W. W. Hansen Experimental Physics Laboratory
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305 - 4085

Gravity Probe B Relativity Mission

Telemetry Data Processing (TDP) in the Non-Real-Time System

GP-B Procedure
P0826 Rev D

Written by: Jennifer Spencer — Date
Data Processing

Approved by: Samantha Patterson — Date
Data Processing

Approved by: Marcie Smith — Date
Mission Operations Manager

Approved by: Ron Sharbaugh — Date
Mission Operations Center Software Manager

Approved by: Kelly Burlingham — Date
Software Quality Assurance

Tom Langenstein ITAR Assessment Performed, ITAR Control Req'd? ☐ Yes ☐ No

Table of Contents:

1. Revision History

Rev Level	Comments/notes	Date	Revised By
-	First release of this operational procedure	17-Apr-01	J Mullins
A	Initial Redlines incorporated. Permissions changes to TDP incorporated. Per ECO 1272.	6-June-01	J Mullins
B	General fleshing out of procedure, added section on transfer of SAFS files to science via secure copy (scp). Changed data processing machine from moc-server to science server. Per ECO 1315.	5 - Nov-01	J Mullins
C	Includes digital ITAR review stamp, interpretation of limit-checking software, GPS processing and IPDU time correlation software. Per ECO 1359.	2-Apr-02	J Mullins
D	Include use of IPDUsplice, tando.pro, parallel processing (if desired) and the data processing checklist. Includes information on auto_import.exp.	10-Jun-03	J Spencer

2. Scope

2.1. This operational procedure details the steps required to process Solid State Recorder telemetry data into the Level 0 and Level 1 Sybase databases using software called "TDP" (Telemetry Data Processing). These databases are detailed in S0331 and S0401, the Data Management Plan and Stanford Post-Processing Operations for Science Mission Data, respectively.

3. Operational Personnel Responsibilities and Qualifications

3.1. Operators must be competent working in a Unix environment and must understand such concepts as environment variables and working directories. Operators should be familiar with the Data Management Plan, S0331. If they are not familiar with this plan, they should read it before performing this operation.

3.2. Operator familiarity with basic commands in UNIX, Sybase and IDL is recommended.

3.3. If there are anomalies while performing this operation, these anomalies must be logged by the operator in the MOC anomaly reporting system. The operator must report anomalies to the Data Processing Lead in a timely fashion and should make appropriate notes in the checklist.

4. Requirements

4.1. Hardware and Software Requirements

Operations are performed on the Sun server machine known as "science " using either a data tape delivered from the Lockheed Martin Integrated Test Facility (ITF) or a SAFS file copied from moc-server. **The user must have his or her binary data file on moc-server or on a tape and know the filename and its full path before proceeding.** The file or file(s) to be sent into the Level 0 and Level 1 databases must be in binary form and as to be expected from the Front End Processor (FEP); that is, containing 1K, 2K or 32K format transfer frames. The data must include IPDU headers and have been recorded using the same or lower version of the LM MSS database as is presently imported into TDP on the science server in Sybase. If the present MSS database is known to be new (within a few days of official release), it is advised that the user contact the database administrator to be sure it's been imported into TDP. Sybase server must be running and IDL software must be currently licensed on the science server.

4.2. Configuration Requirements

The operator must be on the science server in any directory. **Data files awaiting processing are assumed to be in place and their file path to be known before executing this procedure**, and the MSS version used to create the data file is also assumed to be known. If the version of the MSS database used to create the data file is different from the MSS version used to decommutate the data, the file will not be properly processed. Therefore, knowing the MSS version is essential to meeting the success criteria. If the user does not know the name or location of the data file to be processed into the database, or the MSS version used to create the data file, s/he should stop now and find the file and the corresponding MSS version number. If the data file is known to be using an older release of the MSS software, the user must use special options to process the Level 1 data, as detailed in section 10.20.

4.3. Verification and Success Criteria

Success criteria for TDP is defined to be a complete and correct processing of a binary data file containing any APID 100, 200 (and 2XX, where XX is a positive integer less than 100), 300, 301 and 400 data types to the Level 0 and Level 1 databases, including snapshot processing, gps processing and (depending on user requests) time correlation processing. Proper interpretation of the status report issued at the end of each TDP process will supply all data necessary to the user for meeting the success criteria. The report informs the user of such things as the number of packets processed of each APID type, a count of any uncorrectable

Reed-Solomon errors encountered and limit checking for all monitors processed. An example report is shown and explained in detail in section 10. The user should compare the report against his or her expectations of the data file and look for any discrepancies. The user should complete all steps in the checklist given in the appendix of this document.

5. Reference Documents

5.1. Data Management Plan, S0331
5.2. Post-Processing Operations for Science Mission Data, S0401
5.3. Lockheed Martin's SCSE-16, Section 9
5.4. TCAD/TDP Version Description Document, S0503

6. Test Facilities

6.1. Mission Operations Center at Gravity Probe B, Stanford University.

7. QA Provisions

7.1. QA notification of use of this procedure is required for software releases. Otherwise QA notification is not required. The purpose of this document is to explain how to process satellite telemetry using TDP. The validation of TDP software is reviewed separately by QA.

_____ _____

Kelly Burlingham, Software Quality Assurance Date Notified

8. Test Personnel

This operational procedure is for use by the following personnel:

Jennifer Spencer

Ron Sharbaugh

Randy Davis
Paul McGown

Samantha Patterson

Kelly Burlingham (Quality Assurance)

9. General Instructions

9.1. Test operators shall **read this procedure in its entirety** and resolve any apparent ambiguities prior to beginning any tests that reference this procedure.

9.2. Any nonconformance or operational anomaly should be reported by D-Log or Discrepancy Report. Refer to the Quality Plan, P0108, for guidance. Do not alter or break operational configuration if a failure occurs; notify the data processing lead and/or quality assurance.

10. Software Operational Procedure

This section describes how to: log on to the workstation; verify Sybase server's presence; enter a proper time; load a binary file into the Level 0 and Level 1 databases; interpret the status report; and end the session. If a user is sufficiently familiar with this procedure (as determined by the data processing lead) and

it is not being run for software validation/release, the user may use the checklist provided as an appendix to this document as an alternative to running this procedure.

Start Date & Time: _____

Executed By: _____ Signature: _____

10.1. Ascertaining that the binary spacecraft telemetry file is in the right place. If this is a SAFS file (either ftp'd over from the ITF in simulation or a file from SAFS), then its proper starting location is on the moc-server under /home/safs. The user must be authorized to read /home/safs in order to see this file. If this file has not been already transferred to the "/home/tdp" directory on the science server, you the user must use the secure copy ("scp") program to move the file. It is likely you will want to put your file on the science server in a subdirectory under /home/tdp/ such as “functionals”, “thermal_vac” or “sim5”, for example. It is recommended that once you have chosen the proper subdirectory on science, you should create a new directory for your files named after the data run date (e.g. ‘Jun09_2003_18:00:30’).

To move your file from moc-server to science, you must be logged into the moc-server or one of its clients. Once you are logged in to a machine on the moc-server net go to the /home/safs directory and type:

"scp *filetomove* tdp@science:/home/tdp/"

where filetomove is the binary spacecraft telemetry file in question.

Log out of the moc-server network and move to a science machine, or if you prefer, you may ssh to science as user "tdp". To ssh, type "ssh science " and make sure you know user tdp's password.

File location:_____

____Done

10.2. Logging on to the science server as user tdp. If you are already logged on or have ssh'd over from the moc-server network, then skip to step 10.3. If you are on a science client, you still need to ssh to the server for efficiency reasons.

Enter science server console login: tdp

Enter password: [user tdp's password entered here]

____Done

10.3. Looking for Sybase.

Open a terminal (window) and look for your science server prompt: {tdp@science}

Type “showserver” at the prompt. Expect to see four (4) instances of Sybase server, one of which is a backup server.

```
{tdp@science:4} showserver
UID   PID  PPID  C    STIME TTY      TIME CMD
root  4696  4695  0   Apr 10 ?        1:07 /apps/licensed/sybase11.9.2/bin/dataserver -
sscience_server -d/dev/rdsk/c1t2d0s3 -e
root  4693  4692  0   Apr 10 pts/7    0:00 /apps/licensed/sybase11.9.2/bin/backupserver -
Sscience_server_back -e/apps/licensed
root  4805  4696  0   Apr 10 ?        0:01 /apps/licensed/sybase11.9.2/bin/dataserver -
ONLINE:1,0,0x101a92e,0xdfc00000,0x1
root  4806  4696  0   Apr 10 ?        0:01 /apps/licensed/sybase11.9.2/bin/dataserver -
ONLINE:2,0,0x101a92e,0xdfc00000,0x1
```

If only one line of all-caps data is returned:

```
UID   PID  PPID  C    STIME TTY      TIME CMD
```

that means that the Sybase server is not running on this machine and may need to be restarted. Present working directory is irrelevant to the expected output from Sybase and all users can issue the showserver command.

The only reason the user might see the above line when in actuality Sybase is running is the user is logged into a client machine and not into the server running Sybase. Make sure you're actually on the machine "science" and not one of its clients before becoming concerned. Call the database administrator or the systems administrator for assistance if Sybase is down. If the Sybase server is running, proceed to the next step.

____Done

10.4. If one is planning to enter a file using a spacecraft clock time of zero other than noon on January 1, 2000, a new cycle number and corresponding clock "zero" time will need to be entered into the SCtime table in Sybase. The user will need to contact data processing in case of a new spacecraft zero time. All pre-launch telemetry files will probably require a new cycle number assignment, so **be sure you have a new cycle number ready from data processing before you continue with this procedure.**

Cycle Number: _____

____Done

10.5. Run IPDUsplice. This program must be run if you expect spacecraft clock "hiccups", resets or jumps in this data file; it will separate out the various sets of data into files by time segment. If you do not expect spacecraft clock issues, but you know the data may be rough or contain serious errata, run IPDUsplice to clean the file.

10.5.1. Do an "ls" on the directory where your data to be cleaned/spliced resides.

10.5.2. Change directories to /apps/supported/lasp/src/tdp/ if you are not already there, and set the following environment variables.

- setenv GPBDB /apps/supported/lasp/src
- setenv IDL_STARTUP idl_startup.pro

Then type idl –32 at the prompt.

10.5.3. Once IDL is started, type " .run IPDUsplice " at the prompt.

10.5.4. You should see a few modules compiled. Type "IPDUSCRUB, '/home/tdp/filetobespliced' at the prompt, where "/home/tdp/filetobespliced" is the full path and filename of the telemetry data file you need cleaned or spliced.

10.5.5. IPDUsplice will scrub your file and return your summary results in the directory where your data resides. It creates two summary files and it may create one or more data files. The vc*.is_exp file is an explanation of the times found in your data, listed by splice file. The vc*.is_sum file is a full summary of the cleaning that was performed and any packet errors found. The vc*.is000N (where N is 1 or greater) file(s) are your data files for further processing.

10.5.6. If IPDUsplice shows more than one splice, read the explanation and summary files and judge how best to process data. If, for example, there is data starting at 0 seconds and continuing for 30 minutes, then data that starts at year 2001 day 247 at 07:30 and goes for eight hours, you probably have functional data.

Different sets of data have different processing rules; there are too many to detail in this document, and they often change from test to test. If you are uncertain about what to process, ask the user who

supplied the data or requested you process it what they want to see. If there is a small gap, such as a ten-minute gap, found, process all files and include discussion of the gap in your summary report to the users. Always include any data removal or known gaps/errata to the users.

If IPDUsplice gave only one splice, check the filesize of the splice file against the size of the original file. If there are no differences, remove the splice file for housekeeping reasons and process the original. If there is a size difference, process the splice file because it will be cleaner data, and inform the users in your summary.

IPDUsplice Required? Yes No ____Done
IPDUsplice Run? Yes No ____Done
Number of splices received? _____ ____Done
File processed (circle one): Original Splice(s) #s_____ ____Done

10.6. Automated processing. Run this subsection only if automated processing is desired. Automated processing incorporates steps 10.8 through steps 10.28, but you must still check the data summaries as indicated in step 10.6.4.

10.6.1. Create a cmd file. Create a file in your data directory named “cmd” (or “cmd2” - et cetera - if you have more than one cmd file in your data directory). The content of the cmd file should have multiple lines, one for each data file, and those lines must contain the following content:

vcfile cyclenumber mssid -options

where **vcfile** is the full path and filename of each data file (or the name of your splice file)
cyclenumber is the cycle number assigned (see step 10.4)
mssid is the current MSS delivery ID in hex (see step 10.20 in Level 1 processing below). Note that omitting the MSSID will force the program to default to the latest MSS build (the highest hex value of MSSID available in TDP’s database)
and **–options** include any of the following options:

CO_0	Run level 0 data as checkonly.
CO_1	Run level 1 data as checkonly.
ENGONLY	Imports only the engineering data in level 1.
SCIONLY	Imports only the science data in level 1.
DEBUG	Run level 0 and level 1 as check only. Some increased output.
BACKUP	Backs up the .TMP files to the /home/tdp/auto-CYCLE directory.
AUTO	Skips checking of Files and SCtime table. Uses an alternate data import path named 'auto_MonthDD_HH:MM:SS' for generation to TCAD files, so that it will not conflict with any user-commanded imports and so that multiple imports may be ran concurrently. Please note that at this time, these directories must be cleaned up manually.
SCTIME	Generates a placeholder entry in the SCtime table for this cycle (if it is a new cycle).
EMAIL	This flag sends an email to a pre-defined user list for EACH file in the CMD as soon as it completes. Attached to the message are the Level 0 and Level 1 summary reports and the Post Check summary.

Flags in the CMD file affect ONLY the file they are applied to; you can chose different options for different vcfiles line by line. Flags in auto_import are NOT case sensitive and may occur anywhere in the line.

Some valid examples of cmd files include:

vc1and2_30714181603.bin -4060 -co_0
-co_0 -co_1 -BACKUP -SCTIME vc0_30715141602.bin -4060 13312

cmd file name: _____

____Done

10.6.2. Run the cmd file. First, log your data import. Begin your import log file (run step 10.7 below). Then change directories to /apps/supported/lasp/src/tdp and type:

auto_import.exp N fullpath_and_name_of_cmd_file

where N is the number of hours to delay the start of processing.

____Done

10.6.3. Respond to the possible queries of auto_import.exp. Unless you use the –Auto keyword, if the data cycle you have entered in your cmd file has already been entered in the SCtime data table, auto_import.exp will query you about whether or not to add your data to it. This is to keep data from being accidentally merged, so if you know this is the proper cycle number, respond with a "yes". When auto_import.exp has been successfully started, you will see a screen message that says: "LATE NIGHT RUN OF LEVEL 0/LEVEL 1 DATA PROCESSING. DO_NOT_TOUCH" – even if it's not late at night.

____Done

10.6.4. Leave the window alone. When processing is complete (we are processing data at roughly a 12:1 ratio – that is, 12 hours of data can be processed in about an hour or less), check your import file as directed in steps 10.13 and 10.23 for interpreting the Level 0 and Level 1 processing summaries, respectively. Fill out the "done" box in those sections as appropriate.

10.7. Log your import. In the directory where you put your data (see step 10.1), type "script *filename*" where *filename* is the name you have give your data log. Data processing recommends names such as "import.log" or "import1.txt", for example. This step is critical for debugging any problems you may encounter and is **required** for automated data processing.

Import log file name: _____

____Done

10.8. The following steps (all following steps) apply to manual processing and include full data summary interpretation.

Start the telemetry data processing and import Level 0 data.

Present working directory is not relevant and is at user discretion

Type "tdp" at the Unix prompt.

10.9. You should see information similar to the following on the monitor:

```
IDL Version 5.4 (sunos sparc). Research Systems, Inc.
Installation number: 93897-0.
Licensed for use by: Stanford University

For basic information, enter "IDLInfo" at the IDL> prompt.

% Compiled module: NEW_DT.
```

```
% Compiled module: DT_ADD_SECS.
% Compiled module: DT_DIF.
% Compiled module: CLOCK_TO_DT.
% Compiled module: DT_TO_STRING.
% Compiled module: STRING_TO_DT.
% Compiled module: CURRENT_DT.
% Compiled module: NEW_SCT.
% Compiled module: NEW_XT.
% Compiled module: SCT_ADD_SECS.
% Compiled module: SCT_DIF.
% Compiled module: DT_TO_SCT.
% Compiled module: SCT_TO_DT.
% Compiled module: DB_CONNECT.
% Compiled module: DB_DISCONNECT.
% Compiled module: DB_SELECT_PACKETS.
% Compiled module: DB_GET_PACKETS.
% Compiled module: DB_SELECT_EVENTS.
% Compiled module: DB_GET_EVENTS.
% Compiled module: DB_SELECT_TMDISCRETE.
% Compiled module: DB_GET_TMDISCRETE.
% Compiled module: DB_SELECT_TMANALOG.
% Compiled module: DB_GET_TMANALOG.
% Compiled module: DB_SELECT_TMAVERAGE.
% Compiled module: DB_GET_TMAVERAGE.
% Compiled module: DB_GET_TMINFO.
% Compiled module: DB_GET_SUBSYSTEMS.
% Compiled module: DB_GET_NAMES.
% Compiled module: DB_GET_MNEMONIC.
% Compiled module: DB_GET_TMDECOM.
% Compiled module: DB_GET_TMAVGID.
% Compiled module: DB_GET_CALIBRATION.
% Compiled module: DB_GET_STATES.
% Compiled module: DB_GET_LIMITS.
IDL>
```

Once you get to the waiting IDL prompt, type in:

```
>.run gpb_tdp_L0
```

This will compile the routines needed to import the level 0 data.
You should receive the following confirmations of compilations with possible additions:

```
% Compiled module: READ_IPDU.
% Compiled module: GET_EVENTS.
% Compiled module: PROCESS_TM_FILE.
% Compiled module: FETCH_PACKETS.
% Compiled module: L0_PROCESS_S100.
% Compiled module: L0_PROCESS_EVENTS.
% Compiled module: OUTPUT_STATS.
```

10.10. At the next prompt type in the routine name to process the telemetry file and the name of the binary file to be processed. Enter a full path within the quotes below along with the file name. The user must have read-access to the file before it can be successfully processed. Always run the vc1and2 (the SSR file) before running its associated vc0 file. TDP can be run in parallel if needed, but you must know whether or not an instance of TDP is presently running first. Running in parallel gains the user very little speed in processing and should only be done if absolutely necessary. To determine whether or not there is an instance of TDP running, run “ps –ef | grep idl” on the science server. If your results have a line including “idl –32” at the end, there is likely another instance of TDP running. You should either check with other members of data

processing before continuing, have the database administrator run "sp_who" in Sybase to look for jobs running under the "gpbops" user, or you should use parallel processing options (see the next section for more information on parallel processing).

Other TDP instances running? Yes No ____Done

If the project is in test (pre-launch), each separate data file processed will likely need its own unique cycle number. If one test is continuing across several data files without a spacecraft clock reset, the data files will all use the same cycle number. See step 10.4 above regarding cycle numbers. **To get a new cycle number for your file, please contact data processing.** In the following example, the cycle number is –??, but all pre-launch data sets are assigned their own negative cycle numbers. TDP uses cycle number 0 if one is not entered at the prompt. Since there is very likely data already in the database with cycle number 0, user data may be rejected due to uniqueness violation (or "duplicate key" as it's known to Sybase).

```
IDL> PROCESS_TM_FILE, "/home/tdp/functionals/Oct_3_2002/vc0_002082125.bin",
cycle=-?? (enter your assigned cycle number here)
```

Some keyword options exist for process_tm_file. Specifically, If you want to see a more detailed screen log during this process, you may type /debug after the command above. Or to do all processing without actually copying the data into Sybase, use the "/checkonly" keyword at the end of the command above. The /checkonly option is useful when you need a data summary in a hurry or you want to see if you may need to run IPDUsplice on the file for data quality reasons before putting the data into the Level 0 database. These options are not case-sensitive. An example is listed below:

```
IDL> PROCESS_TM_FILE, "/home/tdp/functionals/Oct_3_2002/vc0_002082125.bin",
cycle=-1000, /DEBUG, /checkonly
```

Level 0 TDP Data Processing started? Yes No ____Done

10.11. Parallel processing is done via the program gpb_tdp_setdir.pro. To activate it, type "tdp" and issue the command ".run gpb_tdp_setdir". Once it has compiled, issue the command: GPB_TDP_SETDIR, "*directoryname*" where *directoryname* is the name of the alternate directory in which you want your temporary files created. Note that for each parallel TDP process you wish to run, you must create a parallel data processing directory. By default, TDP creates temporary files in the directory /apps/supported/lasp/src/data . Running TDP in parallel without running the setdir program would overwrite existing (and in-use) temporary files, causing several problems for all instances of TDP. Note that you must have write permissions to *directoryname*'s path or this process will not work.

Parallel Processing Temporary Data Directory _____

Run gpb_tdp_setdir.pro? Yes No _____Done

10.12. The Level 0 processing time is rather short. Most files take 5-10 minutes to go through Level 0, including all copying to Sybase. After a few minutes, you should see something resembling the following example returned to the screen:

```
Extracting frames from GPB telemetry file vc0_002082125.bin ...
Fetching frames from GPB IPDU file and packets from frames

Processing programmable telemetry packets into database...

Starting copy...

Batch successfully bulk-copied to SQL Server.
1000000 rows sent to SQL Server.
```

```
Batch successfully bulk-copied to SQL Server.
2000000 rows sent to SQL Server.
```

[A list of similar messages follows. TDP is copying the Packet data into Sybase 1,000,000 rows at a time.] At the end of the message you'll see something like:

```
10000640 rows copied.
Clock Time (ms.): total = 19000 Avg = 0 (5296.84 rows per sec.)
```

[The number of rows and clock time will vary depending on the size of the file and data content recorded. "Total" time listed is in milliseconds – 19000 milliseconds in this example.]

```
Processing MRO packets into database...

Starting copy...

4591 rows copied.
Clock Time (ms.): total = 1000  Avg = 0 (4591.00 rows per sec.)
```

[If Memory ReadOut packets are included in the data set, you will see a non-zero number of rows copied into Sybase. Again, clock time and number of packets vary depending on data content.]

```
Processing DBRO packets into database...

Starting copy...

0 rows copied.
Clock Time (ms.): total = 1
```

[In this example, no Database ReadOut packets were in the data file, so zero rows were copied to Sybase.]

```
Processing raw snapshot packets into database...

Starting copy...

Batch successfully bulk-copied to SQL Server.
10000 rows sent to SQL Server.
Batch successfully bulk-copied to SQL Server.
20000 rows sent to SQL Server.
```

[This may continue for some time]

```
Batch successfully bulk-copied to SQL Server.
170000 rows sent to SQL Server.
Batch successfully bulk-copied to SQL Server.
180000 rows sent to SQL Server.

180135 rows copied.
Clock Time (ms.): total = 43000  Avg = 0 (4189.19 rows per sec.)
```

[In this example, approximately (180,135)/3 Snapshot packets were in the data file. TDP divides the raw packets into subpackets (where there are 3 subpackets per APID 400 packet) before copying the data into the database, for ease of later re-assembly. Normally, one would expect three times the number of snapshot packets detected in the later summary to match the number of rows (or subpackets) copied to Sybase. However, there is some editing done by TDP about which subpackets are legitimate. More on this at the summary discussion in step 10.13 below.]

```
Processing event records into database...
```

```
    Starting copy...
    Server Message:  - Msg 3604, Level 10, State 0:
    Duplicate key was ignored.

    Batch successfully bulk-copied to SQL Server.
    200000 rows sent to SQL Server.
    Server Message:  - Msg 3604, Level 10, State 0:
    Duplicate key was ignored.

    Batch successfully bulk-copied to SQL Server.
    400000 rows sent to SQL Server.
    Server Message:  - Msg 3604, Level 10, State 0:
    Duplicate key was ignored.

    bcp copy in partially failed

    91 rows copied.
```

[Frequently, due to the nature of the GP-B telemetry, events are repeated. The flight computer will stuff the same event report into its telemetry stream until the event changes, so we expect to see many duplicates of no meaningful value. TDP presently relies on Sybase's unique index to weed out these duplicates. Because of that, the user should expect to see "Duplicate Key was ignored" in the Event processing report. Sybase considers that its bulk copying utility has partially failed if duplicate "key" or index messages are issued, but there is no cause for concern; TDP is functioning correctly in this case. In this example, there were only 91 unique events, so only 91 rows went into Sybase.]

10.13. Eventually you should see a data summary resembling the following (this example may not be consistent with the example in the step above):

```
Level 0 Data Processing Summary (Version-1.8):

  FILE= /home/tdp/functionals/Mar20_2003_04:04:47/vc1and2_303192310.bin

  Earth Receipt Time (ERT) from IPDU
  First        2003/078-23:12:44.0
  Last         2003/078-23:25:13.0
  Min          2003/078-23:12:44.0
  Max          2003/078-23:25:13.0


  SpaceCraft Time (SCT) from P100 - aka VTCW
  Cycle                                -13622
  Min          2001:249:07:35:57.1     530337571
  Max          2001:249:13:05:47.0     530535470

       74384  IPDUS Processed
           0  IPDUS with errors in header
           0  Frames with uncorrectable errors
           1  Frames with packet errors (abandons)
           7  Gaps in VCDU sequence
              >=         1040 frames in the gaps (by VCC mod 256)
              and hence it may be greater than this number of frames

     Occurs   VCID
    --------  ----
       57731     1
```

```
    16653     2


 Packets   Valid                              Records     File Size
---------  -----------------              ----------   ----------
   247950  Programmable TLM  (APID  100)     247950     48350250
    17513  Memory ReadOut    (APID  2xx)                 3590165
     1354  DataBase ReadOut  (APID  300)                  151648
    73029  Event             (APID  301)     841414      6731312
     6569  Snaptemp          (APID  400)      14729      1104675
    99888  Idle              (APID 2047)

 Packets   Invalid
---------  -----------------
        1  Unrecognizable    (APID xxxx)                     128
        0  Improper length   (APID  100)                      0
        0  Improper length   (APID  2xx)                      0
        0  Improper length   (APID  300)                      0
        0  Improper length   (APID  301)                      0
        0  Improper length   (APID  400)                      0
        0  Improper length   (APID 2047)
        0  Without P100 time (APID  2xx)
        0  Without P100 time (APID  300)
        0  Without P100 time (APID  400)

 Packets   Abandoned
---------  -----------------
        1  Packet Size  98   (APID  3xx)
        0  Packet Size 192   (APID<>3xx)

 Records   P100 ERROR Types
---------  ----------------
        0  TOTAL ERRORS                                        0
        0  VTCW lt        0
        0  VTCW ge 7fffffff
        0  SF_Format_ID   eq   0
        0  SF_Format_ID   invalid
        0  SF_Frame_Count eq   0
        0  SF_Frame_Count gt 100
        0  SF_Frame_Count gt xxx for a valid SF_Format_ID

 Records   P301 ERROR Types
---------  ----------------
        0  TOTAL ERRORS                                        0
        0  VTCW lt        0
        0  VTCW ge 7fffffff

 Records   P301 DATA Zeroes
---------  ----------------
   327050  VTCW, AppNum, EvtNum (each=0)

 Records   P400 DATA Zeroes
---------  ----------------
     1926  ID number = 0
     3052  Sequence  = 0
```

```
P100 OCCURs  Time Hiccups (if any)
-----------  ---------------------
          0  VC0 - CCCA reset
          0  VC0 - TIME rewind
          0  VC0 - TIME neg leap
          0  VC0 - TIME pos leap
          0  SSR - CCCA reset
          0  SSR - TIME rewind
          0  SSR - TIME neg leap
          0  SSR - TIME pos leap

P100 occurs  FMT
-----------  ---
     184621  202
      63329  216



P100 occurs  MSSid  MSSid      MSSid
-----------  -----  -----      -----
       2480  x3401  13313  =>  13312

SSid  P400s
----  -----
   1    690
   2    690
   3    690
   4    760
  11    966
  12    966
  13    966
  14   1057
  51   5883
 110    128
 111    128
 112    128
 113    101
 114    101
 115    101
 120    128
 121    128
 122    128
 123    101
 124    101
 125    101
 130    128
 131    128
 132    128
 133    101
 134    101
 135    101

    6.40  Minutes spent parsing the VC file into Level 0 packets
    0.33  Minutes spent BCPing APID 100 - Programmable TLM
    0.02  Minutes spent BCPing APID 2xx - Memory ReadOut
    0.00  Minutes spent BCPing APID 300 - DataBase ReadOut
    0.41  Minutes spent BCPing APID 301 - Event
```

```
    0.02  Minutes spent BCPing APID 400 - Snaptemp
    7.19  Minutes total run time

Complete at 2003:114:18:04:10
```

Proper interpretation of this report is important.

The first paragraph describes the timestamps on the data. ERT (Earth Receipt Time) is the UTC of the first bit of the transfer frame in the data file, and is listed using the day-of-year date convention. The SCT is the Spacecraft Clock Time listed in the packets multiplied by 10. Listed with the SCT (times 10) is the corresponding cycle number (-13622 here). The two combined make a unique key for all data in the Level 0 database.

The second paragraph shows data for the IPDUs (Internet Protocol Data Units). This paragraph shows how many IPDUs were in the file, whether there were any with errors in the headers (such as the wrong spacecraft number associated with the packet), if there were any uncorrectable Reed-Solomon errors (TDP won't process packets with uncorrectable errors), if there were any frames with packet errors (such as a packet with too many or too few bytes for its packet type which are abandoned when encountered) and any gaps in the VCDU sequence (see SCSE-16, section 9 for more information about this sequence).

The third paragraph describes how many IPDUS were found in Virtual Channels 0, 1 and 2 and lists those numbers for the user.

The fourth paragraph of the summary describes the contents of the data file and how many of each packet type were found. This is important in determining the success of final processing. The lines here listing the numbers of packets should square exactly with the number of rows "bcp'd" into Sybase step 10.11 above. If not, the total number missing should be made up by a non-zero number in the "Packets Invalid" paragraph, or those following it regarding abandoned packets and errors reported. The only exception to this would be the repetitive Event data packets as detailed above in step 10.11 and in the Snapshot processing. Snapshots are entered into the Level 0 database Snaptemp table as "subpackets" which are one third of a full packet. This is done to ease later assembly of full snapshots by the ssbuild utility described in step 10.14 of this document. The number of rows bulk-copied into Sybase should be three times the number (at most) of Snapshot Packets shown in the summary. The number bulk copied may be slightly less than three times the number of Snapshot packets shown in the summary because APID 400 data Snapshot packets with a snapshot ID number of zero are not copied into the database. You may have some of these ID zero packets in your data – if so, they will be counted in the "`Records P400 DATA Zeroes`" column. If you have more than three times the Snapshot Packets found listed as rows bulk-copied into Sybase, there is a problem potentially with the data. Please notify data processing and other ops personnel as appropriate. If the number of Programmable tlm packets (here, shown as 247950) listed in the summary is not the same as the number at the end of the bcp messages seen during the "`Processing programmable telemetry packets into database...`" step then there may be a problem. If "duplicate key" errors were seen during the programmable packet bcp, the discrepancy may be explained as Solid State Recorder wraparound, or overlap of data already existing from the last file processed in this cycle. If either of these options are not possible for this import, then there's a different problem somewhere requiring resolution and this run of TDP was not successful. Similarly with DBROs, MROs and Snapshots: the numbers in this step and step 10.11 must be equal unless there was Solid State Recorder wraparound or data overlap to explain the differences.

The fifth through tenth paragraph describe all possible errata in data. Ideally, TDP should never have "Unrecognizable packets" (the last line of the programmable packet "Packets Invalid" summary). That category is intended as a last-resort catch-all for the TDP software should something unforeseen occur. If there are non-zero numbers shown in the "Unrecognizable packets" row, save the summary and as much of the data run screen information as possible into a text file or a printout and alert the data processing team. In ground testing, this type of data has been seen periodically, and is usually a result of bad RF or human intervention (e.g. people handling umbilical cables during data recording). All other errors found in processing are in the report and are self-explanatory.

The eleventh paragraph, "`P100 OCCURs  Time Hiccups (if any)`", describes any possible vehicle clock jumps found during the data file. If time jumps occur, it may be necessary to examine the data file further using IPDUsplice.

The twelfth paragraph lists the programmable telemetry data formats that were found in this file. The thirteenth paragraph lists the MSSID found in this telemetry, and shows the one that should be used to process the file to Level 1 (e.g. 13313 => 13312 implies that the MSSID found was 13313, and the closest MSS build in hex is 13312).

The next paragraph shows how many Snapshot packets were found for each Snapshot ID in the P400 data, and lists out all that were found in this file.

The last paragraph shows how long each set of bcp's took to complete.

After the summary, you will be returned to the prompt.

```
IDL>
```

Level 0 Data Summary Reviewed? Yes No ____Done
Level 0 Errata present? Yes No ____Done

10.14. To correctly process Snapshot data, if there are any Snapshot packets listed in the above summary, you will need to run the ssbuild routine.

Snapshot data present in this data set? Yes No ____Done

At the prompt, type ".run ssbuild"

```
IDL> .run ssbuild
```

You should see the following module compiled:

```
% Compiled module: SSBUILD.
```

At the prompt, type SSBUILD to run the Snapshot building utility for the APID 400 packets. There are keyword options for this program, including the /CHECKONLY and CYCLE=N option. If a specific cycle number is indicated for processing, ssbuild will comprise snapshots only for snapshots of that cycle number. Otherwise, it will try to build snapshots in every cycle number. It is recommended that you use ssbuild only for your cycle, as shown below.

```
IDL> SSBUILD, Cycle=-1000
```

The ssbuild.pro routine generates three summary files. These files are snapshots.sum, snapshots.pktsum and snapshots.idseq. The snapshot summary file details how many snapshots were gathered, their types, groups and the number of packets used to process them, broken out by cycle number and snapshot ID. Snapshots.pktsum details the status of all packets that includes any dropped or skipped packets and the reasons why, broken out into cycle number. Snapshots.idseq is a listing of snapshot IDs including the number of packets, sequence number, Snaptemps, etc. Details are found in S0401. When running ssbuild, you should receive a lengthy on-screen summary describing the number of packets and snapshots built per snapshot ID, the number of errata and the following:

```
Starting snapshot generation
Snapshot generation complete
     1201 snapshots total
     1201 SRE snapshots
       0 TRE snapshots
```

```
       0 GSS snapshots
     20535 Snapshot packets not processed

Starting copy...
     1000 rows sent to SQL Server.
     1201 rows copied.
     Clock Time (ms.): total = 1201  Avg = 16 (61.60 rows per sec.)
```

10.15. Interpreting the summary is straightforward. The total number of snapshots listed in the first line should be the sum of the next three lines (number of SRE, TRE & GSS snapshots). The number of rows sent to SQL Server should be the same as the total number of snapshots (1201 in this example). The number of snapshot packets not processed may include duplicate data, snapshots with gaps, filler data, snapshots with a sequence number of zero, incomplete snapshots or snapshots given an invalid "type" (or "id") number, including id 0. The ssbuild routine will only process snapshot types: {[1,8] ∪ [11,18] ∪ [51,56] ∪ [101,106] ∪ [110,115] ∪ [120,125] ∪ [130,135]}. Please see S0401 for more details if there are problems apparent from the screen report and notify data processing.

Successful run of SSBUILD? Yes No ____Done

10.16. To correctly process IPDU header data, which should only be done if a GPS solution could be resolved (that is, during flight or special pre-flight test), you will need to run the process_tcor routine. **Please note that you should only process IPDU data on VC0 (Real-time) files.** The process_tcor routine will bounce your request if you try to use any files that do not contain VC0 data. To process a VC0 file's IPDU headers, at the prompt, type ".run process_tcor". You may use the /checkonly keyword if you want to run the program without copying data into Sybase. You may also wish to use the /RAW or /FIT keywords: /RAW enters all data into the database and /FIT enters the least-squares fit information only into the database. Please see S0401 for more details.

Process_tcor data needed? Yes No ____Done

```
IDL> .run process_tcor
```

You should see the following modules compiled:

```
% Compiled module: READ_IPDU.
% Compiled module: GET_FRAME.
% Compiled module: GET_PACKET.
% Compiled module: GET_EVENTS.
% Compiled module: PROCESS_TM_FILE.
% Compiled module: FETCH_PACKETS.
% Compiled module: L0_PROCESS_P100.
% Compiled module: L0_PROCESS_P200.
% Compiled module: L0_PROCESS_P300.
% Compiled module: L0_PROCESS_SNAPS.
% Compiled module: L0_PROCESS_EVENTS.
% Compiled module: OUTPUT_STATS.
% Compiled module: PROCESS_TCOR.
% Compiled module: FETCH_TCORS.
% Compiled module: FIT_TCOR.
% Compiled module: TCOR_RAW_TO_DB.
% Compiled module: TCOR_FIT_TO_DB.
% Compiled module: OUTPUT_STATS.
```

To process your IPDU data, you will need the file name of the binary data that includes the IPDU times. This is the same name as the file referred to in step 10.1.

```
IDL> process_tcor, '/home/tdp/ vc0_002082125.bin', CYCLE=-99

Extracting frames from GPB telemetry file /home/tdp/vc0_002082125.bin

Starting copy...
300 rows copied
Clock time(ms.): total =1 Avg = 0 (300000.00 rows per sec.)

Data Processing Summary:

ERT of First IPDU:       2001/073-00:03:10.8
ERT of Last IPDU:        2001/073-00:00:55.5
SCT of Earliest P100 Packet:     -99,   371378111
SCT of Latest  P100 Packet:     -99,   371478750

         450 IPDUS processed
           0 IPDUS with errors
         450 Good frames processed
         300 Time correlations extracted
```

10.17. Interpreting the data summary. The ERT of the first and last IPDU should be consistent with user expectation based on the previous summary of Level 0 data (all reported IPDU times for the same file should be the same). SCT refers to spacecraft time of the earliest and latest packet processed. According to the above example, 450 IPDUs were processed from the pass, there were no errors found (any IPDUs with errors will not be processed to the database) and 300 time correlations were extracted. The maximum number of time correlations that may be extracted is 300. You will only see the 'Starting copy...' set of 3 lines if you included the '/RAW' keyword in your parameters to process_tcor.

Successful run of process_tcor? Yes No ____Done

10.18. The Level 0 data hopefully has been successfully imported. If it has not been a successful import thus far, you may wish to abort the rest of the procedure. If the failures did not appear to affect any programmable telemetry packet data (number of rows copied matched in the summary and there were no unexpected Sybase error messages), you may continue to import the Level 1 data. If you wish to abort the procedure, type "exit" at the IDL prompt.

Level 0 processing result: PASS FAIL (circle one) _____ initials

10.19. Start the telemetry data processing to import Level 1 data.

If you exited IDL and are returning start TDP and IDL again by typing "tdp"

You should get the same compiled modules as detailed above in section 10.9. Go to the next line and type the .run command listed.

If you have not exited IDL, then

Run the following at the prompt:

```
IDL> .run gpb_tdp_L1
```

to import Level 1 data. Expect to see some of the following modules compiled:

```
% Compiled module: PROCESS_PTM.
% Compiled module: BUILD_DECOM.
% Compiled module: DECOM_TLM.
% Compiled module: THIN_DISCRETE.
% Compiled module: L1_PROCESS_ANALOG.
% Compiled module: L1_PROCESS_DISCRETE.
% Compiled module: OUTPUT_STATS.
```

Upon returning to the IDL> prompt, type the following to compile the limit checking procedure (if desired):

```
IDL> .run post_check.pro
% Compiled module: POST_CHECK.
% Compiled module: LIMIT_CHECK.
% Compiled module: STATE_CHECK.
```

10.20. At the next IDL> prompt, you will import your file. The file you want to operate on was created by the Level 0 routines and is named "p100.tmp" because it's a temporary file full of 100 series packets. If you are doing an export from the Level 0 database (see P0911 for reprocessing data to full resolution), your filename will be packet100.tmp instead, though its format will be the same. To run your series 100 packets file through tdp and add your data to the Level 1 database, type:

```
IDL> PROCESS_PTM, "p100.tmp"
```

If you want to do limit-checking on the data, be sure to type the keyword "/POSTCHECK" after your initial command:

```
IDL> PROCESS_PTM, "p100.tmp", /POSTCHECK
```

If you want to see a detailed screen log during this process, you may type instead:

```
IDL> PROCESS_PTM, "p100.tmp", /debug
```

Or to process the file without actually copying the data into Sybase, use the "checkonly" command.

```
IDL> PROCESS_PTM, "p100.tmp", /CHECKONLY
```

Keywords /engonly and /sciony are also available. Engonly processes 1K & 2K data formats only and skips 32K data formats while decommutating. Sciony processes only 32K data formats and skips the 1K and 2K data. This is particularly useful when you have a mix of both data in the same input file, because otherwise, new decom maps have to be built every time a new format is encountered (which may be as often as every few seconds). It is recommended that if you have a mix of data types, you run the process twice: once with /sciony and then once with /engonly.

Data will be processed using TDP's most current MSS database build by default. If you do not want it to use that build, you'll need to use the MSSID= nnnn keyword as detailed below.

MSSID used:_____ ____Done

What to do about older data files and the MSS version number:

If you need to run older data using a different MSS build than the most current one in Sybase (for example, the present MSS is 3.2.5, but you are processing a file created two months ago using MSS 3.2.4), you'll need to add the number of the MSS build in hex to your PROCESS_PTM command:

```
IDL> PROCESS_PTM, "p100.tmp", MSSID=12864
```

12864 is an example relevant only to MSS build 3.2.4.0 (12864 = 3240 in hex). To get the correct value of your "MSSID" variable, type at the prompt:

```
IDL>print, 'nnnn'x
```

Where nnnn is your MSS build with the periods removed and an added zero at the end (for MSS 3.2.4, nnnn should be 3240). The print statement will return the hexidecimal value of nnnn at the prompt (which in this case is 12864).

Level 1 TDP Data Processing started? Yes No ____Done

10.21. After issuing the PROCESS_PTM command, you will get the following in return:

```
Extracting frames from programmable TM file p100.tmp …
Using MSS ID    12880
```

[This is the MSS build number (here 3250) in hexidecimal. This number may be important if you are using older data requiring an older decom map.]

```
Decommutating telemetry data with format =    2…
4658
% Compiled module: UNIQ.
4658—Decommutation complete
Decommutating telemetry data with format =   11…
4658
4658—Decommutation complete
Decommutating telemetry data with format =    2…
2027
2027—Decommutation complete
Decommutating telemetry data with format =    2…
4658
4658—Decommutation complete
```

[TDP creates a decom map for each format type as it encounters it. After it creates the decom map for format =2, it finds all instances of each telemetry monitor and its value at a particular time and writes those to an array. When it encounters a new format type, it creates a new decom map and does the same process again. The numbers shown before the "Decommutation Complete" line are the number of packets processed in that particular format. The first instance of format 2 in this example contained 4658 packets. Some example format types are:

202 The 32K telemetry format
11 The JK telemetry format (Attitude Control)
201 The 1K Launch telemetry format
205 The 2K telemetry format
215 The science format

After TDP has decommutated the data, it copies it into Sybase, unless /checkonly is specified.]

```
Starting copy...
Batch successfully bulk-copied to SQL Server.
2000000 rows sent to SQL Server.
Batch successfully bulk-copied to SQL Server.
3000000 rows sent to SQL Server.
Batch successfully bulk-copied to SQL Server.
4000000 rows sent to SQL Server.
```

[This may go on for some time…]

```
Batch successfully bulk-copied to SQL Server.
17000000 rows sent to SQL Server.
```

```
17295623 rows copied.
Clock Time (ms.): total = 780360  Avg = 0 (23567.83 rows per sec.)

Starting copy...
Batch successfully bulk-copied to SQL Server.
300000 rows sent to SQL Server.

391488 rows copied.
Clock Time (ms.): total = 13000  Avg = 0 (5440.54 rows per sec.)
```

[Because TDP processes the analog and discrete monitors differently, it copies all the analog monitors into the database first and then goes back and copies in the discrete monitors. Hence the two copying statements.]

10.22. Expect a data summary of the Level 1 import. This should either confirm that the entire import into Level 1 was successful, or, if not, will detail what exactly was imported. After the summary, you will be returned to the IDL prompt. Note that in this example, we are using a different processing report than the previous Level 0 example.

```
Level 1 Data Processing Summary (Version-1.8):

  SpaceCraft Time (SCT) from P100 - aka VTCW
  Cycle                                  -13622
  Min          2001:249:07:35:57.1     530337571
  Max          2001:249:13:05:47.0     530535470

 P100 occurs  FMT  TLM  vld
 -----------  ---  ---  ---
      184621  202  32k   1
       63329  216  32k   1

 P100 occurs  MSSid  MSSid
 -----------  -----  -----
        2480  x3401  13313
    x3400 was the SF_MSS_ID base found
    x3400 was the SF_MSS_ID base used/requested


      247950  Programmable telemetry (APID 100) packets read in

    17295623  Analog   data values extracted
     9495654  Discrete data values extracted
      391488  Thinned Discrete data values to be transferred to the database

    11.83  Minutes spent doing the decommutation
     3.50  Minutes spent BCPing TMdiscrete
     13.20  Minutes spent BCPing TManalog
     0.00  Minutes spent doing post_check   (NO POSTCHECK)
 --------
    28.53  Minutes total run time

  Complete at 2003:119:17:07:21
```

10.23. The SCT refers to the earliest and latest spacecraft times detected in the data. The first line is the cycle number associated with this data. The second line is the minimum time detected in the data given in GMT

and then 10 times the Vehicle Time Clock Word, the third line is the same but is the maximum time detected. After that, a format report is listed for user convenience, then the MSSIDs used to compile this data and to decom it are listed. To interpret the rest of the summary, check the number of Analog data values extracted against the number successfully copied into SQL Server (a.k.a. Sybase) during the bcp reporting to the screen. The numbers should be the same. That should be true of the Thinned discrete data values as well – this number should be the same as the number successfully copied during the last copy statement above (the end of step 10.21).

10.24. TandO.pro. The tando.pro program uses as input the same p100.tmp file as step 10.20 above. It breaks out orbit vectors and timing information from the first 81 bytes of each telemetry format and puts those into a special data table in Sybase. Tando.pro should be run for every data file that was created using MSS 3.3.3 and above. It will not work for any files older than MSS 3.3.3.

10.24.1. To run tando.pro (which may be done before gpb_tdp_L1.pro or afterwards), at the IDL> prompt, type ".run tando" to compile the routines. Then type "TANDO, p100.tmp", assuming your input file is named p100.tmp.

10.24.2. Keyword options include: /CHECKONLY, /DEBUG, MSSID=nnnn, DIGITB= nnnn, /POSITION, /VELOCITY, /OSTAMP, and /TIMES. The /CHECKONLY option is standard – it will run the program without copying the resulting data to Sybase. All the other options are used for debugging and only serve to send more output to the screen. The tando.pro results are stored in a tando.sum file, and they are sent to the screen. Interpretation is self-explanatory. If a summary says that it was unable to bcp data, for example, or the vehicle times appear out of kilter with prior data summaries on this file, you should consider this anomalous and contact data processing. A sample report is given below:

```
TANDO Data Processing Summary (Version-1.8):
  Cycle                                  -13622
  Min          2001:249:07:35:57.1     530337571
  Max          2001:249:13:05:47.0     530535470

    Packets               Processing Descriptions
   ---------              -----------------------
      247950              P100 Programmable Telemetry packets
                0         1k TLM packets
                0         2k TLM packets
           247950         32k TLM packets
                  24793   Occurrences of Frame *0
                   2481   Occurrences of Frame 18
                   2481   Occurrences of Frame 19
                   2481   Occurrences of Frame 20

    Packets               Filtering Descriptions
   ---------              ----------------------
        2480              SF_MSS_ID  = x3401
   ---------
        2480              SF_MSS_ID >= x3330
 =>
                  24783   Occurrences of Frame *0 with proper SF_MSS_ID
                   2480   Occurrences of Frame 18 with proper SF_MSS_ID
                   2480   Occurrences of Frame 19 with proper SF_MSS_ID
                   2480   Occurrences of Frame 20 with proper SF_MSS_ID

Timing Data:
        8250  All four data mnemonics are zeroes (DISCARDED)

Orbit Data: (searching for x50)
        1101  Occurrences of SP_RecvrMode1 = x00
```

```
          1379  Occurrences of SP_RecvrMode1 = x50

    Orbit Data: (more checks)
           486  Data where SP_GPSTimeWeek1 is static (DISCARDED)
             0  Data where SP_GPS_Digit_B  is static (DISCARDED)

    Data Creation:
         16533  GPS TIMING      filesize=      330660
          1217  GPS ORBIT       filesize=       53548

       1.26  Minutes spent decommutating and comprising T and O data
       3.00  Minutes spent bcping TIMING data
       1.00  Minutes spent bcping ORBIT data
       5.26  Minutes total run time
```

TandO Data Processing successful? Yes No ____Done

10.25. To do limit checking on this file, make sure to have typed /POSTCHECK as an option on gpb_tdp_L1, so that it will pass all the appropriate parameters to post_check.pro. Otherwise, running post_check is outside the scope of this document. See code release in S0503 (the VDD) for program code version used and read code to run. To properly interpret post_check.pro's output, you must read the file named post_check.tmp in /apps/supported/lasp/src/tdp. See step 0 below for interpreting post_check.tmp.

Interpreting post_check.tmp: If post_check.pro was run during Level 1 processing, doing a "more post_check.tmp" reveals the following sample information. Header data is broken down into lines containing the date this file was generated, the cycle number, the time range, and the MSSID used to process this data. Columns for alarms are generated with TAB separation. There are no spaces between time values. Types of warnings for a single mnemonic are broken out onto individual lines for each type of violation. The Limit column contains the limit value as defined in the database. MIN/MAX provides the minimum or maximum value that the mnemonic produced. TIMES break down the occurrence of limit violations by time, displaying when they occurred and if they occurred as a sequence or as individual hits. The analog monitors are always listed first in the post_check.tmp report; the discretes are listed second.

```
POST CHECK REPORT:  Generated 2003/056-11:00:00
Cycle: -2801
Time Range: 530363413 - 530503152
MSSID: 3.3.0


LIMIT CHECKING:

 STATUS    SUBSYSTEM MNEMONIC              TYPE   LIMIT  MIN/MAX NUM    TIMES
 ------------------------------------------------------------------------------------
 *ALARM*     ECU    BE_HPM___OnOffA     REDHI  512      516  010    530363415-530363425
 *ALARM*     ECU    BE_HPM___OnOffB     REDHI  512     1024  010    530363415-530363425
 *ALARM*     ECU    BE_VflyVSeOnOfA     REDHI   99    32767  ***    530363413-530503152
 *ALARM*     ECU    BE_VflyVSeOnOfB     REDHI   99    32767  ***    530363413-530503152
 *ALARM*     ECU    CE_UVLampA_I        REDLO  1.0     0.21  327    530363950-530364277
 *ALARM*     ECU    CE_UVLampA_I        REDHI  5.0    99.99  001    530363949
 *ALARM*     ECU    CE_UVLampB_I        REDLO  1.0     0.17  327    530363950-530364277
 *ALARM*     ECU    CE_UVLampB_I        REDHI  5.0    99.99  001    530363949
 *ALARM*     CTU    DC_B_15V_Mon        REDHI 12.1     14.0  400    530363500-530363700,
530363900-530364099,530364121
 *ALARM*     CTU    DC_CTU_A_Load       REDHI 1000     1002  005
      530363600,530363610,530363620,530363630,530363640,530363650
 Warning     EPS    IW_Bat_1            YELLO  2.0    1.001  037    530503100-530503137
 *ALARM*     EPS    IW_Bat_1            REDLO  1.0    0.000  ***    530503138-530503152
 Occurence   EPS    IW_Bat_1            REDHI 20.0    99.00  002    530363413-530363415
```

```
 Warning     EPS    IW_Bat_1           YELLO  2.0   1.001  037    530503100-530503137
 *ALARM*     EPS    IW_Bat_1           REDLO  1.0   0.000  ***    530503138-530503152
 Occurence   EPS    IW_Bat_1           REDHI 20.0   99.99  002    530363413-530363415

STATE CHECKING:

 STATUS     SUBSYSTEM MNEMONIC             TYPE  NUMTIMES
 ------------------------------------------------------------
 *ALARM*     GSS    PY_TaskORunFl_1     STATE ***    530363413-530503152
 *ALARM*     GSS    PY_TaskORunFl_2     STATE ***    530363413-530503152
 *ALARM*     GSS    PY_TaskORunFl_3     STATE ***    530363413-530503152
 *ALARM*     GSS    PY_TaskORunFl_4     STATE ***    530363413-530503152
 *ALARM*     GSS    PY_UV_bias1         STATE 019    530363510-530363529

End of post-pass checking
```

Limit Checking Completed? Yes No ____Done

10.26. To process GPS data, which will be required if your data set includes any format 155, 156, 215, 216 or 217 data (see step 10.21 to see whether you have any of these formats of data), you will need to run the process_gps routine.

Process_gps.pro required? Yes No ____Done

```
IDL> .run process_gps
```

You should expect to see the following modules compiled:

```
% Compiled module: PROCESS_GPS.
% Compiled module: EXTRACT_GPS_TLM.
% Compiled module: ASSEMBLE_GPS_BLK.
% Compiled module: ASSEMBLE_C1_PKT.
% Compiled module: ASSEMBLE_C9_PKT.
% Compiled module: ASSEMBLE_C3_PKT.
% Compiled module: GPS_BLOCKS_TO_DB.
% Compiled module: GPS_PACKETS_TO_DB.
% Compiled module: OUTPUT_STATS.
```

Once returned to the prompt, issue the following command:

```
IDL> PROCESS_GPS, 'p100.tmp'
```

Again, keywords include /CHECKONLY & /DEBUG as possible options. Below is a rather poor example because there was no Format 155, 156, 215, 216 or 217 data available. If one did have appropriate format data, then one would expect to see below reports of many rows copied into the database. Successful GPS processing would be indicated in the summary by a non-zero number of valid packets existing and being copied into Sybase.

```
GPS Data Processing Summary (Version-1.8):

  SpaceCraft Time (SCT) from P100 - aka VTCW
  Cycle                                  -2950
  Min          2001:249:15:02:37.3     530605573
  Max          2001:249:15:04:06.2     530606462

        1810  Programmable telemetry (APID 100) packets read in
```

```
  ---------
          0  APID 100 packets -  1 k format
          0  APID 100 packets -  2 k format
       1810  APID 100 packets - 32 k format
  ---------
          0  APID 100 packets - format=155/156/215/216/217 and frames 1-87
          0  APID 100 packets with valid C1 packets using SP_Packet_ID1
          0  APID 100 packets with valid C9 packets using SP_Packet_ID9

Statistical breakout of SF_MSS_ID:
P100 occurs  MSSid
-----------  -----
          9  x3303

Statistical breakout (of minor frames and C* packets) by SF_Format_ID:
        FMT   Raw Frames   GPS Frames   C* packets
        ---   ----------   ----------   ----------
          2         1810            0            0


Data Creation:
          0  gps_pkt.tmp (GPS C* packet sets)   filesize=           0
          0  gps_blk.tmp (Incomplete blocks)    filesize=           0

    0.01  Minutes spent decommutating and comprising GPS data
    0.00  Minutes spent BCPing GPS packets
    0.00  Minutes spent BCPing GPS blocks
    0.02  Minutes total run time
```

GPS Processing successful? Yes No ____Done

10.27. Troubleshooting. Possible errors indicating trouble with Level 0 or Level 1 processing are as follows:

A. Duplicate Key errors listed in the previous step. If you get these during the bulk copy, that means that Sybase thinks your data is already in the database, or someone using the same cycle number got there first. You should see in the summary that your numbers of values into the database are lower than perhaps expected. Call the database administrator if you see Duplicate Key errors in the previous step.

B. Sybase ran out of LOCKS. If, while BCPing into Sybase at any stage, an “out of LOCKS” message is issued, this means that there was a memory or user contention problem (several people logged in and all wanting to look at data in the same section you attemped to BCP to). It also means that your data did not get into the database! You will need to call data processing and they will need to re-bcp the data in question to the proper database.

C. Seg_analog or Seg_discrete full (or any seg_*** listed as full). This means that the database is out of space. You will need to contact the database administrator as soon as possible to either allocate more space or clean existing data storage space. You will not be able to process any data to the segments indicated in the error message until a Sybase DBA has resolved the problem.

D. The processing may simply appear to stop with and issue no reports. This may mean that the Sybase transaction log is full. Call the database administrator if the processing stops for more than a few minutes – Sybase may be frozen for all users if this is the case.

E. Bad data. It is possible that you may Sybase list a “bcp failed” message for data that is extremely bad. In this case, you should run IPDUsplice on the file(s) and clean them up.

F. An error indicating that the routine "POST_CHECK" is unknown would indicate that the user did not issue the ".run post_check" command before processing the p100.tmp file. To remedy this problem, please check with data processing personnel.

Level 1 processing result: PASS FAIL (circle one) _____ initials

10.28. Exit IDL at the prompt:

```
IDL> exit
```

10.29. Report a successful import to users, or debug an unsuccessful import of data. Call for help if necessary or file a Dlog or DR if required.

Report sent to users: Yes No ____Done

10.30. View Level 1 telemetry using TCAD if desired, or view Level 0 using Sybase if qualified. See P0827 for details.

10.31. Log out of the science server if no further work is to be done.

APPENDIX A: Data Processing Checklist.

PROCESSOR: _____ **DATE:**_____

IN	Task	Contingency/Additional
1	Locate data to process and identify the requestor.	Person(s) called to locate data:
2	Manual select from SCtime for planned cycle number (s) to verify it is empty.	Select count(*) from SCtime where SCT_Cycle = PROPOSED_CYCLE
3	Assign cycle numbers	Number(s):
4	No other instances of TDP are running.	May use alternate dir.
5	Confirm that entire dataset is present if possible.	
6	Notify rest of DB team (via e-mail) that you are handling the files.	List of Files:
7	Transfer files to Science server. DIR:	Use tapes as a last resort.
8	Identify type of data in files; use README file if it exists. Type:	Sim/Flight Data: No special processing Functional Data: No VC0 processing. Framex Files: Manually splice files. Other: Specify
9	Add placeholder entries into SCtime for cycles.	Insert into SCtime values(…)
10	Generate command-file for auto_import.exp	Auto_import double-checks cycles and file names. If not using auto_import, fill out **MANUAL DATA PROCESSING CHECKLIST.**
11	Run 'script' to log all import data.	Script file in same dir as CMD file.
12	Run auto_import.exp	Process VC1&2 files, then VC0 files.
13	Read logs after completion. Were there any error messages? Examples:'Out of Locks', 'TMAnalog Full', '0 records BCP'd', program error, etc. Verify PROCESS_TM_FILE and PROCESS_PTM inputs.	If errors, report to DP Lead. Report delay to customer(s). Reported to: Time:
14	Perform secondary verification by manual select from GPB_L0..Packets	Select count(*) from Packets where SCT_Cycle=<Cycle Number>
15	Update entries in SCtime table to reflect actual time ranges.	
16	E-Mail user group about new data.	Verify that requestor of data is on the distribution list.
17	Deliver **Data Processing Checklist** to DP lead.	Date: Time: