an integer ?
         0 = HDR information
         2 = RPT asc-ii file

2
        2
```

The program confirmed the ASCII file choice. Then program returned/confirmed that APID 100 packets will be used as those with spacecraft time stamps (only APID 100 data packets have vehicle times in the packet):

```
SubType =  100
Including ID=
                  100 = TLM
```

The next output marked the beginning of formal binary processing by this program. It shows synchronization pattern and start bytes:

```
...

74C2472C at    2
74C2472C at 1298
1296
```

The system takes the difference of the two above numbers to compute the transfer frame length. Repeat word count in this case is 1298-2=1296 (the length of the transfer frame in bytes above).

The program then reported that it had finished and gives the name and location of the ASCII file it produced (vc0_111132137.rpt.100.tcor.raw):

```
EXITing gracefully after 300 TCOR values
/home/pmcgown/VC-TCOR/vc0_111132137.rpt.100.tcor.raw
```

Appendix E: Original Documentation of TDP4: TDP v1.7 Processing APID 400 SRE and TRE Snapshot Data to Level 0

Document Purpose:

To update analysis report from the first revision of this document. Previously, we proved by analysis that TDP (the non-real-time Telemetry Data Processing software) is correctly processing data from raw, Gravity Probe B spacecraft binary files (such as the kind GP-B will receive in flight) to the Level 0 database. We also documented proof of correct IPDU header processing. The specification above has a Demonstration and an Analysis verification approach. The Analysis portion is needed to show that not only is our binary spacecraft data being processed to Level 0, but that it's being processed properly. This document serves as proof of accuracy of TDP code. In this revision, one change was made specifically to the TDP Level 0 processing code in Snapshot processing, and the ssbuild.pro subroutines in the Level 0 code was substantially changed. These changes only are analyzed for correctness in this document.

The other things we did were as follows (only to gpb_tdp_L0.pro):
- tightened up checks for invalid APID by breaking out and defining each APID type that is acceptable, specifically we defined the MRO 2xx series of acceptable APID numbers. Previously it was 200 -> 299.
- Eliminate any 400 packets with Snapshot_ID=0 or Snapshot_sequence=0, both of which are used as filler data from the spacecraft computer and only serve to clutter the snaptemp data table.
- added check on packet length (defined by scse-16, sec 9) per APID. The one before was a positive check, not a negative check – it only checked to see that the packet length was non zero. This was insufficient.
- Time stamp by packet type. P100 and P301 are the only packets with time stamps in the header; the time stamps in the 400 packets are only valid for the first packet of a snapshot sequence, and is taken care of by the ssbuild.pro routine. p200 and p300 have no time stamps in their headers and get time stamps only from their neighboring p100 packets per IPDU. Now, if a CCSDS time stamp is negative or greater than binary 0CCCCCCC (day 295 year 2006), we do not accept it. The VTCW time stamp will be corrupted to a negative time stamp number if we attempt to multiply it by ten, which we must if we intend to store it in Sybase. True for all time stamps. This time check is also performed on the first (sequence number =1) packet per snapshot in SSBUILD.pro for the P400 packets while snapshots are processed.
- Reports the number packets that are presumed lost based on encountering an invalid packet length. Basically, if a bad packet (bad APID or length for the APID) is found in any IPDU (transfer frame), the rest of the packet processing for that frame is abandoned. This is for data integrity. If we were to try to keep processing that frame serially, who knows where we should begin and whether they would be any good, so we just stop processing that frame. In fact, we changed processing routines to make sure they discontinued processing the frame when they encountered bad lengths and APIDs.
- New statistics presented at the end of both gpb_tdp_L0.pro (and ssbuild.pro). Even the expected file size for the various APIDs and lost packets is computed. See below for a sample. Nice inclusions are GMTs next to the vehicle times and further breakout of invalid packets encountered (and why exactly they were invalid).
- Added the accounting to break out all the statistics
- Added the /lostpackets keyword option to gpb_tdp_L0.pro so that we could optionally store all bad data that is not processed by gpb_tdp_L0.pro (all the crud we discard).

How we know we did the right thing:
- It fixed a whole lot of problems and things process now.
- Looked at all crud discarded from /lostpackets keyword. Broke out the time stamps (CCSDS versus VTCW time stamps) to prove that they should have been discarded (see p100.inv_time and p300.inv_time for proof). Gpb_tdp_L0 statistics output agree with numbers found in the inv_time files.
- Broke out invalid APIDs and examined as binary via od –tx1 (see p000.inv_apid) to be sure that they were all invalid. These are all listed out in inv_apid.lis. In fact, instances of APID 203, 207, 208, 215, for example, were found in the invalid list that would previously have been copied, incorrectly, into the MRO data table!
- Broke out the invalid packet lengths (see p***.inv_pktlen files where *** is 100, 200, 300, 301 and 400), listed them out in pktlen_lis via an od –tx pipe.
- Used TERLoutX.pl as a sanity check against the IDL output files *.tmp.

Summary Analysis:

Previously, in the first version of this document, data processing from raw binary files to "Level 0" data via the TDP (Telemetry Data Processing) system was analyzed thoroughly and compared to the already-verified Lockheed EDRS (Engineering Data Reduction System). The same binary data was run through both systems, generating independent results, and then run through a third program, TERLoutX.pl, written by Paul McGown. TERLoutX produces two sets of binary files that use the same format as EDRS and TDP, respectively. Only the snapshot portion of the LM EDRS was used for verification in this case.

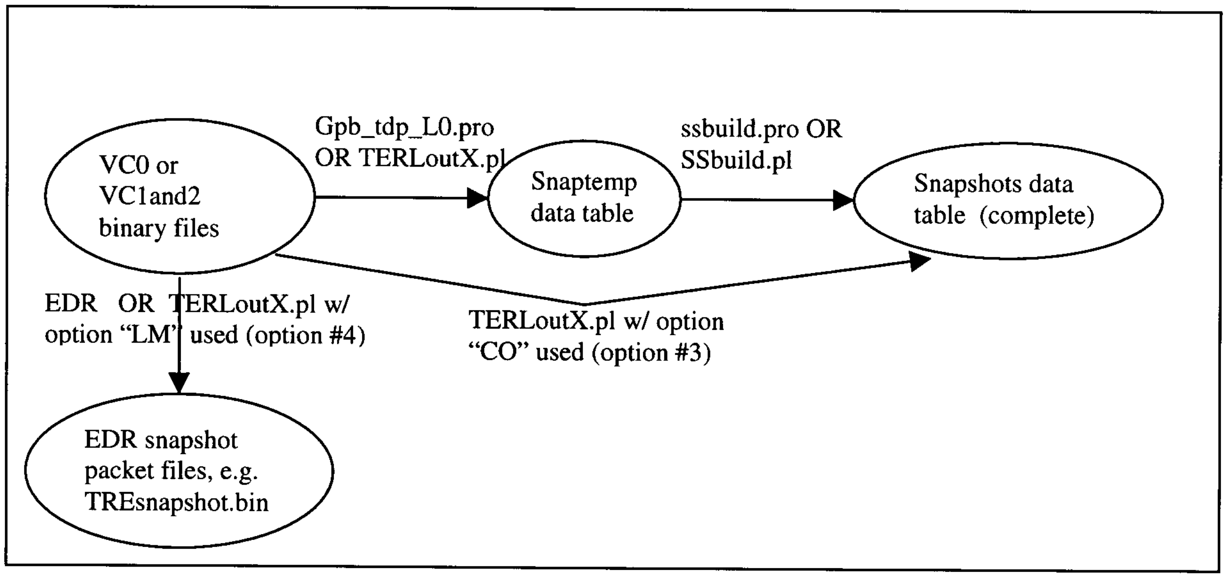
Please note that prior versions of this document verify TDP except for TRE and GSS snapshots, and that this revision only addresses snapshot processing changes required. See prior versions, rev – and rev A, for information on other verifications of TDP.

General methodology of proof of proper snapshot processing is as follows:

- Use a VC1and2 file from Sim 4 (vc1and2_207121829.bin) with all types of SRE & TRE snapshots taken. GSS snapshots were not present but design of GSS snapshot data is still in flux and cannot be verified at this time.
- Process this file through four programs: LM EDRS, ssbuild.pro, ssbuild.pl and TERLoutX.pl
- Compare results and look for zero differences for analytic success.

Specific Snapshot Analysis Follows:

Figure 1: Snapshot Processing Options for Raw Spacecraft Binary Files


VC0 or VC1and2 binary files
Gpb_tdp_L0.pro OR TERLoutX.pl
Snaptemp data table
ssbuild.pro OR SSbuild.pl
Snapshots data table (complete)
EDR OR TERLoutX.pl w/ option "LM" used (option #4)
TERLoutX.pl w/ option "CO" used (option #3)
EDR snapshot packet files, e.g. TREsnapshot.bin

-------------------------------------Part 1:

vc1and2_207121829.bin
is the SSR telemetry source file. This file was one of several dumps gathered in Sim 4, but is full of all the snapshot types we could gather during the sim, and includes all SRE & TRE snapshot IDs.

TREsnapshot.bin
SREsnapshot.bin
are the L0 raw snap packets output from Lockheed Martin's EDR package. These files were generated by running EDR on the vc1and2 file listed above. Note that there were no GSS snapshots in this VC file.

vc1and2_207121829.binLM.400.TRE
vc1and2_207121829.binLM.400.SRE
are the L0 raw snap packets output from TERLoutX.pl written by Paul McGown
to mimic the LM EDR format of TREsnapshot.bin and SREsnapshot.bin.

A UNIX diff was performed on the binary files and the TRE files and the SRE files were found to be identical, respectively. That is, there were no differences between the results of TERLoutX and LM's already-verified EDR package.

Note that TERLoutX.pl and LM EDR *snapshot.bin files are written to produce separate raw snapshot packets (not full snapshots) from raw telemetry.

---Part 2:
ss_4200.snaptemp.111662
ss_4200.snaptemp.111662.STexplode

is the output from Sybase via bcp of raw snap packets from the GPB_L0..Snaptemp table. Note that there are 111662 records. This data table represents a "way station" of partial assembly of raw snapshot packet pieces. The raw pieces are stored in the SnapTemp table before ssbuild.pro attempts to concatenate together the packets into full snapshots. The Snaptemp data table holds data for all snapshot pieces, not just the ones from the listed vc1and2 file, so its output should contain more data than the output from our vc1and2 file.

vc1and2_207121829.binCO.400.snaptemp.109005
vc1and2_207121829.binCO.400.snaptemp.109005.STexplode

is the output of raw snap packets from TERLoutX.pl written by Paul McGown to mimic the Sybase Snaptemp format, generated to the Level 0 database by gpb_tdp_L0.pro (Level 0 TDP processing). Note that there are only 109005 records.

STexploder.pl was written to explode out the SnapTemp files into a unique format that is both sorted and time sequenced.

The difference in file size is accounted for because one file is a true subset of the other (all the Sim 4 data is in Snaptemp, not just that data from vc1and2_207121829.bin). This was checked and can be re-verified if desired by a diff command or by inspection. Other than that, the file outputs were identical.

snaptemp.diff
is the difference of the aforementioned files in the form of additional SnapTemp records.

Note that visual inspection of snaptemp.diff shows time stamps which are present in the SnapTemp output records, but not in the vc1and2 file output records. There was most likely a real-time contact during that 5-8 minute period of time that accounts for the lack of data contained in the vc1and2 file.

----------------------Summary of Parts 1 and 2:

Part 1 indicates TERLoutX generates intermediate snapshot packets identically formatted to compare exactly with the output of LM Engineering Data Reduction (EDR).

Part 2 indicates that TERLoutX generates SnapTemp intermediate packets that compare exactly to the gpb_tdp_L0.pro results in the Snaptemp database table.

This proves that the results from TERLoutX are reliable and that its output may be used as verification for intermediate snapshot assembly, and that SnapTemp data table (created by gpb_tdp_L0.pro processing) contents are reliable.

----------------------Part 3:

vc1and2_207121829.binCO.400.snaptemp.109005SNAPSHOT.bin
vc1and2_207121829.binCO.400.snaptemp.109005SNAPSHOT.exp
vc1and2_207121829.binCO.400.snaptemp.109005SNAPSHOT.dbg

are the L0 complete snapshots comprised from the 109005 records of raw SnapTemp data packets, copied out from Sybase as detailed in Part 2 (above). These files were created using SSbuild.pl written by Paul McGown.

vc1and2_207121829.binCO.400
is the creation of complete snapshots from raw VC data files to full-on complete snapshots by TERLoutX.pl. These binary files (the two *.bin files above) are the same size, but binarily different due to time sequencing. One file is sorted by time and the other is by snapshot ID, so they appear different in binary. We need to "explode" these files from binary to ASCII to see if their contents differ.

vc1and2_207121829.binCO.400.snaptemp.109005SNAPSHOT.bin.SSexplode
vc1and2_207121829.binCO.400.SSexplode

are the sorted and time sequenced outputs from SSexploder.pl in the same format. They are the ordered outputs from the two aforementioned binary files. SSexploder.pl was written to explode out the SnapShot files into a unique format that is both sorted and time sequenced.

A UNIX diff was performed on these properly ordered text files and shows them to be identical.

------------------------CONCLUSION of Part 3:

SSbuild.pl and TERLoutX.pl both generate complete snapshots that when sorted and sequenced were identical, even though SSbuild is a two-step process and TERLoutX is a one-step process. Ssbuild.pl properly processes snapshot packets from the Snaptemp data table format through to completion (completed snapshots).

SSbuild.pl is a valid program for comparison against the output of the SSBUILD.pro code, as follows:

--------------------------------------Part 4:

Contents of /home/pmcgown/OFILE/SS/ on the moc-server:

```
31366200 Oct  2 18:03 SNAPTEMP.all
20918516 Oct  2 18:11 snapshots.tmp
    3255 Oct  2 18:11 snapshots.sum
    1566 Oct  2 18:11 snapshots.exp
20918516 Oct  2 18:46 SNAPSHOT.bin
     511 Oct  2 18:46 SNAPSHOT.exp
    4746 Oct  2 18:46 SNAPSHOT.dbg
```

SNAPTEMP.all
is the binary contents of GPB_L0..Snaptemp of 418216 rows for ***ALL*** SCT_Cycles (retrieved from Sybase via bcp).

SNAPSHOT.bin
SNAPSHOT.exp
SNAPSHOT.dbg
are the binary and explanations files associated with the SnapShot generation process from raw SnapTemp packets into complete SnapShots. This is the output from SSbuild.pl.

snapshots.tmp

snapshots.sum
snapshots.exp
are the binary and explanations files associated with the SnapShot generation process from raw SnapTemp packets directly from Sybase into complete SnapShots. This is the output from ssbuild.pro (IDL).

The binary output files from ssbuild.pl and ssbuild.pro both contained data reduction algorithms for all snapshot types, including GSS snapshots per bitmaps listed in SCSE-16, section 9, rev G. While final GSS snapshot bitmaps are not available, our snapshot assembly algorithm (ssbuild.pro) is at least able to process rev G versions consistently. The final bitmaps will require further verification.

A UNIX diff was performed on the binary files (SNAPSHOT.bin and snapshots.tmp) and showed them to be identical.

----------------------------------RESULT of Part 4:

SSbuild.pl and ssbuild.pro generate identical binary results despite the complexity of added checks for data continuity, and bogus SnapTemp data packets with contents such as Snapshot_Sequence=0.

<u>Therefore SSBUILD.pro is verified and ready for release.</u>

All binary files and programs used for comparison are stored on tape with the MOC configuration manager in the tape library.

Reasoning behind changes to SSBUILD.pro

Previous to lasp-1.7's release, half of the GSS snapshot IDs were not being processed at all, the GSS snapshot IDs that were being processed were being improperly processed, and TRE snapshot IDs of 52, 53, 54, 55 and 56 were being improperly processed. None of these snapshot IDs had been verified in the original TDP/TCAD buyoff because no data was available from the spacecraft for verification. After full investigation of the ssbuild.pro algorithm, SRE snapshots are suspect across different cycles. All snapshot packets in the SnapTemp data table should be re-processed, and contents of the Snapshots table replaced, by this latest version of the ssbuild algorithm, as detailed in MCR #29.

Some of the processing issues described above were due to incorrect documentation in SCSE-16, section 9, or complete lack of documentation, and some were due to continuity checks in the SSBUILD.pro routine itself, and fill data (snapshot sequence and ID =0).

Cleanups on SSBUILD.pro are as follows:

- Exclusions (2): snapshot_ID=0, snapshot_sequence=0. That is, any snapshot with ID=0 or sequence number =0 is not allowed through the gpb_tdp_L0.pro routine into the SnapTemp table. This cuts down on unneccessary storage and later confusion, since such packets are filler only and not valid. SSBUILD.pro also refuses to process them if it somehow finds them, and continues work on valid packets. This is also discussed in MCR #28.

- Four (4) continuity checks were inserted in to SSBUILD.pro. They are done in the following order: Snapshot_ID should be the same for all packets under consideration, SCT_Cycle should be the same, snapshot_sequence should be stepwise increasing, and the SCT_VTCW should have a gap of < 2 seconds between packets. If a set of snapshot packets being considered for assembly into a single snapshot does not pass these continuity checks, it will not be placed into the final Snapshots table.

- One final exemption: We have observed in data a time discontinuity. Sometimes in snapshots with a sequence number =1, the snapshot collection time is greater than 4 bytes once converted to VTCW (resulting in a time stamp beyond 2006). The cause unknown, but is probably because the box is electronically switched off while snapshots were commanded to be collected. Or could be due to bad collection method during test. This issue will be further investigated, as detailed in notes of S0613, rev B. However, for now the snapshots displaying this problem are simply rejected from Snapshots table.

- We are now processing GSS snapshots compatible with SCSE-16, Section 9, Rev G. The TRE snapshots are being processed per rev G with an additional piece of information from LM about their construction (errata in rev G were acknowledged). <u>Any new revisions of SCSE-16, Section 9 should be examined for snapshot changes, particularly for GSS and TRE snapshots.</u>

Action Items:

Need to add new Snapshot table data field “status_mode” (a four byte integer). This may have an implication to the SnapRead.m data retrieval program (addressed in MCR #41). We will also need to reprocess and repopulate the Snapshot table from the Snaptemp table (addressed in MCR #29).