W.W. Hansen Experimental Physics Laboratory
STANFORD UNIVERSITY
STANFORD, CA 94305 – 4085

Gravity Probe B Relativity Mission

LASP Software Verification Report
Version 2.3

S0973 Rev B
3/24/04

Approvals

NAME	SIGNATURE	DATE
Samantha Patterson *Data Processing*	Samantha Patterson	3/25/04
Paul McGown *Chief Bitsmith*	for Paul McGown	3/25/04
Jennifer Spencer *Data Processing Lead*		3/24/04
Ron Sharbaugh *S/W Manager*	Ron Sharbaugh	3/24/04
Marcie Smith *MOC Project Manager*	Marcie Smith	24 Mar 2004
Kelly Burlingham *Software Quality Engineer* for	Dorene Ross	Mar. 25, 2004

Tom Langenstein ITAR Assessment Performed, ITAR Control Req'd? __Yes __No

History

REV	DATE	AUTHOR	COMMENTS
-	3 Dec 2003	Rjs	Lasp-2.1 testing results
A	4 March 2004	Rjs	Lasp-2.2 testing results
B	24 March 2004	Jls	Lasp-2.3 testing results

1 INTRODUCTION

This Software Verification Report is comprised of the "as-run" Software Test Plan Pdocs and additionally the MCRs with "Verification Test Case" sections utilized in the testing for Version 2.3.

1.1 Organization

2 APPLICABLE DOCUMENTS

Document No.	Document	ALIAS
S0503	Version Description Document for TDP/TCAD	
S0603	Telemetry Data Processing Level 1 Verification S0603 Telemetry Data Processing (TDP) in the non-real time system	
P0949	Operational Procedure for Regression Testing of TDP	
P0950	Operational Procedure for Regression Testing of TCAD	
P0981	Automated Import Scripts Software Test Document	
P1078	Operational Procedure for Baseline & Regression Testing of Auto Import	

3 TEST CASES

The following STPs and MCRs comprised testing for this release:

3.1 First Attachment: P0949 rev D (TDP) "as-run"

OVERALL: Recommend Release.

FAILURES:

8.1 THROUGHPUT SPEED

Failed BCP, spec is 18,500 rows per second and testing achieved 13,500 rows per second. A Dlog was opened to investigate the speed problem. Because decommutation times were within spec, and in fact better than the prior release of software, and it was only the Sybase bcp utility causing a problem, lasp software is not at fault for the speed concern. Recommend release.

NOTES AND REDLINE INFO:

1. MSS Database Import – eliminated
2. Average Level 1 TM Analog Data: Test was re-written.

3.2 Second Attachment: P0950 rev E (TCAD) "as-run"

FAILURES:

MCR #312 was tested but failed. The code written for MCR #312 works if a workaround is put into place: create a file named "temp" with read/write access for the group "users" in the directory /apps/supported/lasp/tcad/analyze/. Note added to "Installation Instructions" section of the VDD and workaround in place.

NOTES AND REDLINE INFO:

1. Testing generated one MCR: #314

3.3 Third Attachment: P0981 rev A (cron) "as-run"

NOTES AND REDLINE INFO:

1. Removed moc_auto.cron testing

3.4 Fourth Attachment: P1078 rev - (auto_import) "as-run" two copies.

NOTES AND REDLINE INFO:

1. Two copies of the as-run are needed because one was run on science and the other on moc-server.

3.5 Attached MCRs:

210
229
301
306
307
308
309
310
312
313
314

3.6 LASP 2.3 Code Walk-through Notes

4 SUMMARY

OK to release.

As-Run

P0949 Rev D
February 26, 2004

Attach To S093-B

W.W. Hansen Experimental Physics Laboratory
STANFORD UNIVERSITY
STANFORD, CA 94305 – 4085

Gravity Probe B Relativity Mission

Operational Procedure for Baseline and Regression Testing of TDP

P0949 Rev D
2/26/03

Approvals

NAME	SIGNATURE	DATE
Jennifer Spencer *Data Processing Lead*		
Paul McGown *Chief Bitsmith*		
Ron Sharbaugh *S/W Manager*		
Marcie Smith *MOC Project Manager*	ECO 1479	
Kelly Burlingham *Software Quality Assurance*		2/27/04

Required Signatures prior to Execution

NAME	SIGNATURE	DATE
NAME: *Test Engineer*		22 Mar 2004
Kelly Burlingham *Software Quality Assurance*		Mar 22, 04

Tom Langenstein ITAR Assessment Performed, ITAR Control Req'd?

___ Yes ___ No

Table of Contents:

1 REVISION HISTORY

REV	DATE	AUTHOR	COMMENTS
-	16 Oct 2002	JLM	initial version
A	30 May 2003	JLS	Added baseline testing to this document, changed it from a regression-test-only document to a full test plan.
B	9 Sept 2003	JLS	Section 10.1 updated (TDP5) to reflect new data tables.
C	11 Nov 2003	JLS	Re-arranged order of test sections to reflect actual order of tests. No content changes.
D	26 Feb 2004	JLS	Updated section 10.5 in response to code changes.

2 SCOPE

This Test Plan Document details the TDP software package and how to both baseline and regression test it.

3 OPERATIONAL PERSONNEL

Jennifer Spencer
Paul McGown
Qualified QA Rep: Kelly Burlingham

4 REQUIREMENTS & CONSTRAINTS

4.1 Hardware and Software Requirements

Operations are performed on the Sun server machine known as "science". If spacecraft or simulated data is required, it will be from either a data tape delivered from the Lockheed Martin Integrated Test Facility (ITF) or an FTP'd data file from the ITF or a POD.

4.2 Configuration Requirements

The operator must be logged into the server "science" as the user "tdp" and must know the MSS Version of any data they are using for test purposes. The user should reserve a directory for test data and save this data for QA.
Directory: /apps/supported/lasp-2.3/data

4.3 Constraints

Constraint	Risk
Time required for regression testing.	At least three hours is required for regression testing and considerably more for baseline testing. At this time, data processing is not directly impacted but support for the department may be unavailable due to personnel working tests.
Quality of VC1and2 or VC0 binary data	Exceptionally good quality data could perhaps not fully test TDP's ability to filter out bad packets. Recommend using data known to be poor quality over exceptionally good quality based on risk listed above. Bad data includes: gaps, bad packet lengths, negative time ranges, data with values of floating point infinity and negative zero, for example.

5 REFERENCE DOCUMENTS

Document	Document No.	ALIAS.
Data Management Plan	S0331	
Stanford Post-Processing Operations for Science Mission Data	S0401	
MOC Configuration Control, IONET LAN	S0476	
LM VDD: Real-time Ground Software	LMMS/P479910O	
IONET_CONF		
Telemetry Data Processing "Level 0" Verification	S0572	
Telemetry Data Processing "Level 1" Verification	S0603	
TDP/TCAD Software Release (Design Document)	S0613	
MOC Requirements	MO-02	
Processing Data from High-resolution to an Averaged Data Set	P0905	
Event Display in Non-Real-Time System	P0832	

DBRO Data Retrieval and Display	S0638	
MOC VDD for EVENTget-pl	S0515	

6 QUALITY ASSURANCE PROVISIONS

Quality Assurance must be given 24 hour notification before this test is run; presence is at their discretion.

QA Notified Date & Time: 3/19/04 By: 10:00 AM Ron Shaubough QA Initials: SU GP-B INSP 31

7 TEST ENVIRONMENT

This software is tested on the science server at SU under the following configuration:

Software Configurations	Version Number (fill in)
IDL	5.4
Sybase	12.5.03
Solaris	5.8
~~MSS~~	N/A
~~PERL~~	N/A

8 TEST CASES FOR TDP GENERAL ROUTINES

Test case components are listed in the table below, along with whether or not they need to be run and if so, initials of who has run the tests. Enter either B for baseline testing or R for regression testing under type of verification required. The objective for regression testing each component is always to get a zero difference when comparing the old revision's output files (from the old, production software directory) to those from the new revision (in the development directory awaiting release). Regression tests are considered passed if no differences are found in running binary or ASCII differences between old files and new files. If there are differences, or if code module is completely new to this release, baseline testing must be done. Any data output differences (as opposed to simple report formatting differences) must be explained by updated design documentation. Pass/Fail criteria for each test are set forth below, and supercede any general instructions for specific test cases. Test set up for final verification of software requires that no further modifications will be made to code and that no code is checked out of RCS for editing. Execution procedures and result descriptions follow each case below.

If the code in any of the following files has changed since the previous software release, the tests in the corresponding section number must be run and checked off for verification.

LASP version being tested: 2.3

VERIFICATION REQUIRED • Baseline (B) • Regression (R)	FILE (under lasp-x.x/src)	UNIX date & filesize on program directory	TEST NAME	TEST SECTION	VERIFIED BY *(initials)*	DA
	Throughput/Speed (no filename - use src/tdp directory) /apps/supported/lasp/tdp		TDP1	Section 8.1		
	MSS Import src/db/load_x_x_x directory		TDP10	Section 8.2		

8.1 TDP1: Throughput/Speed

❑ (B) ❑ (R) initials

Test Case Verification Number: TDP1

INTRODUCTION
This test case is used for doing timing measurements of TDP processing.

APPROACH
Simply time how long process per P0826 or P0905 (depending on whether this is testing raw file → final data processing or from L0 → final data processing) takes.

FEATURES TO BE TESTED (CHECK ONE)

- ❑ How long "standard" importation of VC files (spacecraft binary files) from a GN pass takes (requirement MOC-18) to process through Level 1.
- ❑ How long retrieval from L0 to L1 takes (requirement MOC-19).

New Test
See Attach
8.1-1, 2, 3

FEATURES NOT TO BE TESTED

- Accuracy of processing
- File splicing in the event of clock resets
- Data retrieval speeds
- Extra features such as limit checking, GPS processing or Snapshot building, which are only done when those data types are found in the data file.
- Does not include any time required for SAFS data transfer through IOnet.

TESTS

- ❑ Log the wall-clock time or UNIX "date" time at start and end of running P0826 or read the on-screen data summary which contains the time for processing.
- ❑ Log the wall-clock time or UNIX "date" time at start and end of running P0905 or read the on-screen data summary which contains the time for processing.

PASS/FAIL
For MOC-18/19, the duration times must be less than spec. bcp times low

RESULT: PASS FAIL (circle one) initials

8.2 TDP10: MSS Database Import

❑ (B) ❑ (R) _____ initials

Test Case Verification Number: TDP10

INTRODUCTION
To ensure that the latest MSS database from Lockheed is imported properly onto the "Science" server at Stanford's Gravity Probe B Mission Operations Center. This database is used by TDP for decommutating the telemetry from the raw spacecraft files.

APPROACH
Visual inspection of output files to be imported to Sybase.

FEATURES TO BE TESTED

- Check that files to be imported to Level 1 metadata are as expected from the populate scripts

FEATURES NOT TO BE TESTED
- Integrity of MSS database

TESTS

❑ BASELINE
Run the MSS import process as described in P0908 (MSS to TDP/TCAD Database Population Process) through step 11.19. During this step, the shell script will ask the user to hit any key to continue. At this point, halt the process by typing Ctrl+C and inspect the output from the populate shell script. When inspection is complete, inspect the data sets that would be produced by the populate_tm*.sql scripts by running the procedure on Sybase command line to just before the "execute" state. Inspect results.

PASS/FAIL
Data sets to be imported pass visual inspection
The populate shell script created proper scripts and directories.

N/A for this test - done as part of P0926 3/22/04 [illegible]

RESULT: PASS FAIL (circle one) _____ initials

SU GP-B INSP 31

REGRESSION → SEE RESULTS FOR TDP5
If data passes regression or baseline test for TDP5 (Level 1 processing), then the MSS database import was necessarily correct. If the data passes L1 verification then the MSS import was done properly and the MSS testing is working. Regression testing of this routine is not applicable.

While not required to show proof of success, if desired, a "regression" test of each MSS import could be performed by following the steps in P0908 and by then comparing the log files generated by the import process against the LM VDD deliverable document.

9 TEST CASES FOR LEVEL 0

Test case components are listed in the table below, along with whether or not they need to be run and if so, initials of who has run the tests. Enter either B for baseline testing or R for regression testing under type of verification required. The objective for regression testing each component is always to get a zero difference when comparing the old revision's output files (from the old, production software directory) to those from the new revision (in the development directory awaiting release). Regression tests are considered passed if no differences are found in running binary or ASCII differences between old files and new files. If there are differences, or if code module is completely new to this release, baseline testing must be done. Any data output differences (as opposed to simple report formatting differences) must be explained by updated design documentation. Pass/Fail criteria for each test are set forth below, and supercede any general instructions for specific test cases. Test set up for final verification of software requires that no further modifications will be made to code and that no code is checked out of RCS for editing. Execution procedures and result descriptions follow each case below.

If the code in any of the following files has changed since the previous software release, the tests in the corresponding section number must be run and checked off for verification.

LASP version being tested: 2.3

VERIFICATION REQUIRED • Baseline (B) • Regression (R)	FILE (under src/tdp)	UNIX date & filesize on	TEST NAME	TEST SECTION	VERIFIED BY *(initials)*	DATE

		program file				
	tdp start script /lasp/scripts/tdp		TDP13	Section 9.1		
	IPDUsplice.pro		TDP11	Section 9.2		
	gpb_tdp_L0.pro		TDP2	Section 9.3		
	process_tcor.pro		TDP3	Section 9.4		
	ssbuild.pro		TDP4	Section 9.5		
	l02l1		TDP6	Section 9.6		
	L0_to_L1_display.pro		TDP6	Section 9.6		
	L0_to_L1.pro		TDP6	Section 9.6		

See Below

SC GP-B INSP 31

9.1 TDP13: TDP start script

☒ (B) (regression test not applicable) JS for PM initials

Test Case Verification Number: TDP13

☐ BASELINE

INTRODUCTION
This test verifies whether the TDP startup script works.

FEATURES TO BE TESTED
– Properly start TDP and use the proper release version

FEATURES NOT TO BE TESTED
– None

TEST
The following requires a set of VC files that contain all packet types (this can be just one file). Each file is a different test:

i. Run "tdp" at the command line of any science terminal.
ii. This should start /apps/supported/scripts/tdp, which in turn is symbolically linked to /apps/supported/lasp/scripts/tdp.
iii. Inspect the output from the command and read the version number

PASS/FAIL
TDP started using the proper code release version. lasp-2.3/scripts

RESULT: PASS FAIL (circle one) JS for PM initials

9.2 TDP11: IPDU frame parsing on raw files

☐ (B) ☒ (R) NJ initials

Test Case Verification Number: TDP11

INTRODUCTION

TDP processes one binary spacecraft file at a time (the "raw" data). One file represents one pass or "dump" from the spacecraft taken during a pass and shipped to the Mission Operations Center from SAFS. Each

binary file is composed of blocks of data known as "IPDUs" (Internet Protocol Data Units). Each IPDU is made up of a satellite transfer frame (of a known, constant size) and a header (the IPDU header - also a constant size and format).

IPDU headers are assigned to each data packet by the Front-End Processor (FEP) at the receiving ground station. These IPDU headers essentially describe what time the data arrived on the ground and some Reed-Solomon processing status information.

RJS

☒ BASELINE

SU GP-B INSP 31

APPROACH
IPDUsplice.pro is a difficult program to test. It involves a great deal of inspection against original specifications of data assembly found in SCSE-16, section 9.

FEATURES TO BE TESTED
- Correct packets are parceled out and labeled as correct IPDUs
- Bad data is removed
- Good data is not thrown out with the bad
- SSR data wraps taken into account
- SV clock resets to zero taken into account
- SV clock "advances" or "backups" of more than one day are taken into account

FEATURES NOT TO BE TESTED
- None

SU GP-B INSP 31

TESTS

i. Using the specification for an IPDU header and the definition of a valid spacecraft data packet from SCSE-16, section 9, one must inspect all data rejected as "bad" by IPDUsplice and confirm that it is indeed bad data.
ii. Then, run a good, completely clean file through IPDUsplice and the result should be only one splice – the original file contents. Do a binary difference between the original file and the splice; there should be no differences.
iii. For confidence, cycle through the data labeled as good by IPDUsplice. Inspect it visually using an octal dump (od –tx1).

PASS/FAIL
IPDUsplice files are successfully inspected.

RESULT: PASS FAIL (circle one) _____ initials

☒ REGRESSION

TEST
IPDUsplice.pro outputs should be the same between lasp-x.x versions. If the program has been altered, run the released version and the new new version of IPDUsplice on a VC*.bin file in two test directories. Do a binary comparison of the binary output files of this tool (the text files may differ, but they are summaries of information only – essential content should be the same and should be inspected) between the old version of the code and the new version (binary diff as detailed in TDP2 above). Any differences should be explained by documentation and a baseline test should be considered.

/apps/supported/lasp-2.3/data/DIF

PASS/FAIL
Each comparison must yield no difference between released and new code output files.

RESULT: PASS FAIL (circle one) _____ initials

No screen output, so pass

9.3 TDP2: Packet Parsing/Decomposition

☐ (B) ☒ (R)

RB initials

Test Case Verification Number: TDP2

INTRODUCTION
This test verifies the accuracy of packet decomposition.

☐ BASELINE
Specifically, it checks that the content of the output file of SU's TDP (to be bcp'd into Sybase as the Level 0 product) matches that analyzed by LM's EDR (which is a verified deliverable CSCI).

APPROACH
A verification program, gpb_tdp_L0.pro, is developed to exactly duplicate both TDP and EDR processing – i.e. one inputs a VC file, and this third-party program outputs either a file in the format of an SU TDP binary file or in the format of an LM EDR binary file. Note that EDR processing rejects NO packets, while gpb_tpb_L0.pro discards any packet that it can detect as bad (such as date stamps over 2006, improper packet ID or length, et cetera).
The same input VC file(s) is processed four times to verify each APID type:

i. Once by SU TDP.
ii. Once by LM EDR.
iii. Once by the verification program to output SU TDP files.
iv. Once by the verification program to output LM EDR files.

For each VC input file, the output TDP format files are compared, and then the output EDR format files are compared.

FEATURES TO BE TESTED
- Properly parse VC (IPDU) file into spacecraft packets.
- Properly assessing and categorizing packet types.
- Using proper packet lengths (per packet type).
- Assigning proper packet timestamps for each packet (only packets 100 and 301 receive timestamps from the vehicle; all others must use the p100 times within the same transfer frame).

FEATURES NOT TO BE TESTED
- Rejection of bad data is not directly tested. It is tested through regression testing (see P0949) instead. New, better filters and the final data processing summary show direct, verifiably bad pieces as "leftovers" during a before & after file comparison between code versions.

TEST
The following requires a set of VC files that contain all packet types (this can be just one file). Each file is a different test:

i. Run the programs to create the 4 different output files.
ii. Compare the TDP output file (i above) with the test program "tdp format" output file (iii above).
iii. Compare the EDR output file (ii above) with the test program "EDR format" output file (iv above).

PASS/FAIL
Each comparison must be identical.

RESULT: PASS FAIL (circle one) _____ initials

☒ REGRESSION
Checks only that output files from gpb_tdp_L0.pro are identical from one version of lasp-x.x to the next.

TEST
The following requires a set of VC files that contain all packet types (this can be just one file).

i. Run gpb_tdp_L0.pro in both the released and new lasp-x.x code directory with the keyword /CHECKONLY in the command line. This will generate output files p100.tmp, p200.tmp, p300.tmp, events.tmp and snaptemp.tmp.
ii. Do a UNIX binary "diff" for each type of file between released and new, e.g. "diff /apps/supported/lasp-1.7/src/tdp/p100.tmp /apps/supported/lasp-1.8/src/tdp/p100.tmp". Do this for each .tmp file in turn.
iii. If the command above returns to the prompt with no output, that means there is no difference between the two files.

~~[illegible]~~ have all packet types

PASS/FAIL
Each comparison must yield no difference between released and new code output *.tmp files.

RESULT: PASS FAIL (circle one) _____ initials

9.4 TDP3: IPDU Time Correlation

☐ (B) ☒ (R) RJ initials
Test Case Verification Number: TDP3

INTRODUCTION
This test verifies that process_tcor.pro properly breaks out IPDU header times from VC0 files and correctly performs a least-squares fit analysis.

☐ BASELINE

APPROACH
An independent verification program is developed, TERLtcor.pl. TERLtcor.pl emulates TDP's process_tcor (but uses a slightly different least squares algorithm). Process_tcor.pro strips IPDU headers out and stores their Earth Receipt Time (ERT) records. It uses these IPDU header ERT times for analysis against spacecraft clock times for 300 p100 time stamps and produces a least squares fit along with the raw data (for 300 points). The outputs from both programs are compared (using the same input data).

FEATURES TO BE TESTED
-- Properly parse VC0 file IPDU headers and create file(s) for import to database.
-- Properly comparing IPDU header ERT times with vehicle clock (VTCW) times for each transfer frame.
-- Correctly does least-squares analysis between ERT and VTCW times.
-- Ignores VC1and2 data, ignores any 1K or 2K format data
-- Performs same sanity checks as gpb_tdp_L0.pro for validity of data packet (such as date stamps over 2006, improper packet ID or length, et cetera)

FEATURES NOT TO BE TESTED
- Presupposes that any clock resets or time forward jumps are not in the input file (IPDUsplice.pro has been cleanly run).
- None. Please note that routine does not account for other factors involved in time correlation, such as time delays caused by: transmission from on board computer to spacecraft antenna, spacecraft data transmission from the craft to the ground, data transmission from the antenna to the front-end processor, et cetera.

TESTS
The following requires a VC0 file:

i. Process the VC0 file with process_tcor.pro.
ii. Process the VC0 file with TERLtcor.pl.

iii. Compare the TDP output file (i) with the test program output file (ii).

SU GP-B INSP 31

PASS/FAIL
Passes if all entries for both files match. All entries between tcor_raw input files to data tables must match exactly. The least-squares c0 and c1 coefficient results should match to within 0.001.

RESULT: PASS FAIL (circle one) _____ initials

☒ REGRESSION
Process_tcor.pro is the code that processes IPDU header data and generates a least-squares fit of vehicle time against ground receipt time. To verify that its output is the same between versions, follow the same binary UNIX diff comparison steps listed above in TDP2, but run the process_tcor routine using the keywords /CHECKONLY, /FIT, and /RAW to get the proper output files. The files to compare are tcor_raw.tmp and tcor_fit.tmp.

wc0_4022 31632, bin

PASS/FAIL
Each comparison must yield no difference between released and new code output *.tmp files.

RESULT: PASS FAIL (circle one) _____ initials

9.5 TDP4: Snapshot Building

☐ (B) ☒ (R) _____ initials
Test Case Verification Number: TDP4

☐ BASELINE

INTRODUCTION
This test verifies the accuracy of snapshot packet assembly into full snapshots. The spacecraft sends down snapshots in small packets, and depending on the type, the number of packets making up a full snapshots varies. The CSCI gpb_tdp_L0.pro processes these packets into a holding area in Sybase called GPB_L0..Snaptemp. The file that makes up Snaptemp's contents is called snaps.tmp.
Specifically, this verification checks that the content of the snaps.tmp file matches the analysis done by LM's EDR (which is a verified deliverable CSCI). Then, to verify assembly, it checks that the snapshots.tmp output file of SU's TDP ssbuild routine (to be bcp'd into Sybase as GPB_L0..Snapshots end product) matches a third party program's full analysis. Because LM's EDR does not do full packet assembly on snapshot packets, full verification is only possible using third party analysis.

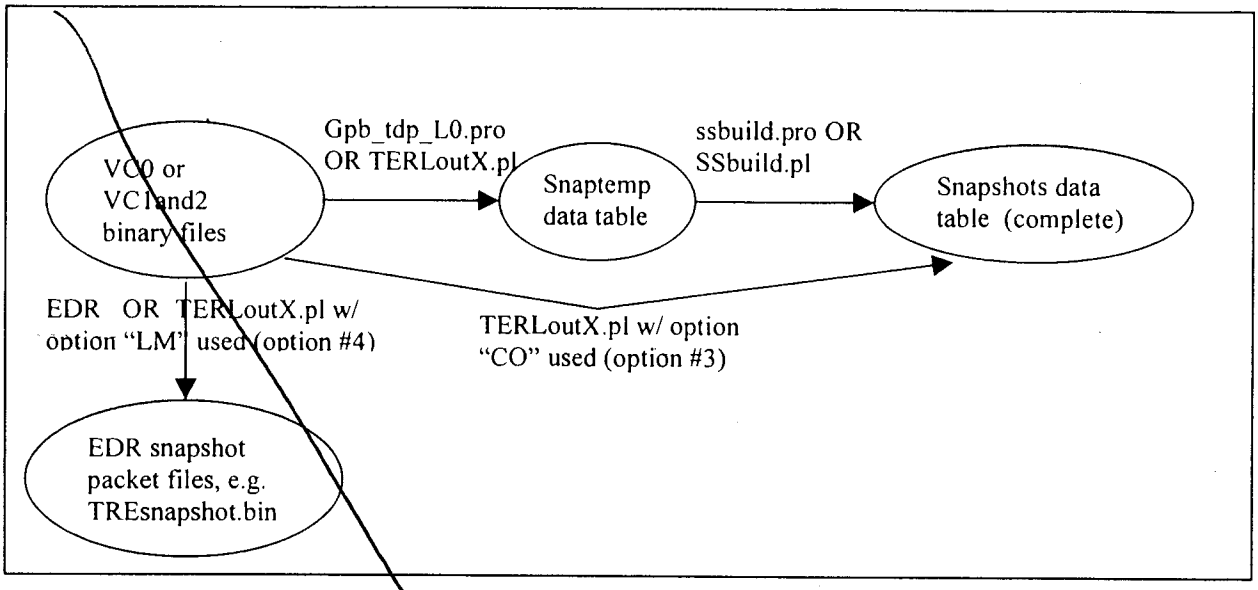
APPROACH
Four verification programs are developed.

SU GP-B INSP 31

- Use a VC1and2 file with as many types of snapshots taken as possible. Verification may require more than one file.
- Process this file through five programs: gpb_tdp_L0.pro, LM EDRS, ssbuild.pro, ssbuild.pl and TERLoutX.pl. It is possible that running code of LM EDRS will not be required after a number of runs - it does not fully process snapshots and prior testing has verified that the third party program mimics it exactly.
- Compare results and look for zero differences for analytic success.

The verification approach is more easily understood with a diagram.

Figure 1: Snapshot Processing Options for Raw Spacecraft Binary Files



i. The TERLoutX.pl program duplicates both TDP (gpb_tdp_L0.pro) and EDR processing – i.e. one inputs a VC file, and this program outputs either a SU TDP binary file called snaps.tmp or a LM EDR binary files. Additionally, this test program can output "ssbuild.pro" format files.
ii. The ssbuild.pl program duplicates the TDP (ssbuild.pro) processing.
iii. Snapshot "exploder" is used to sort the output the 1st verification program so that it matches the output of ssbuild.pro.

Input VC file is processed five times:
i. Once by SU TDP, completing ssbuild.pro processing to final GPB_L0..Snapshots table.
ii. Once by LM EDR, if required.
iii. Once by TERLoutX.pl (to output snaps.tmp) and then by ssbuild.pl (using snaps.tmp as input) to output SU TDP files (snapshots.tmp).
iv. Once by TERLoutX.pl to output the LM EDR files.
v. Once by TERLoutX.pl to output the final "snapshots.tmp" file.

Compares are done between (i) and (iii), then between (ii) and (iv), and then between (iii) and (v).

FEATURES TO BE TESTED
- Properly assemble Snaptemp contents into complete snapshots.
- Recognize but not pass on partial snapshots to Snapshots table.
- Report on number of snapshots comprised and number of partial snapshots, and report on different types of snapshots

FEATURES NOT TO BE TESTED
- Any housekeeping tasks necessary
- Speed of assembly

TESTS
The following requires one, possibly many, VC1and2 file(s) so that all snapshot types being tested are in the sample data:
i. Process the VC1and2 file with SU TDP, completing ssbuild.pro processing to final GPB_L0..Snapshots table.
ii. Process VC1and2 file using LM EDR, if required.
iii. Process VC1and2 file by TERLoutX.pl (to output snaps.tmp) and then by ssbuild.pl (using snaps.tmp as input) to output SU TDP files (snapshots.tmp).

SU GP-B INSP 31

iv. Process VC1and2 file by TERLoutX.pl to output the LM EDR files (if required).
v. Process again by TERLoutX.pl to output the final "snapshots.tmp" file.

OVERALL PASS FAIL

Comparisons are done between (i) and (iii), then between (ii) and (iv), and then between (iii) and (v). All differences should be zero.

EIE

RESULT: PASS/FAIL (circle one) _____ initials

☒ REGRESSION

SU GP-B INSP 31

Ssbuild is the code that processes snapshots from the Level 0 Snaptemp table to the Level 0 Snapshots table. To verify its output is the same between versions, follow the same steps as outlined above in TDP2 and compare snapshots.tmp from both versions. Any differences should be explained by code changes and specific analysis, and baseline test should be considered.

PASS/FAIL

Each comparison must yield no difference between released and new code output snapshots.tmp file.

RESULT: PASS FAIL (circle one) _____ initials

9.6 TDP6: Level 0 to Level 1 reprocessing

☐ (B) ☒ (R) _____ initials

Test Case Verification Number: TDP6

INTRODUCTION

The L0 to L1 Reprocessing tool (named l02l1) is designed to access the Level 0 Sybase database table named Packets and retrieve all APID 100 packets within the time range specified by the user for reprocessing into the Level 1 Sybase database. L02l1 is a stand-alone tool. L02l1 is a unix shell script which runs L0_to_L1.pro written in IDL with calls to the C functions in the LASP db_routines program. The user interface screen for this tool is L0_to_L1_display.pro.

☐ BASELINE

APPROACH

Run the l02l1 script for a specified time period and cycle number and compare the output file (p100.tmp) against the data in the database (GPB_L0..Packets) for the same cycle and time period. If an original p100.tmp file is available for comparison, it is preferable to use this and its corresponding cycle number and time period.

SU GP-B INSP 31

FEATURES TO BE TESTED

- Whether the l02l1 GUI functions properly
- Whether or not the correct packets are selected from the GPB_L0 database

FEATURES NOT TO BE TESTED

- Any attempted duplication of data (e.g. if that set of packets has already been reprocessed)

TESTS

i. Choose a time period and cycle number.
ii. Run l02l1 to retrieve all APID 100 packets for that time period and cycle number.
iii. Extract the APID 100 data from the database for that same time period and cycle number into a file with the same structure as p100.tmp, or use the original p100.tmp file if available.
iv. Use the UNIX "diff" tool to compare the two files. There should be no differences.

v. Check the file size for each file. These should be the same.
vi. Perform a visual inspection of the two data files using the UNIX "od" octal dump tool. There should be no differences.

SU GP-B INSP 31

PASS/FAIL
After comparing the p100.tmp file with the corresponding data in the GPB_L0..Packets table (or the original p100.tmp file), there should be no differences.

RESULT: PASS FAIL (circle one) _____ initials

☒ REGRESSION

TEST
Level 0 to Level 1 reprocessing suite of tools includes the l02l1 startup script, the L0_to_L1.pro processing routine and the L0_to_L1_display.pro user interface screen. If any of these three files are changed, do a binary comparison of the output file of this tool, packet100.tmp, between the old version of the code and the new version (binary diff as detailed in TDP2 above). Any differences should be explained by documentation and a baseline test should be considered.

PASS/FAIL
Each comparison must yield no difference between released and new code output packet100.tmp file.

RESULT: PASS FAIL (circle one) _____ initials

10 TEST CASES FOR LEVEL 1

Test case components are listed in the table below, along with whether or not they need to be run and if so, initials of who has run the tests. The objective for testing each component is always to get a zero difference when comparing the old revision's output files to those from the new revision. Tests are considered passed if no differences are found in running binary or ASCII differences between old files and new files; if there are differences they must be explained by accompanying documentation such as a revised or new Sdoc. Pass/Fail criteria for each test are set forth below, and supercede any general instructions for specific test cases. Test set up requires that no further modifications will be made to code and that no code is checked out of RCS for editing. Execution procedures and result descriptions follow each case below.

If the date on any of the following files has changed since the previous software release, the tests in the corresponding section number must be run and checked off for verification.

LASP version being tested: Insp - 2.3

VERIFICATION REQUIRED • Baseline (B) • Regression (R)	FILE (under src/tdp)	UNIX date & filesize on program file	TEST NAME	TEST SECTION	VERIFIED BY (initials)	DATE
	gpb_tdp_L1.pro		TDP5	Section 10.1		
	Post_check.pro		TDP9	Section 10.2		
	Process_gps.pro		TDP7	Section 10.3		
	TandO.pro		TDP8	Section 10.4		
	Average_analog.pro		TDP12	Section 10.5		

SU GP-B INSP 31

See Appendix 1.0

10.1 TDP5: Level 1 Processing

☐ (B) ☒ (R)

RS initials

Test Case Verification Number: TDP5

☐ BASELINE

INTRODUCTION

The baseline test proves by analysis that TDP (the non-real-time Telemetry Data Processing software) is correctly processing data from the binary contents of the Level 0 database, which originate from raw, Gravity Probe B spacecraft binary files (please reference section 9 of this document for verification of Level 0 processing). The analysis includes the correct correlation of spacecraft vehicle time to each mnemonic.

SU GP-B INSP 31

APPROACH

One outside verification program, TERL_L1, is developed to emulate the end results expected by the Level 1 database.

Use a VC1and2 file, a VC0 file or a framex file (any spacecraft file is acceptable) and process through gpb_tdp_L0.pro and then gpb_tdp_L1.pro. Process this same file using TERL_L1 and compare the outputs of both processes. These outputs should be identical.

FEATURES TO BE TESTED

- Accuracy of Level 1 data processing on APID 100 Packets
- Spacecraft time stamping for each mnemonic
- Unpacking of parent to child mnemonic data

- Decommutation
- MSS version-dependent decommutation

FEATURES NOT TO BE TESTED

- Speed of processing
- Any packet processing that is not APID 100
- Formation of derived monitors

TESTS

The test requires a clean (sent through IPDUsplice.pro) spacecraft file in the form of a VC1and2 file, a VC0 file or a framex file from a test bed system.

i. The spacecraft file must be processed by TDP (using gpb_tdp_L0.pro) to create Level 0 data, thus creating an output file named p100.tmp, which is full of APID 100 packet data.
ii. This same file, p100.tmp, must be run through both systems – TERL_L1 and TDP – generating independent results in two different parts: analog and discrete.
iii. Perform the following steps are performed on the analog and discrete data generated by both TERL_L1 and TDP:
   a. Select the first 50,000 frames from each file for comparison
   b. Pipe all outputs from TERL_L1 and TDP to ascii files.
   c. Perform the UNIX "sdiff –s" function on the TERL_L1 analog data vs. the TDP analog data, and again on the TERL_L1 discrete data vs. the TDP discrete data.
iv. Perform the steps listed above for the /scionly or /engonly data (1K and/or 2K data is considered "/engonly", 32K data is considered "/scionly). In TDP, the 1K/2K data will go to two different ascii files than the two files created for 32K data, so two repetitions may be required if the source data has mixed telemetry rates.

SU GP-B
MSP 9

PASS/FAIL

After comparing TDP discrete to TERL_L1 discrete, and TDP analog to TERL_L1 analog, the differences should be zero with the following exceptions:

- TERL_L1 processes its zeroes as "0" and TDP processes zeros as "-0" and "0". TERL_L1 processes all "0" and "-0" as "0" while TDP keeps "-0" and "0" as is from the spacecraft.

RESULT: PASS FAIL (circle one) _____ initials

REGRESSION

TEST

Choose a p100.tmp or packets100.tmp data file (generated by gpb_tdp_L0.pro or the l02l1 tool – this file can be from the same testing run used in section 9 above) preferably containing as many different formats as possible to enable a larger comparison.

i. Run gpb_tdp_L1 with /checkonly keyword on the p*.tmp file file. See P0826 for further instructions on processing Level 1 data if necessary. Note that the p100 file (or packets100 file) should be processed using the same MSSID).
ii. Do the UNIX command "diff" on each binary file (listed below) in the released version of lasp against test version of lasp. (e.g. 'diff filenameA filenameB' where filenameA and filenameB are the output files from the old and new versions of gpb_tdp_L1.pro, respectively).
   a. Testing analog data types: compare tmanalog.tmp old vs new, compare snanalog.tmp old vs new (if 1K/2K data is in source file).
   b. Testing discrete data types: compare tmdiscrete.tmp old vs new, compare sndiscrete.tmp old vs new (if 1K/2K data is in source file).
   c. Testing general functionality of input to limit checking routine: compare tmminmax.tmp and tmbadstate.tmp old vs. new.

Any differences should be explained by changes made to code. Documentation must be produced to explain any differences made by code (such as a revision of S0613). If there were no gpb_tdp_L1, gpb_db.pro or dbroutines.so code changes, there should be no differences.

PASS/FAIL
Each comparison must yield no difference between released and new code output files.

RESULT: PASS FAIL (circle one) ____ initials

timing differences as expected.

10.2 TDP9: Post Check

☐ (B) ☒ (R) ____ initials

Test Case Verification Number: TDP9

INTRODUCTION
This test case is used to verify the generation of limit checking and state checking summaries, which are generated after the Level 1 data processing.

☐ BASELINE

APPROACH
Comparison of generated file against database selections and TCAD plots using post_check.pro.

FEATURES TO BE TESTED
- Reporting analog data samples which exceed the limit settings in the MSS database.
- Reporting discrete values which are outside the acceptable states for this variable.
- Calculation speed.

FEATURES NOT TO BE TESTED
- Validation of limit values.
- Choice of MSS database for limit generation.
- Interpretation of state and limit report by users.
- Import of state and limit report into TQSM application.

TESTS
- Record the reported time of post_check start, and the reported time of post_check completion.

  START TIME:_____ STOP TIME:_____

  SU GP-B INSP 31

- After running a full level 1 import (run the gpb_tdp_L1.pro program with the /checkonly keyword if the data was previously BCP'd to the Level 1 databases) verify that the file 'postcheck.tmp' was generated during the run of this import.
  - Open the postcheck.tmp file in a text editor or "more" the file. Identify the two section headers in the file, 'Limit Checking' and 'State Checking'.
  - Choose two or more mnemonics from different subsystems that are reported as out of limits. Using the earliest violation time and the latest violation time, select a time range for this cycle in TCAD and plot the selected mnemonics with 'Display Red Limits' selected in the 'Plot Options' screen. The plotted mnemonic should be seen to cross the limit line as reported in the postcheck.tmp file.
- For secondary validation of limit checking:
  - Look up the TMID of each mnemonic, using either the TM INFO screen in TCAD or a manual selection from the GPB_L1.. TMnames data table.

- Select * from TMlimits where MSSID=(select max(MSSID) from GPB_L1..TMlimits) AND TMID=#your_TMID#.
- Select * from TManalog where SCT_Cycle=#your_CYCLE# and MSSID=#your_MSSID# and (Value > #Yellow_Low_Value# OR Value < #Yellow_High_Value#)
- This will output all values exceeding limits for this mnemonic (using the highest MSSID) and may manually be compared against the values reported in postcheck.tmp

- Choose two or more mnemonics from different subsystems that have reported state violations. Using the earliest reported state violation time and the latest reported violation time, select a time range for this cycle in TCAD and plot the selected mnemonics to the screen. The plotted mnemonic should be seen to exceed the states listed in the plot.

PASS/FAIL
Post Check report is generated in under 5 minutes per 100M of data in the p100.tmp file.
Limit violations are reported accurately.
State violations are reported accurately.

SU GP-B
INSP 31

RESULT: PASS FAIL (circle one) _____ initials

☑ REGRESSION

TEST

i. Output file from post_check.pro is called post_check.tmp. Run post_check.pro to generate output files from a tmminmax.tmp file and tmbadstate.tmp file.
ii. Do the UNIX command "diff" on each binary file in the released version of lasp against test version of lasp. (e.g. 'diff filenameA filenameB' where filenameA and filenameB are the output files from the old and new versions of process_gps.pro, respectively). Binary diff should show no differences between the files.

Any differences should be explained by changes made to code. Documentation must be produced to explain any differences made by code (such as a revision of S0613). If there were no post_check.pro, gpb_db.pro or dbroutines.so code changes, there should be no differences.

PASS/FAIL
Each comparison must yield no difference between released and new code output files.

RESULT: PASS FAIL (circle one) _____ initials

10.3 TDP7: Process GPS

☐ (B) ☑ (R) _____ initials
Test Case Verification Number: TDP7

☐ BASELINE

INTRODUCTION
The baseline test proves by analysis that all GPS data is being properly processed by TDP from the binary contents of the Level 0 database, which originate from raw, Gravity Probe B spacecraft binary files, to Level 1 data.

SU GP-B
INSP 31

APPROACH
Use an independent verification program, TERLgps.pl, which reads telemetry packets (APID 100 type) from telemetry where format=155, 156, 215, 216 and 217 and interprets the contents of the GPS-specific bytes.

Query the Sybase Level 0 Packets table, by Format ID, to generate a file of p100.tmp format. Process this file using TERLgps.pl and compare it to its corresponding Level 1 data that has already been processed using process_gps.pro (TDP).

FEATURES TO BE TESTED

- Accuracy of special GPS processing routines
- Data in formats 155, 156, 215, 216 and 217

FEATURES NOT TO BE TESTED

- Data in any other telemetry formats
- Speed of GPS processing routines

TESTS

i. Testing starts with a p100.tmp file that is created by querying the existing Sybase Level 0 Packets table (using a "where" statement on Format_ID), or has been produced in a previous test, such as in the section above. This p100.tmp file must have data in one of the formats listed above or the verification will be invalid.

ii. The p100.tmp file is processed using the independent verification program TERLgps.pl and then compared against the data found in the Sybase Level 1 database in the GPS packets and telemetry tables for the same spacecraft times.

iii. There may be existing data in the Sybase Level 1 database already processed by TDP using the process_gps.pro program. If not, run process_gps.pro with the keyword /CHECKONLY to generate the GPS table data from the p100.tmp file. Output files from process_gps.pro are named gps_pkt.tmp and gps_blk.tmp.

iv. The data from TERLgps.pl is exported to ASCII files and the existing Sybase Level 1 data (or that generated from process_gps.pro in /CHECKONLY mode) is exported to corresponding files.

v. Each set of ASCII files is compared using UNIX "diff" commands.

PASS/FAIL

The diff commands should yield null results, corresponding to no differences in TDP processing (process_gps.pro) vs. TERLgps.pl processing.

RESULT: PASS FAIL (circle one) _____ initials

☒ REGRESSION

TEST

i. Output files from process_gps.pro are gps_pkt.tmp and gps_blk.tmp. Use keyword /CHECKONLY to generate output files from a p100.tmp file (or packets100.tmp file).

ii. Do the UNIX command "diff" on each binary file in the released version of lasp against test version of lasp. (e.g. 'diff filenameA filenameB' where filenameA and filenameB are the output files from the old and new versions of process_gps.pro, respectively). Binary diff should show no differences between the files

Any differences should be explained by changes made to code. Documentation must be produced to explain any differences made by code (such as a revision of S0613). If there were no process_gps.pro, gpb_db.pro or dbroutines.so code changes, there should be no differences.

Please note that an MSS database change may affect this program. Code on this program should be checked on change of MSS versions if specific monitors are changed in MSS, per the MSS import process document, P0908. See VDD and code for more details.

PASS/FAIL

Each comparison must yield no difference between released and new code output files.

RESULT: PASS FAIL (circle one) initials

10.4 TDP8: TandO, or "Timing and Orbit"

☐ (B) ☒ (R) initials

Test Case Verification Number: TDP8

INTRODUCTION

For MSS delivery 3.3.3 and higher, the TandO.pro routine decommutates the first 81 bytes of each 32K telemetry format and places specific timing and GPS monitors in two separate tables. This routine creates two files (gps_timing.tmp, gps_orbit.tmp), whose contents are in turn stored in two data tables: GPB_L1..GPS_Timing and GPB_L1..GPS_Orbit. The first part of TandO handles timing mnemonics, the second part handles orbit mnemonics.

APPROACH

Simple comparison of the values of specific mnemonics in two separate data files over the same five-minute time interval.

FEATURES TO BE TESTED

- Accuracy of parsing done by tando.pro against the accuracy of gpb_tdp_L1.pro

FEATURES NOT TO BE TESTED

- Speed of tando.pro
- Fetching of any data outside the first 81 bytes of any 32K telemetry format.

TESTS

Over the same five-minute interval select the following data from the specified Sybase data tables:

i. Over the same five-minute interval select the following data into tempdb tables from the specified Sybase data tables:
   a. SF_Frame_Count, SQ_SciVehTime32, SQ_SciVehTime8, SQ_Sci10HzTime, SQ_PPSv16F_Time, SCT_Cycle and SCT_VTCW from GPB_L1..GPS_Timing.
   b. SF_Frame_Count, SQ_SciVehTime32, SQ_SciVehTime8, SQ_Sci10HzTime, SQ_PPSv16F_Time, SCT_Cycle and SCT_VTCW from GPB_L1..TManalog.
   c. SF_Frame_Count, SP_RecvrMode1, SP_GPS_WeekNum1, SP_GPSTimeWeek1, RP_ECEFPosX, RP_ECEFPosY, RP_ECEFPosZ, RP_ECEFVelX, RP_ECEFVelY, RP_ECEFVelZ, SP_GPS_Digit_B, SP_GPS_Fract, SP_Time_Trans from GPB_L1..GPS_Orbit.
   d. SF_Frame_Count, SP_RecvrMode1, SP_GPS_WeekNum1, SP_GPSTimeWeek1, RP_ECEFPosX, RP_ECEFPosY, RP_ECEFPosZ, RP_ECEFVelX, RP_ECEFVelY, RP_ECEFVelZ, SP_GPS_Digit_B, SP_GPS_Fract, SP_Time_Trans from GPB_L1..TManalog.

ii. Using the Sybase bcp utility, bcp out the four tempdb tables created in step (i). Do an ASCII diff and visual inspection on the data in the exported tables created in steps (a) & (b) and then against those created in steps (c) & (d).

SU GP-B INSP 31

PASS/FAIL

Visual inspection is required because the tando.pro data is stored in its spacecraft natural and most dense form (4 byte floats and 2 and 4 byte long integers). Therefore, it will never directly nor binarily agree with TDP Level 1 data that is stored in 8 byte floats.

Accordingly, TIMINGget and ORBITget (.pl) retrieve and convert the data into human engineering values to be compared. Plus, tando.pro filters out a bunch of imperfect "orbit" conditions by SF_MSS_ID, by non science 32k data, by SF_Frame_Count, and by byte location (first 81 bytes), in conjunction with

assembling only data from same orbit solution, and by tossing data of RecvrMode1 not x50 (80 decimal).

There should be some differences with respect to the "T" part of tando.pro (the Timing output in steps a & b) and Level 1 analog data, as that tando.pro only extracts data from: the first 81 bytes (555 format), the 32k science formats, of SF_MSS_ID >= 3.3.3.

There should be definite differences in the "O" part of tando.pro (the Orbit output in steps c & d) with respect to Level 1 analog data.

There should be no differences in the comparison discussed in step (i) SU GP-B INSP 31

RESULT: PASS FAIL (circle one) _____ initials

☐ REGRESSION

FEATURES TO BE TESTED
- Consistency of data created by tando.pro between one code release and the next

FEATURES NOT TO BE TESTED
- Correctness of tando.pro
- Time it takes to perform tando.pro.

TEST
i. Output files from tando.pro are gps_timing.tmp and gps_orbit.tmp. Run tando.pro with the /CHECKONLY keyword to generate output files. Do this in the directories of the released version of the software and the new version of the software.
ii. Do the UNIX command "diff" on each binary file in the released version of lasp against test version of lasp. (e.g. 'diff filenameA filenameB' where filenameA and filenameB are the output files from the old and new versions of process_gps.pro, respectively). Binary diff should show no differences between the files.

Any differences should be explained by changes made to code. Documentation must be produced to explain any differences made by code (such as a revision of S0613). If there were no significant MSS changes to the first 81 bytes of every 32K format, to tando.pro, gpb_db.pro or dbroutines.so code changes, there should be no differences.

PASS/FAIL
Each comparison must yield no difference between released and new code output files.

RESULT: PASS FAIL (circle one) _____ initials

10.5 TDP12: Average Level 1 TManalog data

☒ (B) ☒ (R) _____ initials

Test Case Verification Number: TDP12

INTRODUCTION
In order to control the size of the telemetry data in the Level1 database, the TManalog table is averaged on five-minute centers. The routine used to run this process is average_analog.pro, and the averaged data is saved in the TMaverage table. To decide which mnemonics are to be averaged, the routine reads TMavgid data table for real, analog values from 32K format data.

Full-resolution (all of the analog data) is all data for monitors sampled at the spacecraft rate of every 10th of a second. The averaging of TManalog data is baselined for every few weeks; however, the frequency of the averaging process can be changed by the database administrator. The content of the averaged data consists of the minimum, maximum, average and standard deviation values (per five-minute interval) for each monitor sampled over that five-minute interval, and the number of times the monitor was sampled during the interval.

According to requirement MOC-017, the data processing shall be able to retrieve and display averaged programmable telemetry data for a selected monitor from the entire mission set within 1 hour of a request.

❑ BASELINE

APPROACH
To test speed, run TCAD, choose a monitor and measure the amount of time it takes to show data. Do this with several different high-frequency sampled monitors for performance measurements. To show capability of viewing averaged data, run TCAD, choose a monitor and select a time interval and cycle number that retrieves averaged data.

To test accuracy and correctness of average_analog.pro, select a monitor from TCAD with the "Averaged" box checked in TCAD's "setup" options. After plotting data, zoom in on any segment of data and check the accuracy of the given numbers via visual inspection of the plot. Alternatively, if zooming does not provide an adequate inspection, get the data via table and perform a numerical inspection if desired.

FEATURES TO BE TESTED
- Time it takes to retrieve averaged data for display from the TMaverage table.
- Accuracy of averaged data w.r.t. minimum, maximum and averaged points.
- Correctness of average_analog.pro

FEATURES NOT TO BE TESTED
- Deletion of any data from the TManalog table (which is not done by average_analog.pro)
- Averaging SNanalog data (no 1K or 2K data is averaged by this routine)

TESTS
i. First, check the amount of data in the average data table, GPB_L1..TMaverage via a select count(TMID) statement in Sybase. If there is no data, create some using P0905, "Processing Data from High-resolution to an Averaged Data Set".
ii. Start TCAD and chose monitors to display. The choice of which monitor(s) to display may be influenced by the existence of those TMIDs in the TMaverage data table. That is, one should choose monitors that will return data instead of an empty set.
iii. Time the arrival of results.
START TIME _____ END TIME _____
iv. Inspect the results of the averaged data against the analog data. Does the inspection yield a satisfactory averaging? _____Yes _____No

PASS/FAIL
Successfully retrieve and display averaged programmable telemetry data for a selected monitor from the entire mission set within one (1) hour. Averaged values of data passed visual inspection for plot or table values.

RESULT: PASS FAIL (circle one) _____ initials

☒ REGRESSION

FEATURES TO BE TESTED

- Consistency of data created by average_analog.pro between one code release and the next

FEATURES NOT TO BE TESTED
- Correctness of average_analog.pro
- Time it takes to retrieve averaged data for display from the TMaverage table.

TEST AVERAGE_ANALOG, 2001, 249, cycle=-1322, /spew/checkonly

iii. Output file from average_analog.pro is called tmaverage.tmp. Run average_analog.pro with the /CHECKONLY keyword to generate an output file. Do this in the directories of the released version of the software and the new version of the software.

iv. Do the UNIX command "diff" on each binary file in the released version of lasp against test version of lasp. (e.g. 'diff filenameA filenameB' where filenameA and filenameB are the output files from the old and new versions of process_gps.pro, respectively). Binary diff should show no differences between the files.

Any differences should be explained by changes made to code. Documentation must be produced to explain any differences made by code (such as a revision of S0613). If there were no average_analog.pro, gpb_db.pro or dbroutines.so code changes, there should be no differences.

342 TMIDs

PASS/FAIL
Each comparison must yield no difference between released and new code output files.

RESULT: PASS FAIL (circle one) _____ initials

11 TEST CASES FOR CORE PROGRAMS

VERIFICATION REQUIRED • Baseline (B) • Regression (R)	FILE (under src/tdp)	UNIX date & filesize on program file	TEST NAME	TEST SECTION	VERIFIED BY (initials)	DATE
	gpb_db.pro	SEE ATTACH 11	CORE1	Section 11.1		
	gpb_time.pro		CORE1	Section 11.1		
	Generic_time.pro		CORE1	Section 11.1		
	db_routines.so		CORE1	Section 11.1		

11.1 CORE1: dbroutines.so, gpb_db.pro, generic_time and gpb_time.pro

Underlying code layers include files named dbroutines.so, gpb_db.pro, generic_time.pro and gpb_time.pro. These are underlying software layers between all tdp and tcad routines. If any of these files have changed, do regression testing for TDP2, TDP5, TDP7, TDP8, TDP9 and TDP12, along with TCAD regression tests listed in separate document.

REQUIRED?
Regression of TDP2 TDP5 TDP7 TDP8 TDP9 TDP12 and ~~TCAD P0950~~ ~~SNGP-B~~ INSP 31
YES NO (circle one) _____ initials

12 GLOSSARY

This section contains an alphabetic list and definitions of all acronyms used in the document, all proper nouns, and any words used in a non-standard way.

Word	Detail
CSCI	Computer Software Configuration Item
LASP	Laboratory for Atmospheric and Space Physics, University of Colorado
moc-server	Host name of the SUN computer that is the primary server for the MOC.
Science server	Host name of the SUN computer which is the primary server for science LAN
SAFS	Standard Autonomous File Server (GSFC facility)
TCAD	Telemetry Checking, Analysis, and Display
TDP	Telemetry Data Processing

Successful Completion of P0949:

RUN BY (signature) Ron Shulkowski for Paul McGoon Date: 3/24/04

Print Name: _____

ALL TESTING PASSED (CIRCLE ONE): YES – PASSED NO ~~MCR FILED~~ DLOG created in BCP throughput

QA Review of process Doreen Pom SU GP-5 INSP 31 Date: 3/24/04

O/K to release s/w Ron Shulkowski 3/24/04

Attnch 9.0-1

```
version of "IPDUsplice.pro":                    2.2
version of "L0_to_L1.pro":                      2.2
version of "L0_to_L1_display.pro":              2.2
version of "SCCS":                              Unknown
version of "auto_L1.exp":                       Unknown
version of "auto_import.exp":                   Unknown
version of "average_analog.pro":                2.2
version of "gpb_derive.pro":                    2.2
version of "gpb_tdp_L0.pro":                    2.2
version of "gpb_tdp_L1.pro":                    2.2
version of "gpb_tdp_fmtrpt.pro":                2.2
version of "gpb_tdp_setdir.pro":                2.2
version of "idl_startup.pro":                   2.2
version of "10211":                             Unknown
version of "10211_wrapper":                     Unknown
version of "moc_auto.cron":                     Unknown
version of "post_check.pro":                    2.2
version of "process_gps.pro":                   2.2
version of "process_tcor.pro":                  2.2
version of "scan.pro":                          2.2
version of "science_auto.cron":                 Unknown
version of "ssbuild.pro":                       2.2
version of "tando.pro":                         2.2
version of "tdp":                               Unknown
version of "tdp.cron":                          Unknown
```

```
sccs prt -y gpb_db.pro
SCCS/s.gpb_db.pro:      D 2.3   04/03/14 22:36:05 pmcgown       30 29   00080/00003/01886 Enhanced TMinfo for TLM formats, frames, bytes, and polynomials

sccs prt -y gpb_time.pro
SCCS/s.gpb_time.pro:    D 2.1   04/02/06 05:16:24 ron   10 9    00000/00000/00247 sccs edit -r2 -t SCCS ...followed by: sccs delget 'sccs tell'

sccs prt -y generic_time.pro
SCCS/s.generic_time.pro:        D 2.1   04/02/06 05:16:24 ron   10 9    00000/00000/00315 sccs edit -r2 -t SCCS ...followed by: sccs delget 'sccs tell'

sccs prt -y db_routines.so
SCCS/s.db_routines.so:  D 2.3   04/03/18 02:53:28 local 12 11   06319/04983/00821 copied from src dir after .cct compile - rjs
```

Attachment 11

W. W. Hansen Experimental Physics Lab http://einstein.stanford.edu
Stanford University Stanford CA 94305-4085

Att 8.1-1

-J Spencer

→ L1 throughput TDP

A) > Confirm that L1 decom time is less than 20% different between lasp versions, and less than 20 minutes overall for a 12-hour starting raw data file size.

"starting raw data file size" is filesize of vc.land2*.bin or other data file. # of hours is determined by $\frac{\text{filesize}}{275\text{mg}} * 24 =$ # of hours in data file

a

New 8.1

B) > Confirm that bcp time of tmanalog.tmp would be less than 30 minutes for a 12 hour starting raw data file. Divide size of tmanalog.tmp by 18 bytes, then divide result by 18,500 rows per second. Final result number is the # of seconds that bcp would take, which should be <1800. Compensate as appropriate for files > 12 hours.

L1 Throughput test March 23, 2004 | AH 8.1-2 J Spencer

A). /apps/supported/lasp-2.3/data/T22/L1/tdp-L1.sum

Says decom time was 5.97 minutes for ~ 5 hours of data.

in /apps/supported/lasp-2.3/data/T23/L1/tdp-L1.sum

Says decom time was 5.84 minutes for the same ~5 hours of data.

We show an improvement of 2%, and we pass the throughput test, part A.

3/23/04 Test Engineer.
Jennifer Spencer

B). Processed an 18 hour telemetry file, got bcp speeds of ~13,270 rows per second into TManalog. Sybase seems inappropriately slow for this transfer, but this is unrelated to lasp software but may be related to other Sybase processes. lasp software OK to release but DLog must be opened on performance.

3/24/04 Test Engineer
Jennifer Spencer

SU GP-B INSP 31

DATE/TIME: 2004-03-24 18:45:15.183
AUTHOR: Ron Sharbaugh
TITLE: DLOG - lasp-2.3 failed bcp timing

T
TEXT: lasp-2.3 failed bcp timing

The spec for BCP is ~18000 rows/sec. Lasp-2.3 test was only ~13000 rows per second. T3ba
tteries are ok.

AS-RUN

P0950 Rev. E
February 26, 2004
ATTACH TO S093-B

W.W. Hansen Experimental Physics Laboratory
STANFORD UNIVERSITY
STANFORD, CA 94305 – 4085

Gravity Probe B Relativity Mission

Operational Procedure for Regression Testing of TCAD

P0950 Rev E
2/26/2004

Approvals

NAME	SIGNATURE	DATE
Samantha Patterson *Software Engineer*		
Jennifer Spencer *Data Processing Lead*		
Ron Sharbaugh *S/W Manager*		
Marcie Smith *MOC Project Manager*		
Kelly Burlingham *Software Quality Engineer*	ECO 1480	2/27/04

Required Signatures prior to Execution

NAME	SIGNATURE	DATE
NAME: Samantha Patterson *Test Engineer*	Samantha Patterson	3/23/04
Kelly Burlingham Dorrene Ross *Software Quality Engineer*	Dorrene Ross	3/23/04

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

Table of Contents:

1 REVISION HISTORY

REV	DATE	AUTHOR	COMMENTS
-	16 Oct 2002	SAP	initial version
A	30 May 2003	SAP	Updated document to include new features. Included baseline testing in addition to regression. Added definition to test procedure from earlier notes.
B	31 July 2003	SAP	Updated document to include new TCAD graphical features.
C	12 Sept 2003	SAP	Reorganization of Display test case, addition of SN data test cases.
D	13 Nov 2003	JLS	Updated sections 7.1, 10 (test section list), 10.3, 10.4, 10.7, 10.9, 10.19, 10.24, 10.25 and 10.28 to reflect MCR implementation.
E	26 Feb 2004	JLS	Updated sections 10.19, 10.25 and 10.26 to reflect code additions per MCRs.

2 SCOPE

This Test Plan Document details the TCAD software package and how to both baseline and regression test it.

3 OPERATIONAL PERSONNEL

Jennifer Spencer
Samantha Patterson
Qualified QA Rep: Kelly Burlingham

4 QUALITY ASSURANCE PROVISIONS

Quality Assurance must be given 24 hour notification before this test is run; presence is at their discretion.

QA Notified Date & Time: 3/19/04 10:00AM By: Ron Sharbaugh QA Initials: DMR

5 REQUIREMENTS & CONSTRAINTS

5.1 Hardware and Software Requirements

Operations are performed on the Sun server machine known as "science". This application require that sybase be running and the GPB_L0, GPB_L1 and GPB_L1A databases are available ~~for reading~~.

SU GP-B
INSP 31

5.2 Configuration Requirements

The operator must be logged into the ~~server~~ client "science" as a user in the group 'users' The user should reserve a directory for test data and save this data for QA.

Directory: /apps/supported/lasp-2.3/

5.3 Constraints

Constraint	Risk
Graphical display	TCAD is a graphical application and the X-windows DISPLAY environment variable and xhosts file must be set to use the TCAD system. e.g. setenv DISPLAY machinename:0.0
L0 & L1 databases online	This system accesses the GPB_L0, GPB_L1 and GPB_L1A databases. All must be accessible.

6 REFERENCE DOCUMENTS

Document No.	Document
S0331	Data Management Plan
S0401	Stanford Post-Processing Operations for Science Mission Data
S0476	MOC Configuration Control, IONET LAN
S0515	MOC VDD for EVENTget.pl
S0613	TDP/TCAD Software Release (Design Document)
S0634	Snapshot Database Reading Utility
S0638	DBRO Data Retrieval and Display
S0717	TCAD User's Guide
P0827	Displaying APID 100, 300 & 301 Telemetry Data in the Non-Real-Time System
P0832	Event Display in Non-Real-Time System
LMMS/P479910O	LM VDD: Real-time Ground Software
MO-02	MOC Requirements

7 SOURCE CODE PATH ON SCIENCE

This software is installed under the apps/supported/lasp-X.X/src directory structure at SU on the science network.

7.1 /tcad

Program files and versions are as follows:

File	RCS Version
Tcad	
Tcad.pro	
Tcad_main.pro	See
Idl_startup.pro	
Tcad_main.pro	Attachment
Tcad_tminfo.pro	
Tcad_values.pro	1
Tcad_display.pro	
Tcad_panel_components.pro	
Tcad_dbroget__app_list.pro	
Tcad_display_timetype.pro	
Tcad_time_string.pro	
Tcad_display__save.pro	
Tcad_display__restore.pro	
Tcad_table__setup.pro	
Tcad_table.pro	
Tcad_table__prepare_print.pro	
Tcad_plot.pro	
Tcad_plot__prepare_print.pro	
Tcad_plot__setup.pro	
Tcad_plot__rtmenu.pro	
Tcad_plot__yscale.pro	
Tcad_plot__function.pro	
Tcad_plot__col.pro	
Tcad_clr_index.pro	
Tcad_preferences.pro	
Tcad_display__search_list.pro	
Auto_plot	
Auto_plot.pro	
Auto_plot.sh	
Auto_plot_main.pro	

7.2 /tcad/analyze

Programs and files are as follows:

File	RCS Version
Tcad_eventget.pro	
Tcad_mroget.pro	See
Tcad_dbroget.pro	
snapshot.pro	Attachment
Tcad_formatrpt.pro	
Tcad_ssget.pro	2

7.3 /db

Programs and files are as follows:

File	RCS Version
Gpb_db.pro	See
Gpb_time.pro	
Db_routines.so	Attachment 3
Generic_time.pro	
Time_lib.pro	

7.4 SCRIPTS for TCAD

File	RCS Version
/apps/supported/scripts/tcad → /apps/supported/lasp-x.x/scripts/tcad	

~~8 TEST ENVIRONMENT~~

SU GP-E INSP 31

This software is tested on the ~~science server~~ client at SU under the following configuration:

Software Configurations	Configuration Number (fill in)
IDL Version	5.4
Sybase ASE	12.5
MSS/GSW	

MSS/GSW not used for this test

SU GP-B INSP 31 3/23/04

9 OPERATING SYSTEM

This section describes the other software that is required to be in place for implementation of this delivery.

Operating System	Minimum Version	Description
Solaris Operating System	2.8	SUN's UNIX operating system.

10 TEST CASES FOR TCAD

If the date or RCS Version on any of the following files has changed, the tests in the corresponding section number must be run and checked off for verification. Throughout all phases of testing, all constraints (specified in section 4 of this document) must be met. Additional constraints may be set on an individual test case basis as noted in the Test Sections below. The objective of testing in TCAD is to confirm that the data displayed is accurate to the contents of the database and that this data is easy to access and display via the GUI. As such, the focus of testing for TCAD is about the presentation of the data.

LASP version being tested: 2.3

VERIFICATION REQUIRED • Baseline (B) • Regression(R)	File	Test Name	Verified By (initials)	Date
	Tcad	TCAD MAIN		
	Tcad.pro	TCAD MAIN		
	Tcad_main.pro	TCAD MAIN		
	Tcad_tminfo.pro	TMINFO		
	Tcad_display.pro	DISPLAY, DISPLAY TIME, DISPLAY MENU, DISPLAY HOT, DISPLAY WILD, DISPLAY FMTS, DISPLAY OUT		
	Tcad_display__save.pro	SAVE RESTORE		
	Tcad_display__restore.pro	SAVE RESTORE		
	Tcad_display__timetype.pro	CYCLE SEL		
	Tcad_initial_printer.pro	INIT PRINT		
	Tcad_panel_components.pro	DISPLAY TIME		
	Tcad_table__setup.pro	TABLE SETUP		
	Tcad_table__prepare_print.pro	TABLE PRINT		
	Tcad_table.pro	TABLE		
	Tcad_plot__setup.pro	PLOT SETUP		
	Tcad_plot.pro	PLOT		
	Tcad_plot__prepare_print.pro	PLOT PRINT		
	Tcad_plot__rtmenu.pro	PLOT RTMENU		
	Tcad_plot__function.pro	PLOT RTMENU		
	Tcad_plot__yscale.pro	PLOT RTMENU		
	Tcad_plot__col.pro	PLOT RTMENU		
	Tcad_preferences.pro	USER PREF		
	Tcad_display__search_list.pro	MNE SEARCH		
	Tcad_eventget.pro	EVENT GET		
	Tcad_mroget.pro	MRO GET		
	Tcad_dbroget.pro	DBRO GET		
	Tcad_dbroget__app_list.pro	DBRO GET		
	snapshot.pro	SNAPSHOT		
	Tcad_formatrpt.pro	FMTRPT		
	Tcad_ssget.pro	SSGET		
	Tcad_values.pro	CORE2		
	Gpb_db.pro	CORE2		
	Gpb_time.pro	CORE2		
	Db_routines.so	CORE2		
	Auto_plot	AUTOPLOT		
	Auto_plot.pro	AUTOPLOT		
	Auto_plot.sh	AUTOPLOT		
	Auto_plot_main.pro	AUTOPLOT		

SU GP-B INSP 37 3/23/04 SU GP-B INSP 37 3/23/04 SU GP-B INSP 37 3/23/04

10.1 TCAD MAIN: TCAD Main Menu

☐ (B) ☒ (R)

Test Case Verification Number: TCAD MAIN

INTRODUCTION
This test case is used for verifying initialization of TCAD and the functionality of the main menu. All further testing is based on the sucess of this screen initializing correctly. No additional testing may be performed until this step completes successfully.

APPROACH
Point and click. Baseline and regression tests for this section are identical.

FEATURES TO BE TESTED
- ❑ Generation of Analyze menu.
- ❑ Access to child windows.
- ❑ Termination of child windows on application exit.

FEATURES NOT TO BE TESTED
- The Check menu.

BASELINE TESTS client SU GP-B INSP 31 3/23/04

i. Login on science ~~server~~ and type 'tcad' at Unix command line.
ii. Right-click on the 'Analyze' menu. Record the list of applications in the menu. Exit the application and create a dummy file in the analyze directory TCAD test code with a command like 'echo nothing > test_files.pro'. Reload the application and this file should now appear in the analyze menu. If all files are removed from the analyze directory, this menu item should disappear.
iii. Click each item in each menu. Every sub-window should open when called.
iv. With multiple child windows open, select 'Quit' from the file menu. All windows should close except the window from which TCAD was run.

BASLINE TEST PASS CRITERIA
Passes if TCAD initializes, no error messages are displayed in the startup screen and all menu options are accessible and close properly.

REGRESSION TESTS
i. Load current version of TCAD and visually compare screen with previous version.

REGRESSION TEST PASS CRITERIA
Windows appear similar and respond similarly.
RESULT: PASS FAIL (circle one) _____ initials

10.2 TMINFO: TM Info screen.

☒ (B) ❑ (R)

Test Case Verification Number: TMINFO

INTRODUCTION
Tests the validity of the information displayed by this GUI. The TM Info screen allows the user to fetch information about a particular mnemonic including but not limited to, its TMID, latest MSS version, units, limits, and calibrations.

APPROACH
For the baseline test, fetch data via GUI, compare against the MSS database. The regression test may simply compare a screen snapshot of the previous version with the current version.

FEATURES TO BE TESTED
- ❑ Verifies that the information requested is the information displayed.

FEATURES NOT TO BE TESTED
- Validity of contents of the MSS database.

BASELINE TESTS
i. Select a mnemonic from a subsystem.
ii. Select the information on the same mnemonic from the current MSS database
iii. Compare the values in the display area with the information selected from the MSS database or screen shots from prior version. *
iv. Take no less than three samples of mnemonics with different data types in different subsystems.

BASELINE TEST PASS CRITERIA
Passes if TM Information screen appears, mnemonics can be selected, and data displayed matches the data from the current MSS database loaded on the science system.

REGRESSION TESTS
i. Display a mnemonic in the previous version of TCAD. Visually compare it with the same mnemonic in the current version of TCAD.

SU GP-B INSP 31 3/23/04

REGRESSION TEST PASS CRITERIA
Display for mnemonics is identical between versions.

ATTACHMENT(S): attach 4.txt

RESULT: PASS FAIL (circle one) P initials

10.3 DISPLAY: TCAD Display data window : telemetry selection components

❑ (B) ☒ (R)

Test Case Verification Number: DISPLAY

INTRODUCTION
Test cases for the subsystems, available telemetry items, and telemetry items selected components of the TCAD Display Data screen.

APPROACH
Point and click, visual inspection.

FEATURES TO BE TESTED
- ❑ Subsystems selection
- ❑ Available Telemetry Items Selection
- ❑ Removal of selected items
- ❑ Recall last session time & cycle choice

FEATURES NOT TO BE TESTED
- Cycle and time components
- Wildcard mnemonic selection
- Displaying data
- Configuring displays
- Format selection

BASELINE TESTS

i. Select a subsystem from the selected list by clicking on it.
ii. Select a telemetry item from the available telemetry items list by clicking on it.
iii. Repeat previous steps for 4 different subsystems.
iv. Double-click a mnemonic in the 'Telemetry Items Selected' section.
v. Single-click a mnemonic in the 'Telemetry Items Selected' section and click 'Remove Selected Item Only'
vi. Single-click 'Remove All Items From List'.
vii. Make note of cycle number and time: _____ and any mnemonics selected: _____.
viii. Exit TCAD
ix. Restart TCAD and call up the Display Data screen.
x. Compare listed cycle number and time and mnemonics with those written above.

SU GP-B INSP 31 3/23/04

BASELINE TEST PASS CRITERIA
i. When subsystem is selected, a list of corresponding mnemonics appears.
ii. When a telemetry item is selected, it appears in the 'Telemetry Items Selected' area.
iii. When a telemetry item in the 'Telemetry Items Selected' list is double-clicked, it is removed from the list.

iv. Clicking 'Remove Selected Item Only' removes the currently selected item from the 'Telemetry Items Selected' list.
v. Clicking 'Remove All Items From List' removes all items in the selection list.
vi. Comparison of written cycle, time and mnemonics above is exact against the cycle, time and mnemonics that appear when TCAD's Display Data window is shown at startup.

GP-B INSP 31 3/23/04

REGRESSION TESTS

i. Select a telemetry item.
ii. Select a cycle and time and telemetry item, then close Display Data window

REGRESSION TEST PASS CRITERIA

- Item is added to list.
- Restoring Display Data window shows cycle, time & telemetry item are the same as before.

RESULT: PASS FAIL (circle one) _____ initials

10.4 DISPLAY TIME: Cycle and time panel components.

☐ (B) ☒ (R)

Test Case Verification Number: SAVE RESTORE

INTRODUCTION
This section details testing of time-entry routines. This code represents a common block of data that is resused by the Display Screen, DBRO get, MRO get, Event get, and Snap read. Repeat testing is not required as the same code is being used in each section however, concurrency should be tested.

APPROACH

FEATURES TO BE TESTED

- ☐ Selection of cycle.
- ☐ Window resize.
- ☐ Autofill time button

FEATURES NOT TO BE TESTED

- Cycle data validation.

BASELINE TESTS

i. Input cycle and time range from Display Data, DBRO get, Event get, or MRO get.
ii. Select various combinations of time ranges. Date format, numeric formats, +hours, outside cycle range, inside cycle range, with good and bad formatting of text and numbers.
iii. Clear times and press the Autofill Times button. Select the full time range and compare with the information for the cycle in the cycle selection window.
iv. Select 'Last N hours' time ranges and watch start and stop time fields update.

BASELINE TEST PASS CRITERIA

SU GP-E INSP 3/23/04

- Program should pop-up dialog when submitting bad data and prevent user from moving forward until the offending field is corrected.
- Open two windows that use panel components then close one. The other should still operate correctly.
- Autofill time for the entire range roughly corresponds to the stated time range in the cycle information, unless the cycle selected predates the Files table in which case a popup-dialog will notify the user that they cannot use autofill for this section. In that event, chose a new cycle and retest.
- Changing hours to view modifies the start time accordingly except in the case where the number of hours chosen would make the start time earlier than the start of the cycle in which case the start time is set to the minimum for the cycle.

REGRESSION TESTS

i. Select a cycle.

REGRESSION TEST PASS CRITERIA

- Cycle is selected.

RESULT: PASS FAIL (circle one) P initials

10.5 DISPLAY MENU: Menu items in the TCAD DISPLAY screen

☒ (B)

Test Case Verification Number: DISPLAY MENU

INTRODUCTION
Test cases for all items accessible from the menus in the TCAD display window.

APPROACH
Point and click, visual inspection.

FEATURES TO BE TESTED

- ❑ Menu access to sub-screens

FEATURES NOT TO BE TESTED

- Functionality of sub-screens

TESTS

xi. Make changes to the display (input times, select a cycle, choose mnemonics, etc) Select 'File|Clear Configuration'
xii. Click 'File|Restore Configuration'
xiii. Click 'File|Save Configuration'
xiv. Click 'Option|Cycle Select'
xv. Click 'Option|Preferences'
xvi. Click 'PLOT' under 'DISPLAY AS' on screen.
    a. Click 'File|Print'
    b. Click 'Option|Display Options'
xvii. Click 'TABLE' under 'DISPLAY AS' on screen.
    a. Click 'File|Print'
    b. Click 'Option|Display Options'
xviii. Click 'File|Close'

TEST PASS CRITERIA

vii. Changes made on screen are cleared.
viii. The restore file window appears.
ix. The save file window appears.
x. The cycle selection window appears.
xi. The user preferences screen appears.
xii. The print window for plots appears; the display options for plots appears.
xiii. The print window for tables appears; the display window for tables appears.
xiv. The display window (and any open child windows) close.

RESULT: PASS FAIL (circle one) P initials

10.6 DISPLAY HOT: Display Data Screen Hotkey functions

☒ (B)

Test Case Verification Number: DISPLAY HOT

INTRODUCTION

This test case is a verification for hotkeys in the TCAD DISPLAY DATA screen.

APPROACH
Point and click, visual inspection.

FEATURES TO BE TESTED
- ❑ Hotkey functionality

FEATURES NOT TO BE TESTED
- Functionality of sub-screens.

TESTS
i. Enter data on screen, press CTRL-N
ii. Press CTRL-O
iii. Press CTRL-W
iv. Press CTRL-P
v. Press CTRL-D
vi. Press CTRL-Q

TEST PASS CRITERIA
i. Changes made on screen are cleared.
ii. The restore file window appears.
iii. The save file window appears.
iv. The print window appears.
v. The current settings are displayed to plot or table accordingly.
vi. The display window (and any open child windows) close.

RESULT: (PASS) FAIL (circle one) _____ initials

10.7 DISPLAY WILD: Wildcard lookup of mnemonics

☒(B)
Test Case Verification Number: DISPLAY WILD

INTRODUCTION
The 'Enter Mnemonic' panel of the TCAD_DISPLAY screen is an independent component which may be reused in other windows and tested independently.

APPROACH
Point and click, visual inspection.

FEATURES TO BE TESTED
- ❑ Lookup of mnemonics
- ❑ Handling of many mnemonics
- ❑ Handling of few mnemonics

- ❑ Handling of no mnemonics
- ❑ Handling of too few characters entered

FEATURES NOT TO BE TESTED

- Available telemetry items screen

BASELINE TESTS

i. Enter a pattern when will not match any database mnemonics such as the word 'Eggplant'
ii. Enter a pattern which will match less than 5 database mnemonics such as 'BC_SA_*_Deploy'
iii. Re-enter this pattern as all-capitals and again as all lower-case.
iv. Enter a pattern which will match many mnemonics such as 'rf_force*'
v. Enter one or zero characters in the search text box

TEST PASS CRITERIA

- When no mnemonics match, a message window stating this appears.
- When less than 5 mnemonics match the query they are automatically added to the selection list.
- Changes in the case of the query do not change the number of mnemonics selected.
- When 5 or more mnemonics match the query the 'Available telemetry items' screen appears.
- If fewer than two characters are entered as a search criterion, a message window requests more characters.

RESULT: PASS FAIL (circle one) _____ initials

10.8 DISPLAY FMTS: Display Data Screen 1K/2K and 32K outputs

☒ (B)

Test Case Verification Number: DISPLAY FMTS

INTRODUCTION

1K/2K and 32K data are separate entities and are displayed overlaid in the plot window or side by side in the table display. This section tests the checkboxes that toggle output for each.

APPROACH

Point and click, visual inspection.

FEATURES TO BE TESTED

- ❑ Selection of 1K/2K and 32K data

FEATURES NOT TO BE TESTED

- Correct plotting of 1K/2K and 32K data

TESTS

i. Select a cycle that contains separated 1K/2K and 32K data and a valid time range.
ii. Select a mnemonic that has only 32K data.
iii. Select a mnemonic that has only 1K/2K data.
iv. Select a mnemonic that has both 1K/2K and 32K data.
v. Display the data.
vi. Toggle 1K/2K data and redisplay.
vii. Toggle 32K data and redisplay.

TEST PASS CRITERIA

- When plotted, mnemonics with mixed 1K/2K and 32K data are displayed as overlaid plots.
- When 1K/2K data is disabled, mnemonics that contain only 1K/2K data are blank. Mnemonics that contain mixed formats display only 32K data.
- When 32K data is disabled, mnemonics that contain only 32K data are blank. Mnemonics that contain mixed formats display only 1K/2K data.

RESULT: PASS FAIL (circle one) _____ initials

10.9 DISPLAY OUT: Display Data Output controls

☐ (B) ☒ (R)

Test Case Verification Number: DISPLAY OUT

INTRODUCTION

This test case handles the output controls on the TCAD Display Data screen. These controls include Plot/Table, Update Plot, New Plot, Display Table, Print, and Setup.

APPROACH

Point and click, visual inspection.

FEATURES TO BE TESTED

- ☐ DISPLAY AS Plot/Table selection
- ☐ The 'New Plot' button
- ☐ The 'Update Plot' button
- ☐ The 'Display Table' button
- ☐ The 'Print' button
- ☐ The 'Setup' button

FEATURES NOT TO BE TESTED

- 1K/2K 32K selection (See DISPLAY FMTS test case)
- Functionality of child windows.

BASLINE TESTS

i. Select a valid cycle and time range.
ii. Set 'Display As' to PLOT
    - Press 'New Plot'
    - Press 'Update Plot'
    - Press 'Print'
    - Press 'Setup'
iii. Set 'Display As' to TABLE
    - Press 'Display Table'
    - Press 'Print'
    - Press 'Setup'

BASELINE TEST PASS CRITERIA SU GP-B INSP 31 3/23/04

- When in plot mode, the data is displayed graphically.
- When in plot mode, if no plot windows exist, both 'New Plot' and 'Update Plot' bring up a new plot window.
- When in plot mode, if other plot windows exist, 'New Plot' brings up a new plot window.
- When in plot mode, if other plot windows exist, 'Update Plot' changes the displayed data in the plot window that had the last mouse-over.
- When in plot mode, the printer interface for plotting is displayed.
- When in plot mode, the plot setup screen appears when a 'Setup' is clicked.
- When in table mode, the data is as tab-delimited text.
- When in table mode, the printer interface for tables is displayed.
- When in table mode, the table setup screen appears when a 'Setup' is clicked.

REGRESSION TESTS

i. Select a valid cycle and time range.
ii. Set 'Display As' to PLOT. Click 'New Plot' and 'Update Plot'
iii. Set 'Display As' to TABLE. Click 'Display Table'

REGRESSION TEST PASS CRITERIA

- When in plot mode, the data is displayed graphically.
- When in table mode, the data is as tab-delimited text.

RESULT: PASS FAIL (circle one) _____ initials

10.10 SAVE RESTORE: TCAD Save and Restore screens.

☐ (B) ☒ (R)

Test Case Verification Number: SAVE RESTORE

INTRODUCTION

The baseline for this test verifies that the data saved by the user is the same as the data restored by the user. The regression test simply verifies that older versions of the save file can be loaded with the current tool.

APPROACH

Save a file and restore it or open a previously saved file.

FEATURES TO BE TESTED

- ❑ Generation of save file.
- ❑ Preservation of cycle and date/time selection.
- ❑ Storage of selected mnemonics
- ❑ Preservation of plot options.
- ❑ Preservation of user preferences.
- ❑ Restoration of individual plot settings if user setting is 'preserve'.

FEATURES NOT TO BE TESTED

- Use of save files in auto_plot

BASELINE TESTS

- Select a cycle, time range, and mnemonics in the display screen.
- Select 'Save' from the display screen's File menu.
- Give the configuration file a name and save it.
- Exit the display screen, then re-enter it.
- Select 'Restore' from the display screen's File menu.
- Load the previously saved file.

BASELINE TEST PASS CRITERIA

SU GP-B ASP 31 3/29/04

- File is created and now appears in 'Restore' list.
- Cycle, times, mnemonics, and plot and table configuration options should be restored exactly as they were set before the save.
- Settings in the plot setup screen are preserved.
- The user preferences for color and saving individual plot settings are preserved.
- If individual plot settings are preserved, the settings accessible from the right-click menu in the plot display are preserved.

REGRESSION TESTS

i. Load a saved file.
ii. Save the file with a different name.

REGRESSION TEST PASS CRITERIA

- The unix tool 'diff' shows the two files are identical.

RESULT: PASS FAIL (circle one) initials

10.11 CYCLE SEL: Cycle Selection Screen

☒ (B)

Test Case Verification Number: CYCLE SEL

INTRODUCTION
This section details testing of time selection window.

APPROACH
Point and click/visual inspection.

FEATURES TO BE TESTED
- ❑ Selection of a cycle.

FEATURES NOT BEING TESTED
- N/A

TESTS
- Select 'cycle' from Option menu in Display Data screen.
- Select real time or space-craft time.
- For spacecraft time, select a cycle.

TEST PASS CRITERIA
- Cycle item on Display Data screen reflects cycle number selected (Or real time)
- Selecting and displaying mnemonics for this cycle matches database content on a manual select from Sybase.

RESULT: PASS FAIL (circle one) initials

10.12 INIT PRINT: TCAD printer list generation.

❑ (B) ☒ (R)

Test Case Verification Number: INIT PRINT

INTRODUCTION

This section details testing of printer list generation.

APPROACH
Visual inspection.

FEATURES TO BE TESTED
- ❑ Generation of printer list.

FEATURES NOT TO BE TESTED
- Printer status.

SU GP-B INSP 31 3/23/04

BASELINE TESTS
- Select cycle, time range, and mnemonic.
- Click 'Print' button in Display Data screen.
- Select drop-down list of printers.
- Change mode from plot to table and print the data a second time.

BASELINE TEST PASS CRITERIA
- All printers configured for this system should be displayed in the list.
- Print to each printer.

3/23/04 SU GP-F INSP 31

Regression ~~BASELINE~~ TESTS
- Select cycle, time range, and mnemonic.
- Click 'Print' button in Display Data screen.
- Select drop-down list of printers.
- Change mode from plot to table and print the data a second time.

3/23/04 SU GP-F INSP 31

Regression ~~BASELINE~~ TEST PASS CRITERIA
- All printers configured for this system should be displayed in the list.
- Print to ~~each~~ one printer.

RESULT: (PASS) FAIL (circle one) _____ initials

Attachments 6 & 7

10.13 TABLE: Output to Table

❑ (B) ☒ (R)

Test Case Verification Number: TABLE

This section details testing of configuration for outputting data to table format and provides criteria for validation of displayed data and retrieval times.

APPROACH
Select a short time range for a single mnemonic for data validation. Select a long time range for several mnemonics for speed verification.

FEATURES TO BE TESTED

- ❑ Data display
- ❑ Retrieval speed.

FEATURES NOT TO BE TESTED

- Data Integrity

BASELINE TESTS

i. Select a small range of data for a single analog mnemonic, set output to table and under 'Setup' set the table time output to 'Spacecraft Time' and display data as raw DN. Run the query and save the table data

ii. Use the following SQL statement: select SCT_VTCW, Value FROM GPB_L1A..Tmanalog WHERE TMID=<Your TMID> AND SCT_Cycle=<YOURCYCLE> AND SCT_VTCW BETWEEN <YOURSTARTTIME> AND <YOURSTOPTIME> In this statement <YOURCYCLE>, <YOURSTARTTIME>, and <YOURSTOPTIME> are the values input into TCAD.

iii. Compare this output line-by-line to the table data generated in step i. Only the headers should be different.

iv. Timing: Set up a large query (multiple mnemonics, large time range). In a separate Unix window, use 'date' to get the current time, then return to the TCAD

v. window and click 'Display Data'. Return to the command-line window and type 'date' again when the data is displayed.

SU GP-B INSP 31 3/23/04

BASELINE TEST PASS CRITERIA

- Data displayed in table matches data retrieved by SQL select.
- Retrieval time falls within specification, under 1 hour.

REGRESSION TESTS

i. Select the same range of data for the previous version of TCAD and the current version of TCAD.

REGRESSION TEST PASS CRITERIA

- Comparison of the two outputs is identical.

RESULT: PASS FAIL (circle one) _____ initials

10.14 TABLE SETUP: Setup screen for output to table.

☒ (B)

Test Case Verification Number: TABLE SETUP

INTRODUCTION
This section details inspection of output to table. Table output is used in several test cases to verify that the dataset retrieved is what the user requested.

APPROACH
Visual comparision

FEATURES TO BE TESTED
- ❑ Output of raw versus calibrated data.
- ❑ Output of standard date/time versus spacecraft VTCW.

FEATURES NOT TO BE TESTED
- Text editor.

TESTS

i. Select a mnemonic that is known to have calibration coefficients.
ii. Enter Setup Table screen and display data as a Raw DN and as EU/State data.
iii. Display data with Date/Time stamps and Spacecraft Time stamps.

TEST PASS CRITERIA
- In 'Raw DN' mode, a floating point value is displayed for this mnemonic.
- In 'EU/State' mode the calibrated value of this mnemonic is displayed.
- In 'Date/Time' mode, timestamps take the form of YYYY/DOY-HH:MM:SS.n
- In 'Spacecraft Time' mode, timestamps take the form of a long integer.

RESULT: (PASS) FAIL (circle one) _____ initials

10.15 TABLE PRINT: Setup screen for output to table.

❑ (B) ☑ (R)
Test Case Verification Number: TABLE PRINT

INTRODUCTION
Test printing the table output.

APPROACH
Point and click.

FEATURES TO BE TESTED
- ❑ Output to printer
- ❑ Output to file

FEATURES NOT TO BE TESTED

- Data quality
- Table output options.

BASELINE TESTS

i. Load a very small data set.
ii. In TABLE mode, press the 'Print' button.
iii. Print to a printer.
iv. Print to a file.

SU GP-B INSP 31 3/23/04

BASELINE TEST PASS CRITERIA

- When in 'Printer' mode, a dropdown list of available printers is present.
- OK outputs to selected printer. Cancel aborts.
- When in 'To ASCII File' mode, there is an entry box.
- OK outputs to the filename requested (relative to user's home directory). Cancel aborts.

REGRESSION TESTS

v. Load a very small data set.
vi. In TABLE mode, press the 'Print' button.
vii. Print to a printer.

REGRESSION TEST PASS CRITERIA

- OK outputs to selected printer.

RESULT: PASS FAIL (circle one) _____ initials

See Attach 7

10.16 PLOT: TCAD plot display screen

☒ (B) ☐ (R)

Test Case Verification Number: PLOT

INTRODUCTION

This screen graphically displays the selected mnemonics.

APPROACH

Point and click.

FEATURES TO BE TESTED

- ☐ Display of 1K/2K and 32K data.
- ☐ Mouse-over of plot data.

- ❑ Right-click menu access.
- ❑ Zooming in on a plot.
- ❑ Resizing the plot window.

3/23/04
SU GP-B
INSP

~~FEATURES NOT TO BE TESTED~~

FEATURES NOT TO BE TESTED

- Right-click menu display modifiers.

TESTS

- Select a known cycle, time range, and mnemonic for each type of data: analog, discrete, no data, with limits, without limits, with calibrations, and without calibrations. USE 'Restore Configuration' to load file named /home/tdp/regression/display_test.tcad.
- Output data to as a table.
- Output data to screen as a plot.
- Mouse-over the data and compare the snap-to values displayed at the bottom of the screen with the visual appearance of cursor location on the screen and with the values in the generated table for the time index displayed on the screen.
- Resize the window and repeat the previous step to verify that values are the same.
- Left-click once to begin a rubber-band zoom region. Click a second time to mark the far corner of the zoom rectangle and start the zoom. Again, verify the displayed values with the table values.
- Right-click on a plot to activate the right-click menu.
- In the plot setup screen, select all plots to single plot.

TEST PASS CRITERIA

- No error messages are generated as pop-ups or in the TCAD startup window.
- Data points on screen match those of table.
- Mouse-over of data points shows a value matching the value for the time index in the table data.
- Screen resize does not alter accuracy of data points for mouse-over.
- Screen resize does not create video artifacts.
- Screen resize does not make text illegible (Except in cases of becoming too small to read)
- Zooming does not alter accuracy of data points for mouse-over.
- Zooming does not make text illegible.
- Right-click on a plot brings up the right-click menu.
- Legend accurately reflects settings for mnemonics in plot area on a single plot.
- Legend is not displayed in multi-plot mode.
- In all plots to single plot mode, a legend is displayed on the left side of the screen.

- When a plot which contains only 1K/2K data or when only 1K/2K mode is selected, the mouse-over values reflect the 1K/2K values instead of the 32K values.

RESULT: PASS FAIL (circle one) initials

10.17 PLOT SETUP: General configuration of TCAD plots.

☐ (B) ☒ (R)

Test Case Verification Number: PLOT SETUP

INTRODUCTION
The plot setup screen controls the appearance of all plots in the TCAD plot window and in the printed output.

APPROACH
Plot a set of data, make modifications of the plot settings and click the 'Apply' button to compare the previous screenshot with the current screen.

FEATURES TO BE TESTED
- ☐ Plot As
- ☐ Analog Value Display
- ☐ Plot Layout
- ☐ Date Format
- ☐ Show VTCW
- ☐ Limit display
- ☐ Y-Axis scaling

FEATURES NOT TO BE TESTED
- OK, Apply, and Cancel buttons.
- Save/Restore of settings.

~~BASELINE TESTS~~
- ~~Select a known cycle, time range, and mnemonics.~~
- ~~Select Setup from the Display Data screen or Display Options from the Option menu in plot mode.~~
- ~~Individually test each control using the 'Apply' button to redraw the plotted data.~~

SG GF-B INSP 31 3/23/04

BASELINE TEST PASS CRITERIA

- Display screen initiates.
- No error messages are displayed in pop-up dialogs or in TCAD start-up window during operation. 3/23/04
- Display setting changes affect data as expected. IE: limits displayed match limit settings for this mnemonic in database, Display VTCW time displays numeric time instead of date time and time ranges match expectations, and adding a margin displays in the screen.

REGRESSION TESTS
- Select a known cycle, time range, and mnemonics.
- Select Setup from the Display Data screen or Display Options from the Option menu in plot mode.
- Press the APPLY button.

REGRESSION TEST PASS CRITERIA
- Display screen initiates.

RESULT: PASS FAIL (circle one) _____ initials

10.18 PLOT PRINT: Printing TCAD plots

☐ (B) ☒ (R)

Test Case Verification Number: PLOT PRINT

INTRODUCTION
TCAD plots are more complex than tables and have more printing options.

APPROACH
Plot the same set of data to the various output mechanisms and compare.

FEATURES TO BE TESTED
- ☐ Printer output
- ☐ File output
- ☐ Web output

FEATURES NOT TO BE TESTED
- Data quality.

BASELINE TESTS
- Load a dataset and press 'print'.
- Output to each file type.
- Push a plot to the web.
- Print in color.
- Print in black and white.

- Print in Landscape mode.

BASELINE TEST PASS CRITERIA

- Each of the saved file types can be opened and viewed in xv, photoshop or some other graphical application.
- Requested file is pushed to the web (assuming proper user permissions)
- Portrait and landscape mode print accordingly.
- B&W versus color prints print accordingly.
- The printer list dropdown contains the list of printers accessible to the test user.

REGRESSION TESTS

- Print a plot.

REGRESSION TEST PASS CRITERIA

- Plot prints.

RESULT: PASS FAIL (circle one) P initials

see attach 6

10.19 PLOT RTMENU: Plot Customization via right-click menu.

☒ (B)

Test Case Verification Number: PLOT RTMENU

INTRODUCTION

This section details validation or right-click menu items.

APPROACH

Visual inspection.

FEATURES TO BE TESTED

- ☐ Y-Axis Range
- ☐ Function Overplot
- ☐ Color/Symbol settings
- ☐ Print
- ☐ Plot to new Window
- ☐ Zoom out to Max.
- ☐ Clear settings for plot
- ☐ Close all plots
- ☐ Mnemonic Info

FEATURES NOT TO BE TESTED

- None.

TESTS

i. Select a cycle and time range, display data as a table and as a plot. Right-click on a plot to bring up the menu.
ii. Select 'Y-Axis Range' , alter values, and press the 'Update Plot' button. Plot should resize the Y-Axis accordingly. Mouse-over of data should display the same values as the table data.
iii. Select 'Function Overplot'
iv. Select 'Color/Symbol Settings'. Change appearance of the plot.
v. Select 'Zoom out to Max.'
vi. Select 'Print'
vii. Select 'Mnemonic Info'
viii. Select 'Plot to new Window'
ix. Select 'Clear all settings'
x. Select 'Close all plots'

TEST PASS CRITERIA

Requested modifications displayed as expected, plot(s) output to printer, mnemonic info given matches that shown on TMinfo screen.

RESULT: PASS FAIL (circle one) _____ initials

10.20 USER PREFS: User preferences screen

☐ (B) ☒ (R)

Test Case Verification Number: USER PREFS

INTRODUCTION

This test case outlines procedures for testing the user preferences window and verifying its effect on related windows.

APPROACH

Point and click, visual inspection.

FEATURES TO BE TESTED

- ☐ Preservation of plot settings.
- ☐ Default mode for printing.

FEATURES NOT TO BE TESTED

- Setting plot options.

BASLINE TESTS

i. Load a data set to plot.

SU GP INSP 31 3/23/04

ii. Select 'Preferences' from the 'Option' menu.
iii. Check 'Preserve user preferences' if it is unchecked.
iv. Display the plot and make alterations to its appearance using options from the right-click menu (such as scaling and changing the plot color)
v. Close the plot window and reopen it.
vi. Select 'Preferences' from the 'Option' menu again and uncheck 'Preserve user preferences'.
vii. Close the plot window and reopen it a second time.
viii. Toggle the 'Print in Color' checkbox in the 'User Preferences' window, press 'Okay' and click the 'print' button on the Display Data screen.

BASELINE TEST PASS CRITERIA SIL GP-B INSP 31 3/23/04

- When 'preserve user preferences' is selected, right-menu optional modifiers to the plot are preserved between redisplays of the data. When it is unchecked, the default settings are used.
- The 'Print in Color' on the 'Plot Print' screen is defaulted to match the setting in the preferences.

REGRESSION TESTS

i. Turn on 'preserve user preferences' and make changes to plot color from the right-click menu and reload the plot.

REGRESSION TEST PASS CRITERIA

- Color change is preserved.

RESULT: PASS FAIL (circle one) P initials

see MCR 314

10.21 MNE SEARCH: Select list of mnemonics matching wildcard

☐ (B) ☒ (R)

Test Case Verification Number: MNE SEARCH

INTRODUCTION

The Mnemonics Found screen can be accessed both from Display Data and from the TM Info screen. Behavior is slightly different depending on the calling application.

APPROACH

Point and click, visual inspection.

FEATURES TO BE TESTED

- ☐ Select of mnemonic(s)
- ☐ Entry from different applications.

FEATURES NOT TO BE TESTED

- Application of returned mnemonic(s).

- Activation of mnemonic list by calling routines.

BASLINE TESTS

i. Open The 'TM Info' window and type in a valid wildcard pattern to activate the Mnemonics Found screen.
ii. Hold down the SHIFT key and attempt to select multiple mnemonics.
iii. Select a mnemonic and press 'OK'.
iv. Re-enter the selection screen and press 'CANCEL'.
v. Open the 'Display Data' screen and enter a pattern that will match 5 or more mnemonics to activate the 'Mnemonics Found' screen.
vi. Select 1 mnemonic and press 'OK'.
vii. Re-enter the screen, and select 2 or more mnemonics (by holding down SHIFT or CNTRL) and press 'OK'
viii. Re-enter the screen and press 'CANCEL'

SU-GP-B INSP 31 3/23/04

BASELINE TEST PASS CRITERIA

- Multiple-mnemonic selection is disallowed from the TMInfo screen.
- The selected mnemonic in Tminfo is returned and the data for this mnemonic is displayed.
- The single selected Mnemonic is returned to the Display Data screen.
- The multiple selected Mnemonics are returned to the Display Data screen.
- The CANCEL button returns to the parent window without making any changes to the parent screen.

REGRESSION TESTS

i. Enter the mnemonic selection screen from either TMInfo or Display Data, select a mnemonic and press OK.

REGRESSION TEST PASS CRITERIA

- The selected mnemonic is returned.

RESULT: PASS FAIL (circle one) _____ initials

10.22 EVENT GET: Tcad GUI wrapper to Event Get application.

☐ (B)

Test Case Verification Number: EVENT GET

INTRODUCTION
This section details testing of IDL wrapper to Perl Event-Get program.

APPROACH
Visual inspection.

FEATURES TO BE TESTED

- ☐ Activation of the Event-get Perl script by the GUI.

FEATURES NOT TO BE TESTED

- Data integrity of the event get program. (This is handled in P0832)

TESTS

- Select EventGet from the Analyze menu.
- Select cycle and time range.
- Click Okay

TEST PASS CRITERIA

- By visual inspection of the data retrieved by the event get program, verify that the dates and cycle range entered by the user were the ones passed to the event get application.

RESULT: PASS FAIL (circle one) _____ initials

10.23 MRO GET: Tcad GUI wrapper to MRO Get application.
☒ (B)

Test Case Verification Number: MRO GET

INTRODUCTION
This section details testing of IDL wrapper to Perl MRO-Get program.

APPROACH
Visual inspection.

FEATURES TO BE TESTED

- ☐ Activation of the MRO-get Perl script by the GUI.

FEATURES NOT TO BE TESTED

- Data integrity of the event get program.

TESTS

- Select MROGet from the Analyze menu.
- Select cycle and time range.
- Click Okay

TEST PASS CRITERIA

- By visual inspection of the data retrieved by the MRO get program, verify that the dates and cycle range entered by the user were the ones passed to the MRO get application.

RESULT: PASS FAIL (circle one) _____ initials

10.24 DBRO GET: Tcad GUI wrapper to DBRO Get application.

☒ (B)

Test Case Verification Number: DBRO GET

INTRODUCTION
This section details testing of IDL wrapper to Perl DBRO-Get program.

APPROACH
Visual inspection, independent verification via a more detailed discussion in S0638

FEATURES TO BE TESTED

- ❑ Activation of the DBROget Perl script by the GUI.
- ❑ Ability to select desired Application ID from list of SW Applications or by typing one into the Application ID text box.

FEATURES NOT TO BE TESTED

- Data integrity of the DBROget program. (This is handled in S0638)

TESTS

- Select tcad_dbroget from the Analyze menu.
- Select cycle and time range.
- Enter an Application ID or select one using the "Select Application by Name" button.
- Click Okay

TEST PASS CRITERIA

- By visual inspection of the data retrieved by the DBROget program, verify that the application ID, dates and cycle range entered by the user were the ones passed to the DBRO get application.

RESULT: PASS FAIL (circle one) _____ initials

10.25 SNAPSHOT: TCAD Snapshot Display tool.

☒ (B)

Test Case Verification Number: SNAPSHOT

INTRODUCTION
This section details testing of IDL Snapshot display program.

APPROACH
Visual inspection.

FEATURES TO BE TESTED

- ❑ Launch of snapshot routine with proper parameters.
- ❑ Selecting Subsystem changes snapshot options
- ❑ Typing into Snapshot ID returns expected data
- ❑ Choosing 'Send all to printer' will print data
- ❑ Entering data into 'Show Snapshots' boxes returns only desired number of snapshots.
- ❑ Selecting Snapshot sub-type options returns expected data
- ❑ Right-click menu options display
- ❑ Right-click "print" option prints plots
- ❑ Right-click "Close all plots" option closes all plots

FEATURES NOT TO BE TESTED

- Data matrix generated.
- Locating snapshot data.

TESTS

- Select snapshots from the Analyze menu.
- Select cycle and time range.
- Select Subsystem
- Select 'Show Snapshots' as directed by message windows, as needed
- Select sub-type options in snapshot information box such as "Snapshot Type"
- Check the 'Send all to Printer' box
- Click Okay
- Use right-click menu to print a plot
- Use right-click menu to close all plot windows

TEST PASS CRITERIA

- By visual inspection of the data retrieved by snapshot, verify that the dates and cycle range entered by the user were the ones requested.
- By visual inspection verify that subsystem, type, and number of snapshots sent agree with those entered.
- Paper print arrived at printer
- Right-click menu options behave as expected

RESULT: PASS FAIL (circle one) P initials

SEE MCR 312

10.26 FMTRPT: Tcad GUI wrapper to the L1 and L0 Fmt Report applications.

☒ (B)

Test Case Verification Number: FMTRPT

INTRODUCTION
This section details testing of IDL L0 and L1 Format Report programs.

APPROACH
Visual inspection.

FEATURES TO BE TESTED
- ☐ Activation of the L0 Format Report by the GUI.
- ☐ Activation of the L1 Format Report by the GUI.

FEATURES NOT TO BE TESTED
- Data integrity of the Format Report programs.

TESTS
- Select l0_fmtreport from the Analyze menu.
- Select cycle and time range.
- Click Okay
- Once report is displayed to the screen, proceed with next step.
- Select l1_fmtreport from the Analyze menu.
- Select cycle and time range.
- Click Okay

TEST PASS CRITERIA
- By visual inspection of the data retrieved by the two Format Report programs, verify that the dates and cycle range entered by the user were the ones passed to the Format Report applications.

RESULT: PASS FAIL (circle one) _____ initials

10.27 SSGET: Tcad GUI wrapper to Snapshot Get application.

☒ (B)

Test Case Verification Number: SSGET

INTRODUCTION
This section details testing of IDL wrapper to Perl Snapshot-Get program.

APPROACH
Visual inspection.

FEATURES TO BE TESTED

- ❑ Activation of the Snapshot-get Perl script by the GUI.

FEATURES NOT TO BE TESTED

- Data integrity of the event get program.

TESTS

- Select SSGet from the Analyze menu.
- Select cycle and time range.
- Click Okay

TEST PASS CRITERIA

- By visual inspection of the data retrieved by the Snapshot get program, verify that the dates and cycle range entered by the user were the ones passed to the Snapshot get application.

RESULT: PASS FAIL (circle one) _____ initials

10.28 AUTOPLOT: Command-line plotting tool.

❑ (B) ☒ (R)

Test Case Verification Number: SSGET

INTRODUCTION
The auto_plot tool is a stand-alone script that generates a dummy structure set for the TCAD plotting routines and passes in a series of TCAD save files with an alternate cycle and time range specified. Its purpose is to generate plots identical to those in TCAD but from a command-line scriptable interface.

APPROACH
Entry of command-line parameters, visual inspection of printed plots.

FEATURES TO BE TESTED

- ❑ Use of plot settings and user preferences stored in the saved TCAD file.
- ❑ Generation of plots
- ❑ Use of specific printer

FEATURES NOT TO BE TESTED

- Save file generation
- Formatting of time strings.

- Printer settings

BASELINE TESTS

- From the command-line run auto_plot with the following syntax:
  Auto_plot <cycle> <starttime> <endtime> <tcadfile> <-Pprintername>
  (Consult the TCAD User's Guide SO717 for more information on these parameters)

- From the command-line run auto_plot with the following syntax:
  Auto_plot <cycle> <starttime> <endtime> <tcadfile> -JPG
  (Consult the TCAD User's Guide SO717 for more information on these parameters)

- From the command-line run auto_plot with the following syntax:
  Auto_plot <cycle> <starttime> <endtime> <tcadfile> -JPG -WEB
  (Consult the TCAD User's Guide SO717 for more information on these parameters)

SU GP-B INSP 31 3/23/04

BASELINE TEST PASS CRITERIA

- First plot is printed to printer specified with the –P parameter.
- Second Plot is sent to a file in the user's home directory named, <tcadfile.JPG>. Validate JPG file using unix tool 'xv'
- Third plot is sent to the https://gpbops.stanford.edu/reports/tcad directory and named <tcadfile.JPG>
- The plot visually matches the plot on the screen when the same .tcad file is loaded and run to display output.
- User preferences such as plot symbols and color/B&W printing correctly appear in the printed plot.

REGRESSION TESTS

- From the command-line run auto_plot with the following syntax:
  Auto_plot <cycle> <starttime> <endtime> <tcadfile>
  (Consult the TCAD User's Guide SO717 for more information on these parameters)

REGRESSION TEST PASS CRITERIA

- Plot is printed to printer.

RESULT: PASS FAIL (circle one) _____ initials

See attachment 8

Successful Completion of P0950:

RUN BY (signature) Samantha Patterson Date: 3-23-04

Print Name: Samantha Patterson

ALL TESTING PASSED (CIRCLE ONE): YES – PASSED NO – MCR FILED #312 #314

QA Review of process Dorrene Ray Date: 3/23/04

11 GLOSSARY

This section contains an alphabetic list and definitions of all acronyms used in the document, all proper nouns, and any words used in a non-standard way.

Word	Detail
CSCI	Computer Software Configuration Item
LASP	Laboratory for Atmospheric and Space Physics, University of Colorado
moc-server	Host name of the SUN computer that is the primary server for the MOC.
science server	Host name of the SUN computer which is the primary server for science LAN
SAFS	Standard Autonomous File Server (GSFC facility)
TCAD	Telemetry Checking, Analysis, and Display
TDP	Telemetry Data Processing
Startup window	The window containing the Unix command line from which TCAD was started

Attach 1 tcad/* versions

version of "analyze":	Unknown
version of "auto_plot":	Unknown
version of "auto_plot.pro":	2.2
version of "auto_plot.sh":	Unknown
version of "auto_plot_main.pro":	2.3
version of "calculate_fft.pro":	2.2
version of "dt_timeaxis.pro":	2.2
version of "idl_startup.pro":	2.2
version of "L0_fmtrpt.pro":	2.3
version of "L1_fmtrpt.pro":	2.5
version of "plot_fmt_select.pro":	2.2
version of "retrieve_eng.pro":	2.2
version of "SCCS":	Unknown
version of "tcad":	Unknown
version of "tcad.pro":	2.2
version of "tcad_clr_index.pro":	2.2
version of "tcad_dbroget__app_list.pro":	2.2
version of "tcad_display.pro":	2.2
version of "tcad_display__restore.pro":	2.2
version of "tcad_display__save.pro":	2.2
version of "tcad_display__search_list.pro":	2.2
version of "tcad_display__timetype.pro":	2.2
version of "tcad_initial_printer.pro":	2.2
version of "tcad_main.pro":	2.2
version of "tcad_panel_components.pro":	2.3
version of "tcad_plot.pro":	2.8
version of "tcad_plot__col.pro":	2.2
version of "tcad_plot__function.pro":	2.2
version of "tcad_plot__prepare_print.pro":	2.2
version of "tcad_plot__rtmenu.pro":	2.3
version of "tcad_plot__setup.pro":	2.2
version of "tcad_plot__yscale.pro":	2.2
version of "tcad_preferences.pro":	2.2
version of "tcad_structs.pro":	2.2
version of "tcad_table.pro":	2.4
version of "tcad_table__prepare_print.pro":	2.2
version of "tcad_table__setup.pro":	2.2
version of "tcad_time_string.pro":	2.2
version of "tcad_tminfo.pro":	2.3
version of "tcad_values.pro":	2.15
version of "test.pro":	2.2
version of "timeaxis2.pro":	2.2

Attach 2 — Analyze/* versions

version of "DBROget.pl":	Unknown
version of "EVENTget.pl":	Unknown
version of "FMTreport.pl":	Unknown
version of "l0_fmtreport.pro":	2.4
version of "l1_fmtreport.pro":	2.5
version of "MROget.pl":	Unknown
version of "ORBITget.pl":	Unknown
version of "SCCS":	Unknown
version of "snapshot.pro":	Unknown
version of "SSget.pl":	Unknown
version of "tcad_dbroget.pro":	2.2
version of "tcad_eventget.pro":	2.2
version of "tcad_mroget.pro":	2.2
version of "tcad_ssget.pro":	2.2
version of "time_converter.pro":	2.2
version of "TIMINGget.pl":	Unknown
version of "TLMreport.pl":	Unknown

Attach 3 db versions

```
SCCS/s.gpb_db.pro:        D 2.3    04/03/14 22:36:05 pmcgown        30 29    00080/00
SCCS/s.gpb_time.pro:      D 2.1    04/02/06 05:16:24 ron    10 9    00000/00000/0024
SCCS/s.db_routines.so:    D 2.3    04/03/18 02:53:28 local 12 11    06319/04983/0082
SCCS/s.generic_time.pro:           D 2.1    04/02/06 05:16:24 ron    10 9    00000/00
SCCS/s.time_lib.pro:      D 2.1    04/02/06 05:16:29 ron    28 27    00000/00000/0065
```

Attach 4

```
2> select * from TMnames where TMID=74
3> go
 TMID      MSSID  Initial_MSSID Subsystem        Mnemonic
         Description
         Datatype Start_Bit Length
 -------- ------ ------------- ---------------- ----------------
         ------------------------------------------------------------------------
         -------- --------- ------
       74  13344          12864 ACE              BA_Ace_Enbld
         Ace processing enabled status
         RDU                0      8

(1 row affected)
```

SU GP-B
INSP 31

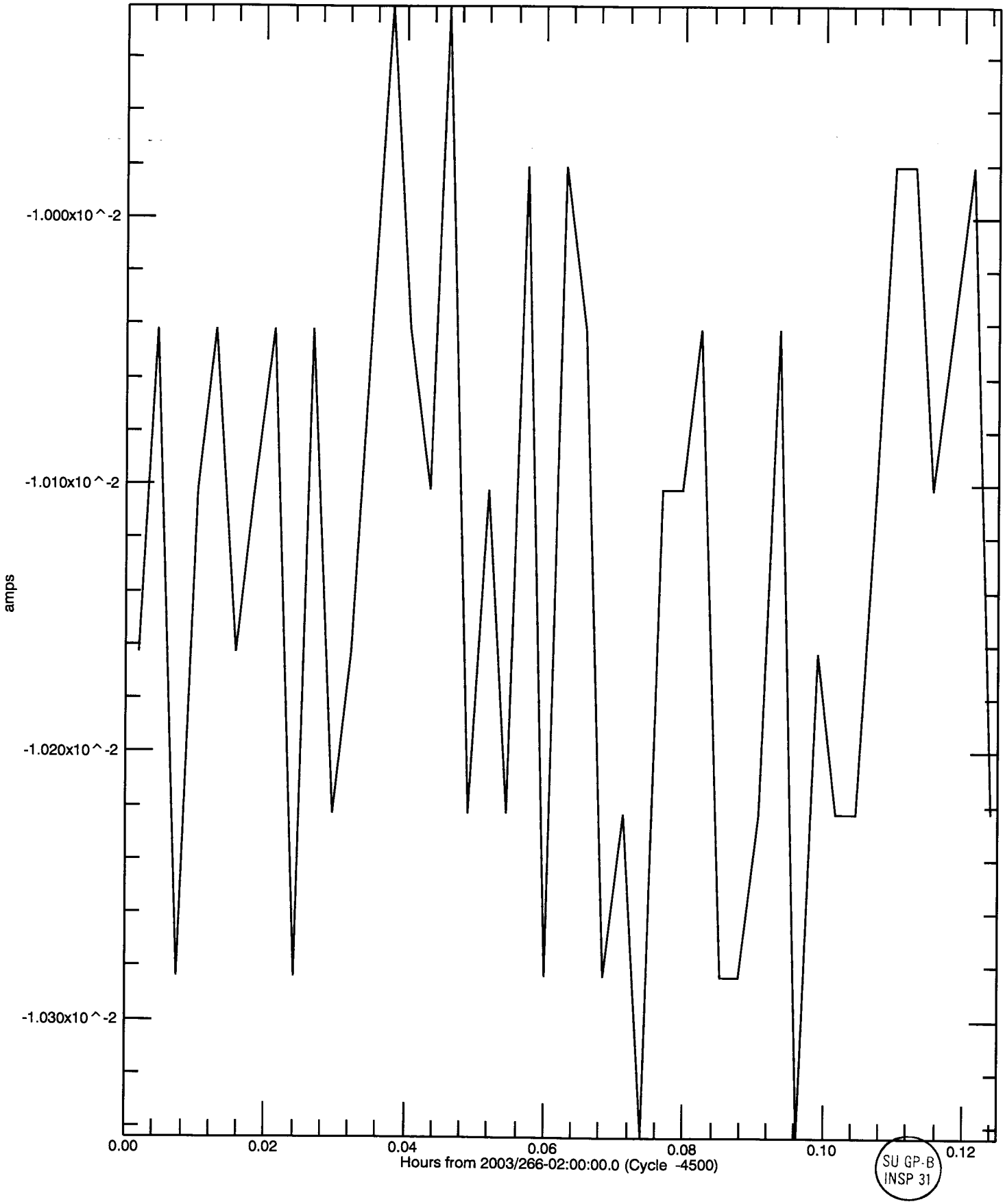
Attach 5

```
GPB_TCAD_CONFIGURATION
        1   -4500
2003/266-02
+1
OPTIONS
        0
        0
       0.00000
        1        0        0
            1           12
        1
        0
        1
            1
CE_HPMH_I_ah01d ECU   0 255 255 -1 -1        0.0000000        0.0000000
```

SU GP-F
INSP 31

Attach 6

ECU - CE_HPMH_I_ah01d



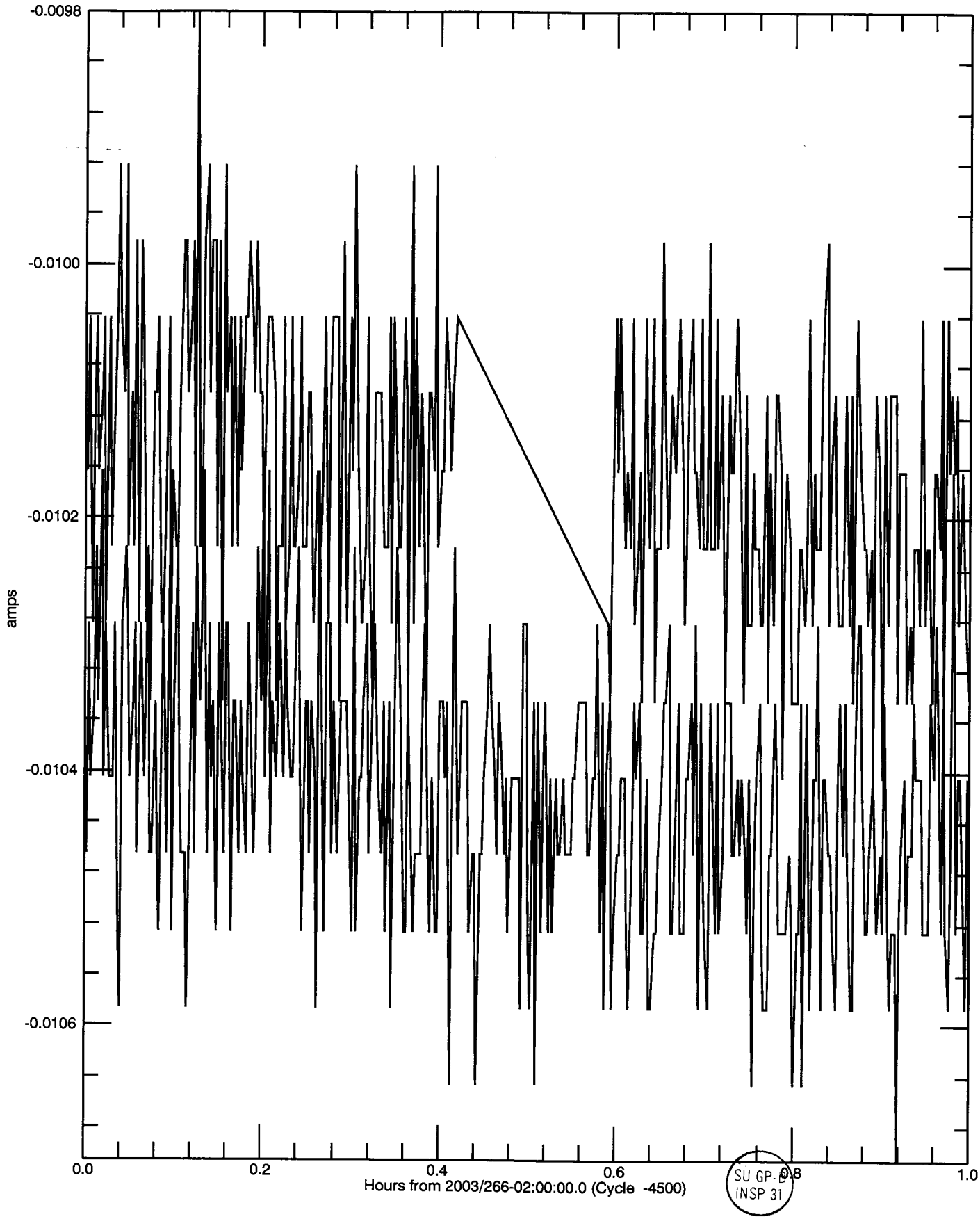
SU GP-B
INSP 31

Attach 7

Cycle -4500	CE_HPMH_I_ah01d
2003/266-02:00:06.5	-0.01016272
2003/266-02:00:16.5	-0.01004174
2003/266-02:00:26.5	-0.01028371
2003/266-02:00:36.5	-0.01010223
2003/266-02:00:46.5	-0.01004174
2003/266-02:00:56.5	-0.01016272
2003/266-02:01:06.5	-0.01010223
2003/266-02:01:16.5	-0.01004174
2003/266-02:01:26.5	-0.01028371
2003/266-02:01:36.5	-0.01004174
2003/266-02:01:46.5	-0.01022322
2003/266-02:01:56.5	-0.01016272
2003/266-02:02:06.5	-0.01004174
2003/266-02:02:16.5	-0.00992075
2003/266-02:02:26.5	-0.01004174
2003/266-02:02:36.5	-0.01010223
2003/266-02:02:46.5	-0.00992075
2003/266-02:02:56.5	-0.01022322
2003/266-02:03:06.5	-0.01010223
2003/266-02:03:16.5	-0.01022322
2003/266-02:03:26.5	-0.00998125
2003/266-02:03:36.5	-0.01028371
2003/266-02:03:46.5	-0.00998125
2003/266-02:03:56.5	-0.01004174
2003/266-02:04:06.5	-0.01028371
2003/266-02:04:16.5	-0.01022322
2003/266-02:04:26.5	-0.01034420
2003/266-02:04:36.5	-0.01010223
2003/266-02:04:46.5	-0.01010223
2003/266-02:04:56.5	-0.01004174
2003/266-02:05:06.5	-0.01028371
2003/266-02:05:16.5	-0.01028371
2003/266-02:05:26.5	-0.01022322
2003/266-02:05:36.5	-0.01004174
2003/266-02:05:46.5	-0.01034420
2003/266-02:05:56.5	-0.01016272
2003/266-02:06:06.5	-0.01022322
2003/266-02:06:16.5	-0.01022322
2003/266-02:06:26.5	-0.01010223
2003/266-02:06:36.5	-0.00998125
2003/266-02:06:46.5	-0.00998125
2003/266-02:06:56.5	-0.01010223
2003/266-02:07:06.5	-0.01004174
2003/266-02:07:16.5	-0.00998125

Attach 8

ECU - CE_HPMH_I_ah01d



SU GP-B
INSP 31

MCR REPORT No. *312*

Change Request: ***TCAD snapshot plotting in TRE section***

D Log No.

Change Type. ***Change Request***

Date of Event: ***03/16/04***

Subystem: ***TDP/TCAD***

Owner: ***K Stahl***

Reported By: ***J Goebel***

PCB Number: ***Not Required***

SW Level:

Action Due: / /

Status: ***Under Investigation***

Planned Code Release:

Actual Code Release:

Description:

1) Plots sent to the printer output large tables of numbers instead of plot.

2) When the plot windows are closed and then new plots ordered without restarting TCAD, there is a halt in the program, requiring restarting of TCAD to clear.

Recommendation:

Repair plot options if reasonable in time allotted.
If required, turn off printing options all together as a quick-fix.

NASA Notification:

Documents Affected:

Verification Test Cases:

Attempt to plot TRE snapshots and send to printer. If user receives a paper plot, code change has been successful. Close plot windows. Use menu to call up more snapshot plots. If application brings up new plot windows successfully, the code change is successful.

Status Notes: 3-23-04: Cannot create tmp file to print.

Signoffs:

MCR REPORT No. *314*

Change Request: ***TCAD User Preserve Color multi-window issues***

D Log No.

Change Type. ***Change Request***

Date of Event: ***03/23/04***

Subystem: ***TDP/TCAD***

Owner: ***S Patterson***

Reported By: ***R Sharbaugh***

PCB Number: ***Not Required***

SW Level:

Action Due: / /

Status:

Planned Code Release:

Actual Code Release:

Description:

testing results with lasp-2.3 suggest that more work is advisable for preserving user preferences on graphs.

Recommendation:

NASA Notification:

Documents Affected:

VDD

Verification Test Cases:

tbd

Status Notes:

Signoffs:

R Brumley _____ _____

K Burlingham _____ _____

R Nevitt _____ _____

R Sharbaugh _____ _____

M Smith _____ _____

MCR REPORT No. *229*

Change Request: ***TCAD: Display Formats for mnemonics somewhere***

D Log No.

Change Type. ***Change Request***

Date of Event: ***10/01/03***

Subystem: ***TDP/TCAD***

Owner: ***J Spencer***

Reported By: ***J Spencer***

PCB Number: ***636***

SW Level:

Action Due: / /

Status: ***In Process***

Planned Code Release: ***2.1***

Actual Code Release:

Description:

Users want to find the formats for a given telemetry item easily and quickly. Add to TCAD display.

Recommendation:

3/16/2004: Approved for LASP 2.3 release by MCCB.
10/1/03 Approved by committee for LASP 2.1
1/14/04 Approved by committee for LASP 2.2

NASA Notification:

Documents Affected:

S0613, S0503

Verification Test Cases:

P0950 section 10.2.

Status Notes:

11/26/03 This did not make 2.1 / 3/23/04 Tested per P0950 Rev E and accepted. DM Pr

SU GP-B INSP 31

Signoffs:

R Brumley _____ _____
K Burlingham _____ _____
R Sharbaugh _____ _____
P Shestople _____ _____
M Smith _____ _____
J Spencer _____ _____

MCR REPORT No. *301*

Change Request: ***TCAD l1fmtrpt - add 1k/2k reporting***

D Log No.

Change Type. ***Change Request***

Date of Event: ***03/02/04***

Subystem: ***TDP/TCAD***

Owner: ***P McGown***

Reported By: ***J Wade***

PCB Number:

SW Level:

Action Due: / /

Status: ***PCB Required***

Planned Code Release:

Actual Code Release:

Description:

l1fmtrpt only analyze the 32k telem stream (GPB_L1..TManalog) for the mnemonic SF_MSS_ID. Users would also like similar reporting from GPB_L1..SNanalog.

Recommendation:

3/16/2004: Approved for LASP 2.2 release by MCCB.

NASA Notification:

Documents Affected:

VDD

Verification Test Cases:

Use STP (TCAD plotting vs TCAD reporting). P0950 section 10.26.

Status Notes:

3/23/04 Tested per P0950 Rev E and accepted DM Pr 3/23/04 SU GP-B INSP 31

Signoffs:

R Brumley _____ _____

K Burlingham _____ _____

J Spencer _____ _____

MCR REPORT No. *306*

Change Request: ***TCAD L0 Format report speed and efficiency***

D Log No.

Change Type. ***Change Request***

Date of Event: ***03/16/04***

Subystem: ***TDP/TCAD***

Owner: ***P McGown***

Reported By: ***J Spencer***

PCB Number:

SW Level: ***Level 3***

Action Due: / /

Status: ***PCB Required***

Planned Code Release:

Actual Code Release:

Description:

Streamline/Improve the query time for Format Report from L0..Packets for L0_fmtrpt (TCAD).

Recommendation:

3/16/2004: Approved for LASP 2.2 release by MCCB.

Check db_routines to see how fetch from L0..Formats is actually accomplished. Streamline the select or the cursor fetch in the core programming.

NASA Notification:

Documents Affected:

None except S/W VDD, S0503

Verification Test Cases:

P0950 10.26

Status Notes:

3/23/04 Tested per P0950, RwE and accepted DMRy

SU GP-B INSP 31

Signoffs:

MCR REPORT No. *307*

Change Request: ***TCAD improve efficiency for Derived Monitors***

D Log No.

Change Type. ***Change Request***

Date of Event: ***03/16/04***

Subystem: ***TDP/TCAD***

Owner: ***P McGown***

Reported By: ***J Spencer***

PCB Number:

SW Level: ***Level 3***

Action Due: / /

Status: ***PCB Required***

Planned Code Release:

Actual Code Release:

Description:

Streamline/Modify/Improve the TCAD query for the DERIVED monitors SF_Format_ID and SF_Frame_Count from GPB_L0..Packets table.

Recommendation:

3/16/2004: Approved for LASP 2.2 release by MCCB.

Investigate and improve efficiency. Users are concerned about the speed of retrieval for these monitors and have requested improvement.

NASA Notification:

Documents Affected:

None except S0503, S/W VDD

Verification Test Cases:

Test speed in prior revision of lasp by selecting to plot monitors listed in Description, then select the same monitors for plotting in the new version of software. If speed improves, the test is successful.

Status Notes:

3/23/04 Tested and accepted DM Ross

SU GP-B INSP 31

Signoffs:

MCR REPORT No. *309*

Change Request: ***TCAD plotting a 1K/2K discrete monitor with no data creates error***

D Log No.	***2004_068_210128***			Change Type.	***Discrepancy***
Date of Event:	***03/16/04***			Subystem:	***TDP/TCAD***
Owner:	***S Patterson***	Reported By:	***J Spencer***	PCB Number:	***Not Required***
SW Level:	***Level 4***	Action Due:	/ /	Status:	***In Process***
Planned Code Release:	***2.3***	Actual Code Release:			

Description:

Plotting a 1K/2K discrete monitor with no data found will cause an error.

Recommendation:

Update program TCAD_VALUES to fix bug.

NASA Notification:

Documents Affected:

S0503

Verification Test Cases:

Select a discrete mnemonic with the checkbox for 1k/2k data checked, plot a time range with no data for this mnemonic. See if program halts per DLOG description. If program does not halt, code has passed testing.

Status Notes:

3/23/04 Tested and accepted JM Rzy SU GP-B INSP 31

Code complete.

Signoffs:

AS-RUN
P0981 Rev A
August 17, 2003
ATTACH TO S0973B

W. W. Hansen Experimental Physics Laboratory

STANFORD UNIVERSITY

STANFORD, CALIFORNIA 94305-4085

Gravity Probe B Relativity Mission

Automated Import Scripts Software Test Document

P0981
Revision A

August 17, 2003

Approvals

NAME	SIGNATURE	DATE
Samantha Patterson *Software Engineer*		
Rodney Torii *Data Processing Lead*		
Ron Sharbaugh *S/W Manager*		
Marcie Smith *MOC Project Manager*	ECO # 1452	
Kelly Burlingham *Software Quality Engineer*		9/17/03

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

Table of Contents:

History

REV	DATE	AUTHOR	COMMENTS
-	25 June2003	sap	initial version
A	16 Sept 2003	sap	Removed duplicate history table from cover sheet; removed pre-typed RCS version numbers; added test case CONTROL1 to accommodate MCR 209.

1 SCOPE

Testing of the moc-server and science server cronjobs for automated imports.

2 OPERATIONAL PERSONNEL

Samantha Patterson
Qualified QA Rep: Kelly Burlingham

3 QUALITY ASSURANCE PROVISIONS

Quality Assurance must be given 24 hour notification before this test is run; presence is at their discretion.

QA Notified Date & Time: 3/19/04 10:00AM By: Ron Shaubaugh QA Initials: DMR

4 TEST ENVIRONMENT

Software Configurations	Version Number
TDP	2.3
Solaris	2.8

5 TEST CASES AND FILE VERSION MATRIX

SU GP-B INSP 31

Files on Moc-Server

File	RCS Ver	Test Name	Test Section
~~/home/safs/apps/moc_auto.cron~~		~~MOC1~~	~~Section 7.1~~
/home/safs/apps/science_auto.cron		SCI1	Section 7.2
/home/safs/apps/cron_control		CONTROL1	Section 7.3

DMR 3/24/04

Files on science

SU GP-B INSP 31

File	RCS Ver	Test Name	Test Section
/home/tdp/apps/sci_auto.cron		~~SCI1~~	Section 7.1
/home/tdp/apps/cron_control		CONTROL1	~~Section 7.3~~

DMR 3/24/04

The following sections describes how to test

Cron Version asp 2.3

Start Date & Time: 18:06 3-24-04

Executed By: Samantha Patterson Signature: Samantha Patterson

Witnessed By: Dorrene Ross Signature: Dorrene Ross

6 APPLICABLE DOCUMENTS

Document No.	Document	ALIAS.
S0401	Stanford Post-Processing Operations for Science Mission Data	
P0904	Data Processing Through a Spacecraft Clock Reset	
S0908	Automated Import Software Requirements Document	
S0909	Automated Import Scripts Software Design Document	
S0912	Automated Import Scripts Version Design Document	
S0913	Automated Import Scripts Software Verification Report	

7 SOFTWARE VERIFICATION PLAN

The following table lists the CSCI software objects that comprise TDP and TCAD, and for each CSCI lists the test cases which must pass to verify it:

SOFTWARE OBJECT	TEST CASES
~~Moc_auto.cron~~	~~MOC1: throughput~~ DMR 3/24/04
Science_auto.cron	SCI1: auto_import initializationation
cron_control	CONTROL1: UI to simplify interaction with cron.

7.1 MOC1: Throughput

☐ (B) ☐ (R)

Test Case Verification Number: MOC1

SU GP-B INSP 31 DMR 3/24/04

INTRODUCTION

This test case verifies the file handling of the cron job on moc-server and may be run independently of any other step in this process.

APPROACH

Copy files into the /home/safs/data/VC0 and /home/safs/data/V12 directory and wait.

FEATURES TO BE TESTED

- ☐ Copying data from moc-server to science.
- ☐ Moving data from VC0 and V12 directories to /home/safs/data/processed directory.

FEATURES NOT TO BE TESTED

- SAFS network data transmission.
- Science_auto.cron functionality.

BASELINE TESTS

i. Copy bin file(s) into the moc-server /home/safs/vc0 and /home/safs/v12 directories and run moc_auto.cron from the command line.

ii. Repeat first test (using a different file name), but this time, start moc_auto.cron while the copy is in process (NOTE: this may require using a slower transfer method such as SCPing the file from podg or a directory on science)

iii. Use 'crontab –e' to insert a cronjob for this script for user SAFS. (See man pages on cron and crontab for crontab format) and repeat step 1 (again using a different file name).

SU GP-B INSP 31 3/24/04

BASELINE TEST PASS CRITERIA

- The file(s) used in step one appeared in the matching subdirectory of the /home/tdp directory on science.
- The unix tools 'sum' and ls –l indicate identical file size and checksum for the file on moc-server and its duplicate on science.
- The file on moc-server was moved to /home/safs/processed.
- The file from the second test is NOT transfered to science when the moc_auto.cron script has finished. (NOTE: the moc_auto.cron script must complete before the file transfer does to validate this step)
- The file(s) used in step 3 are copied to science in approximately the same amount of time (from the start time of the cron job) as they were when run manually.

REGRESSION TESTS

i. Clear any files from the vc0, v12, and framex directories and run moc_auto from the command-line.

REGRESSION TEST PASS CRITERIA

- Status messages indicate all directories were checked but no data was processed.

RESULT: PASS FAIL (circle one) _____ initials

7.2 SCI1: auto_import.exp initialization

☒ (B)

Test Case Verification Number: SCI1

INTRODUCTION
This test verifies assignment of a cycle number and initialization of the auto_import routine for ingestion of data into the level 0 and level 1 databases.

APPROACH
Manual and automated launch of script after placing data in the incoming directory.

FEATURES TO BE TESTED

- ❑ Selection of cycle number.
- ❑ Creation of auto_import.exp command file.
- ❑ Creation of log of auto_import run.
- ❑ Initialization of auto_import.exp
- ❑ ~~Run from science-server.~~
- ❑ Run from moc-server
- ❑ Push import logs to Relgyro for SAFS confirmation.

SU GP-B INSP 31 DM Pm 3/24/04

FEATURES NOT TO BE TESTED

- Transmission of data from moc-server to science.
- import of data into L0 and L1 databases.
- Data confirmation message to SAFS.
- Update of IMPORT_LOG.TXT file.
- Detection of clock-resets.

TEST
Due to the nature of this test, it is strongly recommended that all testing of this script be issued with the command-line argument, '-CHECKONLY' to prevent accidental or erroneous insertion of data into the Sybase databases. The only effect this has on the process is to disable BCP of data into the databases. BCP validation is handled in S0613, the test document for TDP/TCAD:

i. Create the file /home/tdp/processed/IMPORT_LOG.TXT as defined in section 3.1 of this document.
ii. Copy a known good bin file into the /home/tdp/ vc0, /home/tdp/v12, or /home/tdp/framex directory.
iii. Run science_auto.cron manually with the –checkonly flag.
iv. Repeat the second step and rerun step 3 while the copy is in process (once again, this may require a slower copy method such as SCP)
v. Add an entry to the crontab for user tdp on ~~science~~ moc (see man pages for cron and crontab for crontab format) and repeat step 2.
vi. ~~Switch to running this application from moc-server instead of science and repeat all steps prior to this one.~~

SU GP-B INSP 31 DM Pm 3/24/04

TEST PASS CRITERIA

- The file(s) copied into the incoming directory in step 2 were moved to the /home/tdp/processed directory.
- A file named cmd.MMDDYY_HH:mm:SS (where MM is month, DD is day of month, YY is year, HH is hour, mm is minute and SS is second) was created in /home/tdp/processed/commands. This file contains a line for each file put in the

incoming directory and the cycle number specified n the /home/tdp/processed/IMPORT_LOG.TXT.
- A file with the same date and time stamp in the /home/tdp/processed/logs directory exists and contains a log of the auto_import process.
- The file specified in test 4 did not generate a command or log file while the copy of this file was in progress.
- Login to https://gpbops.stanford.edu/dp and confirm update of the pushed logfile.
- PASS conditions are met on both Science and Moc-Server systems.

RESULT: PASS FAIL (circle one) _____ initials

7.3 CONTROL1: Cronjob control program.

☒(B)
Test Case Verification Number: CONTROL1

INTRODUCTION
This test-case handles interaction of the cron_control script with the science_auto.cron program and the crontab.

APPROACH
Command-line testing

FEATURES TO BE TESTED
- ❑ Starting, stopping, resetting, and aborting cron processes.

FEATURES NOT TO BE TESTED
- Running cron processes.

BASELINE TESTS
i. Type 'cron_control'
ii. Type 'cron_control status' then use ps –aef | grep cron; ps –aef | grep auto; and tail the IMPORT_LOG.TXT file.
iii. Type 'cron_control start' then use 'cron_control status' to check the results.
iv. Type 'cron_control stop' then use 'cron_control status' to check the results.
v. Add a 'RESET' line to the IMPORT_LOG.TXT file, run 'cron_control status', then 'cron_control reset' and 'cron_control status' again.
vi. Type 'cron_control show'.
vii. Type 'cron_control abort' then 'cron_control status'.

BASELINE TEST PASS CRITERIA
- When no parameters are given, the arguments are displayed.
- The output of cron_control agrees with the output of ps and grep.
- The start command runs correctly and status now shows the cronjob as enabled.
- The stop command runs correctly and the status now shows the cronjob as disabled.
- The status line shows a clock reset. After reset is run, this status is cleared.

- The show command agrees with the unix command 'ps' about the run-status of various cronjob items.
- The abort function kills any cron-related processes running and disables the cronjob.

RESULT: PASS FAIL (circle one) initials

8 TEST COMPLETION

OVERALL: PASS FAIL

Samantha Patterson 3-24-4
TEST OPERATOR (signature) Date

SU GP-B INSP 31

Dorrene Pr 3-24-04
QA WITNESS Date

10. GLOSSARY

This section contains an alphabetic list and definitions of all acronyms used in the document, all proper nouns, and any words used in a non-standard way.

Word	Detail
CSCI	Computer Software Configuration Item
LASP	Laboratory for Atmospheric and Space Physics, University of Colorado
moc-server	Host name of the SUN computer that is the primary server for the MOC.
science server	Host name of the SUN computer which is the primary server for science LAN
SAFS	Standard Autonomous File Server (GSFC facility)
MOC	Mission Operations Center
TCAD	Telemetry Checking, Analysis, and Display
TCNV	Test Case Number which Validates the fix
TDP	Telemetry Data Processing
Startup window	The window containing the Unix command line from which TCAD was started
FEP	Front End Processor
IPDU	Internet Protocol Data Unit
VDD	Version Description Document
MSS	Mission Support Software. This MSS database is the flight Sybase database containing all telemetry monitor information and format definitions, among other items.
DBROget	Database ReadOut retrieval program for APID 300 packets from the Level 0 database.

MROget	Memory ReadOut retrieval program for APID 2xx packets from the Level 0 database.
Eventget	Event retrieval program for APID 301 packets from the Level 0 database.
Snapread.m	Snapshot retrieval program for APID 400 packets from the Level 0 Snapshots data table.

MCR REPORT No. *308*

Change Request: ***LASP/CRON: combine and release as a package***

D Log No.

Change Type. ***Discrepancy***

Date of Event: ***03/16/04***

Subystem: ***TDP/TCAD***

Owner: ***J Spencer***

Reported By: ***R Sharbaugh***

PCB Number:

SW Level:

Action Due: / /

Status: ***PCB Required***

Planned Code Release:

Actual Code Release:

Description:

auto_import in 2.2.1 is broken. To fix this, combine cron back in with lasp, and for the next release test ALL components.

Once S0501 is approved, allow IDPT the ok to have a child workspace created in /apps/supported that has ONLY the auto_import and cron directories. This workspace will be owned by user 'tdp'.

Recommendation:

3/16/2004: Approved for LASP 2.2 release by MCCB.

NASA Notification:

Documents Affected:

New SVP for auto_import.
VDD
S0613, SDD for lasp package, now includes auto_import and cron.

Verification Test Cases:

cron and auto_import pass unit and integration testing.

Status Notes: Tested & passed, 3/24/04 [illegible] SU GP-B INSP 31

Signoffs:

R Brumley _____ _____
K Burlingham _____ _____
R Sharbaugh _____ _____
J Spencer _____ _____

MCR REPORT No. *313*

Change Request: ***auto_import/cron changes for lasp-2.3***

D Log No.

Change Type. ***Change Request***

Date of Event: ***03/17/04***

Subystem: ***Cron***

Owner: ***S Patterson***

Reported By: ***R Sharbaugh***

PCB Number: ***Not Required***

SW Level:

Action Due: / /

Status: ***PCB Required***

Planned Code Release: ***lasp-2.3***

Actual Code Release:

Description:

The following represent Samantha's cron/auto_import directory putback changes:

1) Changed mail_summaries so that it wouldn't spam users.
2) Changed logbook_reports so that it wouldn't spam the logbook.
3) Added code to science_auto.cron so that it would ignore 0-length files.
4) Changed science_auto.cron to have a config file (science_auto.conf)
5) Changed auto_import to delete contents of output directory prior to running so that no false reports are generated. Also added an additional check to verify we weren't going to 'mingle' data in manual mode. (filecheck.exp)
6) Added server name to cronjob output to make it more friendly to DP team.
7) Fixed minor bugs in the expect handling of SCP/SSH for auto_import.
8) Removed 'src' from paths in defaults.exp. Added a few modules to the additional modules calls.
9) Fixed push to web directory structure change issues.

Recommendation:

NASA Notification:

Documents Affected:

Verification Test Cases:

Validation Test cases for each portion of this MCR are as follows:

SU GP-B INSP 31

1) Changed mail_summaries so that it wouldn't spam users.
--P1078 section 10.5

SU GP-B INSP 31

2) Changed logbook_reports so that it wouldn't spam the logbook.
--P1078 section 10.4

MCR REPORT No. *313*

Change Request: ***auto_import/cron changes for lasp-2.3***

D Log No. Change Type. ***Change Request***

Date of Event: ***03/17/04*** Subystem: ***Cron***

Owner: ***S Patterson*** Reported By: ***R Sharbaugh*** PCB Number: ***Not Required***

SW Level: Action Due: / / Status: ***PCB Required***

Planned Code Release: ***lasp-2.3*** Actual Code Release:

SU GP-B INSP 31 Passed DMPm 3/24/04

3) Added code to science_auto.cron so that it would ignore 0-length files. –
--Use the "touch filename" command in UNIX to make up a file name of zero length in the v12, vc0 and framex directories. If the test is successful, cron will ignore them and they will not enter the "processed" directory.

SU GP-B INSP 31 Passed DMPm 3/24/04

4) Changed science_auto.cron to have a config file (science_auto.conf) –
--Change a setting, such as LOCAL(user) in the configuration file. If the user appears different and does not have proper permissions or paths, the config file is being used.

SU GP-B INSP 31

5) Changed auto_import to delete contents of output directory prior to running so that no false reports are generated. Also added an additional check to verify we weren't going to 'mingle' data in manual mode. (filecheck.exp)
--P1078 section 10.2

SU GP-B INSP 31

6) Added server name to cronjob output to make it more friendly to DP team.
--Check output of cronjob for server name. If it exists and is correct, test is successful.

SU GP-B INSP 31 Passed DMPm 3/24/04

7) Fixed minor bugs in the expect handling of SCP/SSH for auto_import. –
--P1078 section 10.3

SU GP-B INSP 31

8) Removed 'src' from paths in defaults.exp. Added a few modules to the additional modules calls.
--Do a grep on the string "src" in defaults.exp. If nothing returns, then the test is successful. Check that additional modules are running based on expected results.

SU GP-B INSP 31

9) Fixed push to web directory structure change issues.
--P1078 section 10.3

Status Notes: 3/24/04 1-9 have passed testing
DMPm SU GP-B INSP 31

MCR REPORT No. *313*

Change Request: ***auto_import/cron changes for lasp-2.3***

D Log No.

Change Type. ***Change Request***

Date of Event: ***03/17/04***

Subystem: ***Cron***

Owner: ***S Patterson***

Reported By: ***R Sharbaugh***

PCB Number: ***Not Required***

SW Level:

Action Due: / /

Status: ***PCB Required***

Planned Code Release: ***lasp-2.3***

Actual Code Release:

Signoffs:

R Brumley _____ _____

K Burlingham _____ _____

S Patterson _____ _____

R Sharbaugh _____ _____

P Shestople _____ _____

J Spencer _____ _____

AS-RUN
ATTACH TO SOP32-B

W.W. Hansen Experimental Physics Laboratory
STANFORD UNIVERSITY
STANFORD, CA 94305 – 4085

Gravity Probe B Relativity Mission

Operational Procedure for Baseline and Regression Testing of Auto Import

P1078 Rev -
3/18/2004

Approvals

NAME	SIGNATURE	DATE
Samantha Patterson *Software Engineer*	Samantha Patterson	18 March 2004
Jennifer Spencer *Data Processing Lead*	[illegible]	18 March 2004
Ron Sharbaugh *S/W Manager*	[illegible]	18 Mar 04
Marcie Smith *Mission Operations Manager*	Marcie Smith	18 Mar 04
Kelly Burlingham *Software Quality Engineer*	Dorrene Ross for Kelly Burlingham	Mar. 22, 2004

Required Signatures prior to Execution

NAME	SIGNATURE	DATE
NAME: Samantha Patterson *Test Engineer*	Samantha Patterson	3-23-4
Kelly Burlingham *Software Quality Engineer*	Dorrene Ross	3-23-04

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

[illegible] 3-22-04

Table of Contents:

1 REVISION HISTORY

REV	DATE	AUTHOR	COMMENTS
-	18 Mar 2004	SAP	initial version

2 SCOPE

This Test Plan Document details the auto import software package and how to both baseline and regression test it.

3 OPERATIONAL PERSONNEL

Jennifer Spencer
Samantha Patterson
Qualified QA Rep: Kelly Burlingham

4 QUALITY ASSURANCE PROVISIONS

4.1 Notification

Quality Assurance must be given 24 hour notification before this test is run; presence is at their discretion.

QA Notified Date & Time: 3/19/04 10:00 By: Ron Shaubaugh
QA Initials: DMR

4.2 Red-Line Authority

Authority to red-line (make minor changes during execution) this procedure is given solely to the test engineer, and shall be approved by QA.

5 RISKS & CONSTRAINTS

5.1 Hardware and Software Requirements

Operations are performed on the Sun server machine known as "science" or "moc-server". This application require that sybase be running and the GPB_L0, GPB_L1 and GPB_L1A databases are available for reading.

5.2 Configuration Requirements

The operator must be logged into the server as a user in the group 'users'. The user should reserve a directory for test data and save this data for QA.

Directory: /apps/supported/lasp-2.3/auto_import

5.3 Constraints

Constraint	Risk
L0 & L1 databases online	This system accesses the GPB_L0, GPB_L1 and GPB_L1A databases. All must be accessible.

6 REFERENCE DOCUMENTS

Document No.	Document
S0331	Data Management Plan
S0401	Stanford Post-Processing Operations for Science Mission Data
S0476	MOC Configuration Control, IONET LAN
S0613	TDP/TCAD Software Release (Design Document)
P0826	Telemetry Data Processing (TDP) in the Non-Real-Time System

7 SOURCE CODE PATH ON SCIENCE/MOC

This software is installed under the apps/supported/lasp-X.X/src directory structure at SU on the science and moc network.

7.1 /auto_import

Program files and versions are as follows:

File	SCCS Version
auto_import.exp	
defaults.exp	
initialize.exp	
filecheck.exp	See Attachment 1
idl_routines.exp	
network.exp	
sql_routines.exp	
proc_handling.exp	
archive.exp	
dparchive.exp	
dpreport.exp	
logbook_reports.exp	
mail_summaries.exp	

8 TEST ENVIRONMENT

This software is tested on the science and moc server at SU under the following configuration:

Software Configurations	Configuration Number (fill in)
IDL Version	5.4
Sybase ASE	12.5
Server (indicate moc or science)	science

9 OPERATING SYSTEM

This section describes the other software that is required to be in place for implementation of this delivery.

Operating System	Minimum Version	Description
Solaris Operating System	2.8	SUN's UNIX operating system.

10 TEST CASES FOR AUTO IMPORT

If the date or RCS Version on any of the following files has changed, the tests in the corresponding section number must be run and checked off for verification. Throughout all phases of testing, all constraints (specified in section 4 of this document) must be met. Additional constraints may be set on an individual test case basis as noted in the Test Sections below. The objective of testing in auto import is to confirm that the data

processed to TDP automatically is done as it would be by an operator, and that all modules of auto import respond without causing processing to inappropriately stop.

LASP version being tested: 2.3

File	Test Name
auto_import.exp	AI PRERUN
defaults.exp	AI PRERUN
initialize.exp	AI PRERUN
filecheck.exp	AI MAIN
idl_routines.exp	AI MAIN
network.exp	AI MAIN
sql_routines.exp	AI MAIN
proc_handling.exp	AI MAIN
archive.exp	AI WEB
dparchive.exp	AI WEB
dpreport.exp	AI WEB
logbook_reports.exp	AI LOGBOOK
mail_summaries.exp	AI MAIL

10.1 AI PRERUN: Auto Import general functionality

☑ (B) ☐ (R)

Test Case Verification Number: AI PRERUN

INTRODUCTION
This test is a toplevel check of auto_import's general functionality and configuration. The startup for this tool has a lot of built-in flexibility because user requirements change frequently. In the auto_import file **defaults.exp**, any variable in the VARS structure which is named in all upper-case may be changed with a command-line flag. –VARNAME will set the variable to 1. –VARNAME=value will set the variable to the value. Strings may be quoted, the variable name on the command-line is case insensitive.

APPROACH
Use Unix tools, run tests with dummy files and in check-only mode.

FEATURES TO BE TESTED

- ☐ Syntax message.
- ☐ Default configuration file settings.
- ☐ Setting flags on the command line.

FEATURES NOT TO BE TESTED

- Functionality of flags.

BASELINE TESTS

i. ls –l the defaults.exp file. The user tdp should be able to modify this file.
ii. Run auto_import with no parameters.
iii. Run auto_import with a time of 0 and a bogus command-file name.
iv. Repeat step 2 but add various flags. Non-default settings should be echoed to the screen. (compare with defaults.exp as needed)
v. Create a dummy command-file (touch /tmp/junk). Use this command file and a time of 1 and run auto import. (CTRL-C out of this after satisfied it is sleeping)

BASELINE TEST PASS CRITERIA

i. defaults.exp file is editable by tdp user.
ii. Syntax message is displayed.
iii. Flagged variables are echoed to screen.
iv. No file given causes warning message.
v. When a delay is given, program sleeps after pre-run processing.

~~REGRESSION TESTS~~

i. Attempt to run program with dummy command ~~file,~~ some flagged options, and a delay time.

SU GP-B INSP 31 3/23/04

~~REGRESSION TEST PASS CRITERIA~~

~~Flags are echoed, delay occurs for both~~ the new and old code versions.

RESULT: PASS FAIL (circle one) P initials

10.2 AI MAIN: Auto Import data processing

☒ (B) ☐ (R)

Test Case Verification Number: AI PRERUN

INTRODUCTION

This section addresses the core functionality of the application: processing data.

APPROACH

Unless otherwise indicated, tests should be performed with the following flags set to improve performance time and reduce database risk.

-co_0 –co_1 –snaps=0 –postcheck=0 –tando=0

Extreme care should be taken with this section. There are several safety features in place, but if not typed as above, the tester CAN affect L0 and L1 data accidentally.

It is advisable to run the Unix 'script' command prior to starting testing so that there will be a complete log of the auto_import process. Additionally, set the VARS(user_list) variable to the name of the tester ONLY (thus preventing mailing the entire user community).

FEATURES TO BE TESTED

- ❑ Safety features for data integrity.
- ❑ Use of flags.
- ❑ Alternate paths.

FEATURES NOT TO BE TESTED

- Ancillary reports (push to web, logbook entries, etc)

BASELINE TESTS

i. Generate a space-delimited command file with data from an existing cycle
ii. Run
auto_import.exp 0 <command_file> -co_0 –co_1 –snaps=0 –postcheck=0 –tando=0 –splice=0
iii. Test both yes and no answers encountered.
iv. Rerun steps i-iii with the addition of the '-auto' flag.
v. Generate a command file where the data is from a time range different from the time range of the cycle. Re-run steps i-iii.
vi. Create a command file with a dataset known to have splices and some bogus time ranges (functional test data). Issue the following command:
auto_import.exp 0 <command_file> -co_0 –co_1 –snaps=0 –postcheck=0 –tando=0
When prompted, read the .is_exp and answer yes or no to the questions accordingly based on splice information.
vii. Rerun the previous test with the addition of the '-auto' flag.
viii. Create a junk data file in the /apps/supported/lasp/data directory and immediately run auto_import 0 <command_file> -co_0 –co_1 –snaps=0 –postcheck=0 –tando=0 –splice=0
ix. Create a command file with legitimate data for two or more different cycles.
x. Add an invalid MSSID to the line in the command file.
xi. Select known good, and known previously fully imported data for a cycle OR select a dummy cycle. Run auto_import without additional processing flags EX:
auto_import.exp 0 <cmd_file>
Make note of the run time. TIME: 22:06:58 3-23-04

BASELINE TEST PASS CRITERIA

i. In manual mode, user is prompted to ask if they wish to import data into an existing cycle.
ii. In auto mode, the user is not prompted about importing data into an existing cycle.
iii. A time difference warning is generated and import is aborted.
iv. In manual mode, when splice data is present, user is prompted about which splices to run.
v. In auto mode, a file containing large time jumps is skipped.
vi. When a current file is present in the data import directory, the user is prompted with a yes/no about using the directory.
vii. When different cycles are imported, data is sent to the correct cycle.
viii. File is skipped if MSSID is invalid.
ix. Data processing completes normally.

~~REGRESSION TESTS~~

i. Run an import with all the disable flags for an ~~existing cycle.~~
ii. Run an import ~~with none~~ of the disable flags ~~using either~~ already processed data or a dummy cycle. ~~Make note of~~ the run time.
TIME: _____ SU GP-B INSP 31 3/23/04

REGRESSION TEST PASS CRITERIA

i. ~~Disabled~~ actions be have as expected.
ii. Data processing ~~completes normally~~.

RESULT: PASS FAIL (circle one) _____ initials

10.3 AI WEB: Data summaries on the web.

☒ (B) ☐ (R)

Test Case Verification Number: AI WEB

INTRODUCTION
After either the baseline or regression test of section 10.2, entries should have been put on the internet. The time recorded in 10.2 will be used correlate the actual data with the internet files.

APPROACH
Visual inspection.

FEATURES TO BE TESTED

☐ Data processing reports on the web.

FEATURES NOT TO BE TESTED

- Quality of data.

BASELINE TESTS

i. From GPBOPS1 go to the /var/www/published/prod/htdoc/dp directory.
ii. Look for file(s) created slightly later than the start of the test process in section 10.2 in the L0_Summary, and L1_Summary directories. View contents of these files.
iii. Connect to https://gpbops.stanford.edu/ Data Processing section and look at the L0 and L1 summaries.

BASELINE TEST PASS CRITERIA

i. File(s) exist and are readable by 'other' but not writable or executable by 'other'.
ii. Files are non-zero in size and contain summary reports matching the data that was imported in step 10.2
iii. The L0 and L1 summaries appear where they should on the outside website and contain the correct data.
iv. Visually, the files on gpbops and gpbops1 match.

~~REGRESSION TESTS~~

3/23/04 SUP P-B SUP P INSP 391 Verify that new ~~summaries were sent to the outside web~~ server (https://gpbops ~~stanford~~.edu/) when step 10.2 was run.

~~REGRESSION TEST PASS~~ CRITERIA

Files are on web.

RESULT: PASS FAIL (circle one) P initials

10.4 AI LOGBOOK: Data summaries in logbook.

☐ (B)

Test Case Verification Number: AI LOGBOOK

INTRODUCTION

After either the baseline or regression test of section 10.2, entries should have been sent to the logbook. The time recorded in 10.2 will be used correlate the actual data with the logbook entries.

APPROACH

Visual inspection.

FEATURES TO BE TESTED

- ☐ Data processing to logbook.

FEATURES NOT TO BE TESTED

- Mail and web messages.
- Quality of data.

BASELINE TESTS

i. Start TQSM.
ii. View data import entry.

BASELINE TEST PASS CRITERIA

i. Timestamp on entry is consistent with expected time of entry per test 10.2.
ii. Text of entry contains the L0, L1, and Limits summaries.

RESULT: PASS FAIL (circle one) initials

10.5 AI MAIL: Data summaries to e-mail.

☒ (B)

Test Case Verification Number: AI MAIL

INTRODUCTION

After either the baseline or regression test of section 10.2 a message should have been mailed to the users defined in the VARS(user_list) variable. The time recorded in 10.2 will be used to match the mail to the processing.

APPROACH

Visual inspection.

FEATURES TO BE TESTED

- ☐ Mailed summaries.

FEATURES NOT TO BE TESTED

- Logbook and web messages.
- Quality of summary.

BASELINE TESTS

i. Look at VARS(user_list) in defaults.exp. As one of the users in the list, open mail client. Get new messages. Compare e-mailed log messages with web summary.

BASELINE TEST PASS CRITERIA

i. Summaries are consistent.
ii. For a file with multiple splices, they are all comprised into a single e-mail. An overall start and end time is at the top of the mail message.

SU GP-B INSP 31 Results: Pass Fail

initials

11 COMPLETION OF P1078:

ALL TESTING PASSED (CIRCLE ONE): YES – PASSED NO

SOFTWARE RELEASE ALLOWED: YES NO

CONDITIONS/NOTES (IF ANY):

QA Review of process Dorene Pass Date: 3/23/04

TEST RUN BY (signature) Samantha Patterson Date: 3-23-04

12 GLOSSARY

This section contains an alphabetic list and definitions of all acronyms used in the document, all proper nouns, and any words used in a non-standard way.

Word	Detail
CSCI	Computer Software Configuration Item
LASP	Laboratory for Atmospheric and Space Physics, University of Colorado
moc-server	Host name of the SUN computer that is the primary server for the MOC.
science server	Host name of the SUN computer which is the primary server for science LAN
SAFS	Standard Autonomous File Server (GSFC facility)
TCAD	Telemetry Checking, Analysis, and Display
TDP	Telemetry Data Processing
Startup window	The window containing the Unix command line from which TCAD was started

P1078, Rev – Attachment 1

version of "archive.exp":	2.3
version of "auto_import.exp":	2.7
version of "defaults.exp":	2.6
version of "dparchive.exp":	2.3
version of "dpreport.exp":	2.3
version of "filecheck.exp":	1.3
version of "idl_routines.exp":	2.5
version of "initialize.exp":	2.4
version of "logbook_reports.exp":	1.4
version of "mail_summaries.exp":	2.4
version of "network.exp":	2.5
version of "proc_handling.exp":	2.3
version of "sql_routines.exp":	2.4

SU GP-B INSP 31 3/23/04

MCR REPORT No. *310*

Change Request: ***TDP auto import routine: add the -s parameter***

D Log No.

Change Type.

Date of Event: ***03/16/04***

Subystem: ***TDP/TCAD***

Owner:

Reported By: ***J Spencer***

PCB Number: ***Not Required***

SW Level:

Action Due: / /

Status:

Planned Code Release:

Actual Code Release:

Description:

Adding the -s option to auto import's parameter list will allow processing of data to different Sybase server(s). This may be needed during times when GPB is running on the backup system and trying to "catch up" the production server after a maintenance period.

Recommendation:

Incorporate MCR. Without it, personnel will have to hand-process potentially large numbers of data files and wait for all processing to be done. This could create a stress on personnel schedule and manpower.

NASA Notification:

Documents Affected:

Verification Test Cases:

P1078

Status Notes: Tested & accepted DMPass 3/23/04 SU GP-B INSP 31

Signoffs:

MCR REPORT No. *210*

Change Request: ***automated import does not run directly from MOC***

D Log No.

Change Type. ***Discrepancy***

Date of Event: ***08/15/03***

Subystem: ***Cron***

Owner: ***S Patterson***

Reported By: ***S Patterson***

PCB Number: ***633***

SW Level:

Action Due: / /

Status: ***In Process***

Planned Code Release: ***1.1***

Actual Code Release:

Description:

The portion of the code which does the actual import had path and permission issues with running directly on moc-server. The code needs to be changed to allow it to run from this system.

If this code is to be run from a moc client rather than moc-server, additional testing is needed.

Recommendation:

8/8/03 Approved by committee for Cron 1.1

NASA Notification:

Documents Affected:

Verification Test Cases:

Run auto_import in check-only mode on moc-server. Ensure that processes start and end smoothly for success.

Status Notes:

8/23/03 code complete

9/03 failed testing

3/18/04 Recommend Withdrawl of this MCR as it is covered by MCR 308 (MCR 210 should have been dispositioned to a MOC action item as it has to do with network permissions and paths rather than auto_import.exp intrinsically). -J Spencer, R Sharbaugh.

3/23/04 Tested & accepted DMB

SU GP-B INSP 31

Signoffs:

K Burlingham _____ _____

S Patterson _____ _____

R Sharbaugh _____ _____

M Smith _____ _____

R Torii _____ _____

AS-RUN
ATTACH TO 5973-B

W.W. Hansen Experimental Physics Laboratory
STANFORD UNIVERSITY
STANFORD, CA 94305 – 4085

Gravity Probe B Relativity Mission

Operational Procedure for Baseline and Regression Testing of Auto Import

P1078 Rev -
3/18/2004

Approvals

NAME	SIGNATURE	DATE
Samantha Patterson *Software Engineer*	Samantha Patterson	18 March 2004
Jennifer Spencer *Data Processing Lead*	[illegible]	18 March 2004
Ron Sharbaugh *S/W Manager*	[illegible]	18 Mar 04
Marcie Smith *Mission Operations Manager*	Marcie Smith	18 Mar 04
Kelly Burlingham *Software Quality Engineer*	Dorrene Ross for Kelly Burlingham	Mar. 22, 2004

Required Signatures prior to Execution

NAME	SIGNATURE	DATE
NAME: *Test Engineer* Samantha Patterson	Samantha Patterson	3-23-04
Kelly Burlingham *Software Quality Engineer*	Dorrene Ross	3-23-04

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

[illegible] 3-22-04

Table of Contents:

1 REVISION HISTORY

REV	DATE	AUTHOR	COMMENTS
-	18 Mar 2004	SAP	initial version

2 SCOPE

This Test Plan Document details the auto import software package and how to both baseline and regression test it.

3 OPERATIONAL PERSONNEL

Jennifer Spencer
Samantha Patterson
Qualified QA Rep: Kelly Burlingham

4 QUALITY ASSURANCE PROVISIONS

4.1 Notification

Quality Assurance must be given 24 hour notification before this test is run; presence is at their discretion.

QA Notified Date & Time: 3/19/04 10:00 am By: Ron Shaubaugh
QA Initials: DMR

4.2 Red-Line Authority

Authority to red-line (make minor changes during execution) this procedure is given solely to the test engineer, and shall be approved by QA.

5 RISKS & CONSTRAINTS

5.1 Hardware and Software Requirements

Operations are performed on the Sun server machine known as "science" or "moc-server". This application require that sybase be running and the GPB_L0, GPB_L1 and GPB_L1A databases are available for reading.

5.2 Configuration Requirements

The operator must be logged into the server as a user in the group 'users'. The user should reserve a directory for test data and save this data for QA.

Directory: /apps/supported/lasp-2.3

5.3 Constraints

Constraint	Risk
L0 & L1 databases online	This system accesses the GPB_L0, GPB_L1 and GPB_L1A databases. All must be accessible.

6 REFERENCE DOCUMENTS

Document No.	Document
S0331	Data Management Plan
S0401	Stanford Post-Processing Operations for Science Mission Data
S0476	MOC Configuration Control, IONET LAN
S0613	TDP/TCAD Software Release (Design Document)
P0826	Telemetry Data Processing (TDP) in the Non-Real-Time System

7 SOURCE CODE PATH ON SCIENCE/MOC

This software is installed under the apps/supported/lasp-X.X/src directory structure at SU on the science and moc network.

7.1 /auto_import

Program files and versions are as follows:

File	SCCS Version
auto_import.exp	
defaults.exp	
initialize.exp	
filecheck.exp	
idl_routines.exp	See Attachment 7
network.exp	
sql_routines.exp	
proc_handling.exp	
archive.exp	
dparchive.exp	
dpreport.exp	
logbook_reports.exp	
mail_summaries.exp	

8 TEST ENVIRONMENT

This software is tested on the science and moc server at SU under the following configuration:

Software Configurations	Configuration Number (fill in)
IDL Version	5.4
Sybase ASE	12.5
Server (indicate moc or science)	MOC

9 OPERATING SYSTEM

This section describes the other software that is required to be in place for implementation of this delivery.

Operating System	Minimum Version	Description
Solaris Operating System	2.8	SUN's UNIX operating system.

10 TEST CASES FOR AUTO IMPORT

If the date or RCS Version on any of the following files has changed, the tests in the corresponding section number must be run and checked off for verification. Throughout all phases of testing, all constraints (specified in section 4 of this document) must be met. Additional constraints may be set on an individual test case basis as noted in the Test Sections below. The objective of testing in auto import is to confirm that the data

processed to TDP automatically is done as it would be by an operator, and that all modules of auto import respond without causing processing to inappropriately stop.

LASP version being tested: 2.3

File	Test Name
auto_import.exp	AI PRERUN
defaults.exp	AI PRERUN
initialize.exp	AI PRERUN
filecheck.exp	AI MAIN
idl_routines.exp	AI MAIN
network.exp	AI MAIN
sql_routines.exp	AI MAIN
proc_handling.exp	AI MAIN
archive.exp	AI WEB
dparchive.exp	AI WEB
dpreport.exp	AI WEB
logbook_reports.exp	AI LOGBOOK
mail_summaries.exp	AI MAIL

10.1 AI PRERUN: Auto Import general functionality

☐ (B) ☒ (R)

Test Case Verification Number: AI PRERUN

INTRODUCTION
This test is a toplevel check of auto_import's general functionality and configuration. The startup for this tool has a lot of built-in flexibility because user requirements change frequently. In the auto_import file **defaults.exp**, any variable in the VARS structure which is named in all upper-case may be changed with a command-line flag. –VARNAME will set the variable to 1. –VARNAME=value will set the variable to the value. Strings may be quoted, the variable name on the command-line is case insensitive.

APPROACH
Use Unix tools, run tests with dummy files and in check-only mode.

FEATURES TO BE TESTED

- ☐ Syntax message.
- ☐ Default configuration file settings.
- ☐ Setting flags on the command line.

FEATURES NOT TO BE TESTED

- Functionality of flags.

BASELINE TESTS

i. ls –l the defaults.exp file. The user tdp should be able to modify this file.
ii. Run auto_import with no parameters.
iii. Run auto_import with a time of 0 and a bogus command-file name.
iv. Repeat step 2 but add various flags. Non-default settings should be echoed to the screen. (compare with defaults.exp as needed)
v. Create a dummy command-file (touch /tmp/junk). Use this command file and a time of 1 and run auto_import. (CTRL-C out of this after satisfied it is sleeping)

AUCB B INSP 31 3/23/04

BASELINE TEST PASS CRITERIA

i. defaults.exp file is editable by tdp user.
ii. Syntax message is displayed.
iii. Flagged variables are echoed to screen.
iv. No file given causes warning message.
v. When a delay is given, program sleeps after pre-run processing.

REGRESSION TESTS

i. Attempt to run program with dummy command file, some flagged options, and a delay time.

REGRESSION TEST PASS CRITERIA

Flags are echoed, delay occurs for both the new and old code versions.

RESULT: PASS FAIL (circle one) _____ initials

10.2 AI MAIN: Auto Import data processing

☐ (B) ☒ (R)

Test Case Verification Number: AI PRERUN

INTRODUCTION

This section addresses the core functionality of the application: processing data.

APPROACH

Unless otherwise indicated, tests should be performed with the following flags set to improve performance time and reduce database risk.

-co_0 –co_1 –snaps=0 –postcheck=0 –tando=0

Extreme care should be taken with this section. There are several safety features in place, but if not typed as above, the tester CAN affect L0 and L1 data accidentally.

It is advisable to run the Unix 'script' command prior to starting testing so that there will be a complete log of the auto_import process. Additionally, set the VARS(user_list) variable to the name of the tester ONLY (thus preventing mailing the entire user community).

FEATURES TO BE TESTED

- ❑ Safety features for data integrity.
- ❑ Use of flags.
- ❑ Alternate paths.

FEATURES NOT TO BE TESTED

- Ancillary reports (push to web, logbook entries, etc)

BASELINE TESTS

i. Generate a space-delimited command file with data from an existing cycle
ii. Run
auto_import.exp 0 <command_file> -co_0 –co_1 –snaps=0 –postcheck=0 –tando=0 –splice=0
iii. Test both yes and no answers encountered.
iv. Rerun steps i-iii with the addition of the '-auto' flag.
v. Generate a command file where the data is from a time range different from the time range of the cycle. Re-run steps i-iii.
vi. Create a command file with a dataset known to have splices and some bogus time ranges (functional test data). Issue the following command:
auto_import.exp 0 <command_file> -co_0 –co_1 –snaps=0 –postcheck=0 –tando=0
When prompted, read the .is_exp and answer yes or no to the questions accordingly based on splice information.
vii. Rerun the previous test with the addition of the '-auto' flag.
viii. Create a junk data file in the /apps/supported/lasp/data directory and immediately run auto_import 0 <command_file> -co_0 –co_1 –snaps=0 –postcheck=0 –tando=0 –splice=0
ix. Create a command file with legitimate data for two or more different cycles.
x. Add an invalid MSSID to the line in the command file.
xi. Select known good, and known previously fully imported data for a cycle OR select a dummy cycle. Run auto_import without additional processing flags EX: auto_import.exp 0 <cmd_file>
Make note of the run time. TIME:_____

BASELINE TEST PASS CRITERIA

i. In manual mode, user is prompted to ask if they wish to import data into an existing cycle.
ii. In auto mode, the user is not prompted about importing data into an existing cycle.
iii. A time difference warning is generated and import is aborted.
iv. In manual mode, when splice data is present, user is prompted about which splices to run. INSP 31 3/23/04
v. In auto mode, a file containing large time jumps is skipped.
vi. When a current file is present in the data import directory, the user is prompted with a yes/no about using the directory.
vii. When different cycles are imported, data is sent to the correct cycle.
viii. File is skipped if MSSID is invalid.
ix. Data processing completes normally.

REGRESSION TESTS
i. Run an import with all the disable flags for an existing cycle.
ii. Run an import with none of the disable flags using either already processed data or a dummy cycle. Make note of the run time.
TIME: 3-23-04 23:32:20

REGRESSION TEST PASS CRITERIA
i. Disabled actions be have as expected.
ii. Data processing completes normally.

RESULT: PASS FAIL (circle one) P initials

10.3 AI WEB: Data summaries on the web.

☐ (B) ☒ (R)

Test Case Verification Number: AI WEB

INTRODUCTION
After either the baseline or regression test of section 10.2, entries should have been put on the internet. The time recorded in 10.2 will be used correlate the actual data with the internet files.

APPROACH
Visual inspection.

FEATURES TO BE TESTED
☐ Data processing reports on the web.

FEATURES NOT TO BE TESTED
- Quality of data.

BASELINE TESTS

i. From GPBOPS1 go to the /var/www/published/prod/htdoc/dp directory.
ii. Look for file(s) created slightly later than the start of the test process in section 10.2 in the L0_Summary, and L1_Summary directories. View contents of these files.
iii. Connect to https://gpbops.stanford.edu/ Data Processing section and look at the L0 and L1 summaries.

SU GP-B INSP 31 3/23/04

BASELINE TEST PASS CRITERIA

i. File(s) exist and are readable by 'other' but not writable or executable by 'other'.
ii. Files are non-zero in size and contain summary reports matching the data that was imported in step 10.2.
iii. The L0 and L1 summaries appear where they should on the outside website and contain the correct data.
iv. Visually, the files on gpbops and gpbops1 match.

REGRESSION TESTS

i. Verify that new summaries were sent to the outside web server (https://gpbops.stanford.edu/) when step 10.2 was run.

REGRESSION TEST PASS CRITERIA

Files are on web.

RESULT: PASS FAIL (circle one) D initials

10.4 AI LOGBOOK: Data summaries in logbook.

☒ (B)

Test Case Verification Number: AI LOGBOOK

INTRODUCTION

After either the baseline or regression test of section 10.2, entries should have been sent to the logbook. The time recorded in 10.2 will be used correlate the actual data with the logbook entries.

APPROACH

Visual inspection.

FEATURES TO BE TESTED

- ❑ Data processing to logbook.

FEATURES NOT TO BE TESTED

- Mail and web messages.
- Quality of data.

BASELINE TESTS

i. Start TQSM.
ii. View data import entry.

BASELINE TEST PASS CRITERIA

i. Timestamp on entry is consistent with expected time of entry per test 10.2.
ii. Text of entry contains the L0, L1, and Limits summaries.

RESULT: PASS FAIL (circle one) initials

10.5 AI MAIL: Data summaries to e-mail.

☒(B)

Test Case Verification Number: AI MAIL

INTRODUCTION

After either the baseline or regression test of section 10.2 a message should have been mailed to the users defined in the VARS(user_list) variable. The time recorded in 10.2 will be used to match the mail to the processing.

APPROACH

Visual inspection.

FEATURES TO BE TESTED

- ❑ Mailed summaries.

FEATURES NOT TO BE TESTED

- Logbook and web messages.
- Quality of summary.

BASELINE TESTS

i. Look at VARS(user_list) in defaults.exp. As one of the users in the list, open mail client. Get new messages. Compare e-mailed log messages with web summary.

BASELINE TEST PASS CRITERIA

i. Summaries are consistent.
ii. For a file with multiple splices, they are all comprised into a single e-mail. An overall start and end time is at the top of the mail message.

SU GP-B INSP 31 3/23/04 Pass

11 COMPLETION OF P1078:

ALL TESTING PASSED (CIRCLE ONE): YES – PASSED NO

SOFTWARE RELEASE ALLOWED: YES NO

CONDITIONS/NOTES (IF ANY):

QA Review of process _____ Date: 3-23-04

TEST RUN BY (signature) Samantha Patterson Date: 3-23-04

12 GLOSSARY

This section contains an alphabetic list and definitions of all acronyms used in the document, all proper nouns, and any words used in a non-standard way.

Word	Detail
CSCI	Computer Software Configuration Item
LASP	Laboratory for Atmospheric and Space Physics, University of Colorado
moc-server	Host name of the SUN computer that is the primary server for the MOC.
science server	Host name of the SUN computer which is the primary server for science LAN
SAFS	Standard Autonomous File Server (GSFC facility)
TCAD	Telemetry Checking, Analysis, and Display
TDP	Telemetry Data Processing
Startup window	The window containing the Unix command line from which TCAD was started

P1078, Rev – Attachment 1

version of "archive.exp":	2.3
version of "auto_import.exp":	2.7
version of "defaults.exp":	2.6
version of "dparchive.exp":	2.3
version of "dpreport.exp":	2.3
version of "filecheck.exp":	1.3
version of "idl_routines.exp":	2.5
version of "initialize.exp":	2.4
version of "logbook_reports.exp":	1.4
version of "mail_summaries.exp":	2.4
version of "network.exp":	2.5
version of "proc_handling.exp":	2.3
version of "sql_routines.exp":	2.4

SU GP-B INSP 31 3/23/04

ATTACH TO ~~S0303L~~ S0973-B

LASP-2.3 Code Walkthrough 3/18/04

1) tcad/tcad_plot.pro v2.5 is N/C, sccs "parent" accept.
2) tcad/tminfo.pro - All MCR # 229 Attach #1
- sccs hdr ok.
- Mnemonic vs fmt info is new.
3) tcad/analyze/l1_fmtrepat.pro v2.5 Attach 2
- All MCR #301
- sccs hdr good
- New is outfile03, and03, etc ✓
4) dd/db_routines.EP Att 3
- NO sccs header info
- format retrieval additions
- NO cursor fetch chgs for speed
5) analyze/snapshot.pro - Att 4
- No sccs header info
- All chgs per MCR 312

4) tcad/L1_fmtrpt.pro v2.5 MCR 301 Att 5
- sccs ok
-

7) DB_GET_TMFMTS. gph_db.pro v2.3 MCR 229 Att 6
- NO header
8) alpha pathback changes Att 7
2 chgs to tcad dir, both ok.

Summary: tcad + db changes all per MCR 3/24/04
or per alpha bug catch Ron Shulman 3/24/04

SU GP-B INSP 31

```
; /apps/supported/lasp-2.3/tcad/SCCS/s.tcad_tminfo.pro 2.3
; @(#)RELEASE VERSION 2.3
; FILENAME              : tcad_tminfo.pro
; SCCS Delta Date       : 03/14/04,02:45:33
;*****************************************************************************
; TCAD_TMINFO_EVENT - Event handler for the Telemetry Information panel
; GPB version  - Last Update R. Davis - 12 Nov 2001
;*****************************************************************************

pro TCAD_TMINFO_EVENT, event

  ;Check to see if we're resizing the window.
  IF event.id EQ event.top THEN BEGIN   ;Resizing, call resize function.
     TCAD_TMINFO__RESIZE, event
     return
  ENDIF

  ; Get the ID widget_id for the TM Info panel

  base = event.handler

  ; Get the user value attached to this event and use it to determine what
  ; the user did to generate this event

  WIDGET_CONTROL, event.id, GET_UVALUE=uval
  WIDGET_CONTROL, event.top, GET_UVALUE=display_state

  IF NOT KEYWORD_SET(uval) THEN BEGIN
     return
  ENDIF

  case uval of

   "CLOSE": begin

        ; Clicked on Close button. Erase the window and return

        WIDGET_CONTROL, base, /DESTROY

        return

      end

   "MNEMONIC": BEGIN
       TCAD_ITEMS_LIST__GET_MNEMONIC, display_state, /TMINFO
      END

   "SUBSYSTEMS": begin

        ; User clicked within the subsystem list. Display the list of
        ; telemetry items available for the selected subsystem

        TCAD_ITEMS_LIST__DISPLAY_ITEMS, display_state

      end

   "ITEMS": begin

        TCAD_ITEMS_LIST__GET_SUBSYSTEM, display_state
        TCAD_ITEMS_LIST__GET_ITEM, display_state
        TCAD_TMINFO__DISPLAY_DATA, display_state

      end

    else:

  endcase

  return

end

PRO TCAD_TMINFO__RESIZE, event
   WIDGET_CONTROL, event.top, GET_UVALUE=display_state

   base = (*display_state).base
   WinG = widget_info(base,/geometry)          ;Get new size of window.

   ;Create new X and Y sizes for info component.
   newx = event.x - WinG.xpad
   newy = event.y - WinG.ypad

   IF (WIDGET_INFO((*display_state).items_widget,/VALID_ID )) THEN BEGIN
      WIDGET_CONTROL, (*display_state).subsystems_widget, scr_ysize=newy * 0.4
      WIDGET_CONTROL, (*display_state).items_widget, scr_xsize=newx-120, $
                      scr_ysize=newy * 0.4
```

```
; /apps/supported/lasp-2.2.1/tcad/SCCS/s.tcad_tminfo.pro 2.2
; @(#)RELEASE VERSION 2.2
; FILENAME              : tcad_tminfo.pro
; SCCS Delta Date       : 02/06/04,21:45:56
;*****************************************************************************
; TCAD_TMINFO_EVENT - Event handler for the Telemetry Information panel
; GPB version  - Last Update R. Davis - 12 Nov 2001
;*****************************************************************************

pro TCAD_TMINFO_EVENT, event

  ;Check to see if we're resizing the window.
  IF event.id EQ event.top THEN BEGIN     ;Resizing, call resize function.
     TCAD_TMINFO__RESIZE, event
     return
  ENDIF

  ; Get the ID widget_id for the TM Info panel

  base = event.handler

  ; Get the user value attached to this event and use it to determine what
  ; the user did to generate this event

  WIDGET_CONTROL, event.id, GET_UVALUE=uval
  WIDGET_CONTROL, event.top, GET_UVALUE=display_state

  IF NOT KEYWORD_SET(uval) THEN BEGIN
     return
  ENDIF

  case uval of

   "CLOSE": begin

        ; Clicked on Close button. Erase the window and return

        WIDGET_CONTROL, base, /DESTROY

        return

      end

   "MNEMONIC": BEGIN
       TCAD_ITEMS_LIST__GET_MNEMONIC, display_state, /TMINFO
      END

   "SUBSYSTEMS": begin

        ; User clicked within the subsystem list. Display the list of
        ; telemetry items available for the selected subsystem

        TCAD_ITEMS_LIST__DISPLAY_ITEMS, display_state

      end

   "ITEMS": begin

        TCAD_ITEMS_LIST__GET_SUBSYSTEM, display_state
        TCAD_ITEMS_LIST__GET_ITEM, display_state
        TCAD_TMINFO__DISPLAY_DATA, display_state

      end

    else:

  endcase

  return

end

PRO TCAD_TMINFO__RESIZE, event
   WIDGET_CONTROL, event.top, GET_UVALUE=display_state

   base = (*display_state).base
   WinG = widget_info(base,/geometry)          ;Get new size of window.

   ;Create new X and Y sizes for info component.
   newx = event.x - WinG.xpad
   newy = event.y - WinG.ypad

   IF (WIDGET_INFO((*display_state).items_widget,/VALID_ID )) THEN BEGIN
      WIDGET_CONTROL, (*display_state).subsystems_widget, scr_ysize=newy * 0.4
      WIDGET_CONTROL, (*display_state).items_widget, scr_xsize=newx-120, $
                      scr_ysize=newy * 0.4
```

MCR 229

display MNEMONIC vs FORMAT

SUGP-B INSP 31

Attach 1 of 8

```
      TMinfoBase = WIDGET_INFO (base, /CHILD)    ;Get the subwindow.
      TTG  = widget_info(TMinfoBase,/geometry)      ;Get window geometry.
      newy = event.y - WinG.ypad  - (TTG.scr_ysize + TTG.ypad)
   ENDIF

   WIDGET_CONTROL, (*display_state).data_widget, scr_xsize=newx, scr_ysize=newy
END

PRO TCAD_TMINFO__DISPLAY_DATA, display_state
   tmid          = 0
   datatype      = ' '
   start_bit     = 0
   length        = 0
   mssid         = 0
   initial_mssid = 0

   name = (*display_state).selected_item
   DB_GET_TMINFO, name, TMID=tmid, DATATYPE=datatype,       $
                  START_BIT=start_bit, LENGTH=length,       $
                  MSSID=mssid, INITIAL_MSSID=initial_mssid

   n = 0
   data_list = STRARR (300)

   data_list(n+0) = "TMID = " + STRING (tmid, FORMAT="(I5)")
   data_list(n+1) = "Mnemonic = " + STRING((*display_state).selected_item)



   case STRMID (datatype, 0, 1) of
     'R': outstring = "Real "
     'S': outstring = "Subfield "
     'D': outstring = "Derived "
   endcase

   case STRMID (datatype, 1, 1) of
     'D': outstring = outstring + "Discrete "
     'A': outstring = outstring + "Analog "
   endcase

   case STRMID (datatype, 2, 1) of
     'F': outstring = outstring + "(Floating Point)"
     'S': outstring = outstring + "(Signed Integer)"
     'U': outstring = outstring + "(Unsigned Integer)"
     else:
   endcase

   data_list(n+2) = "Latest MSS ID = " + STRING (mssid, FORMAT="(Z4)") + " hex, " + $
                    "  Initial MSS ID = " + STRING (initial_mssid, FORMAT="(Z4)") + " hex"
   data_list(n+3) = "Data Type = " + outstring

   data_list(n+4) = "Start Bit = " + STRING (start_bit, FORMAT="(I2)") + $
                    ",  Length = " + STRING (length, FORMAT="(I2)")

   n = n + 4

   DB_GET_TLMFMTS, tmid, tlmfmts
   PRINT, ""
   PRINT, "  FORMATS = ",tlmfmts.num_fmts
   PRINT, ""
   ;PRINT, tlmfmts.fmt_id
   for ifmt = 1, 255 do begin
       xfmt = ifmt
       if (xfmt eq 255) then xfmt = 555
       irate = tlmfmts.fmt_rate(ifmt)
       if (tlmfmts.mfbyte(ifmt,0) eq 0) then begin
           for iframe = 1, 100 do begin
              if (tlmfmts.mfbyte(ifmt,iframe) ne 0) then begin
                  data_list(n+1)  = "FMT= " + STRING (xfmt, FORMAT="(I3)") + $
                                    STRING (irate, FORMAT="(I4)") + "k" + $
                                    "  MF= " + STRING (iframe, FORMAT="(I3)") + $
                                    "  Byte= " + STRING (tlmfmts.mfbyte(ifmt,iframe), FORMAT="(I
                  n = n + 1
              endif
           endfor
       endif else begin
                  data_list(n+1)  = "FMT= " + STRING (xfmt, FORMAT="(I3)") + $
                                    STRING (irate, FORMAT="(I4)") + "k" + $
                                    "  MF= ALL" + $
                                    "  Byte= " + STRING (tlmfmts.mfbyte(ifmt,0), FORMAT="(I3)")
                  n = n + 1
       endelse
   endfor
```

```
      TMinfoBase = WIDGET_INFO (base, /CHILD)      ;Get the subwindow.
      TTG  = widget_info(TMinfoBase,/geometry)      ;Get window geometry.
      newy = event.y - WinG.ypad  - (TTG.scr_ysize + TTG.ypad)
   ENDIF

   WIDGET_CONTROL, (*display_state).data_widget, scr_xsize=newx, scr_ysize=newy
END

PRO TCAD_TMINFO__DISPLAY_DATA, display_state
   tmid          = 0
   datatype      = ' '
   start_bit     = 0
   length        = 0
   mssid         = 0
   initial_mssid = 0

   name = (*display_state).selected_item
   DB_GET_TMINFO, name, TMID=tmid, DATATYPE=datatype,       $
                  START_BIT=start_bit, LENGTH=length,       $
                  MSSID=mssid, INITIAL_MSSID=initial_mssid

   n = 0
   data_list = STRARR (300)

   data_list(n+0) = "Telemetry Identifier (TMID) = " + STRING (tmid)

   data_list(n+1) = "Mnemonic is:        " + STRING((*display_state).selected_item)

   case STRMID (datatype, 0, 1) of
     'R': outstring = "Real "
     'S': outstring = "Subfield "
     'D': outstring = "Derived "
   endcase

   case STRMID (datatype, 1, 1) of
     'D': outstring = outstring + "Discrete "
     'A': outstring = outstring + "Analog "
   endcase

   case STRMID (datatype, 2, 1) of
     'F': outstring = outstring + "(Floating Point)"
     'S': outstring = outstring + "(Signed Integer)"
     'U': outstring = outstring + "(Unsigned Integer)"
     else:
   endcase

   data_list(n+2) = "Latest MSS ID = " + STRING (mssid, FORMAT="(Z4)") + " hex, " + $
                    "  Initial MSS ID = " + STRING (initial_mssid, FORMAT="(Z4)") + " hex
   data_list(n+3) = "Data Type = " + outstring

   data_list(n+4) = "Start Bit = " + STRING (start_bit, FORMAT="(I2)") + $
                    ",  Length = " + STRING (length, FORMAT="(I2)")

   n = n + 4
```

MCR 229

MCR 229 new tlmfmts ✓

SUBP-B MSP 31

2

```
if STRMID (datatype, 1, 1) ne 'D' then begin

  DB_GET_CALIBRATION, tmid, units, ctype, min_dn, c

  if !db_status gt 0 then begin

    data_list(n+1)  = "Units  = " + units
    n = n + 1

    nn = !db_status

    for i = 0, nn-1 do begin

      data_list(n+1)  = "Coefficient Set " + STRING (i+1, FORMAT="(I2)")+ ':'
      n = n + 1

      if (ctype(i) eq 1) then begin

        data_list(n+1)  = "Type = Polynomial"
        data_list(n+2)  = "Min DN = " + STRING (min_dn(i))
        data_list(n+3)  = "C0 = " + STRING (c(0,i))
        n = n + 3
        if ((c(7,i)) ne 0) or ((c(6,i)) ne 0) or ((c(5,i)) ne 0) or ((c(4,i)) ne 0) or $
           ((c(3,i)) ne 0) or ((c(2,i)) ne 0) or ((c(1,i)) ne 0) then begin
            data_list(n+1) = "C1 = " + STRING (c(1,i))
            n = n + 1
        endif
        if ((c(7,i)) ne 0) or ((c(6,i)) ne 0) or ((c(5,i)) ne 0) or ((c(4,i)) ne 0) or $
           ((c(3,i)) ne 0) or ((c(2,i)) ne 0) then begin
            data_list(n+1) = "C2 = " + STRING (c(2,i))
            n = n + 1
        endif
        if ((c(7,i)) ne 0) or ((c(6,i)) ne 0) or ((c(5,i)) ne 0) or ((c(4,i)) ne 0) or $
           ((c(3,i)) ne 0) then begin
            data_list(n+1) = "C3 = " + STRING (c(3,i))
            n = n + 1
        endif
        if ((c(7,i)) ne 0) or ((c(6,i)) ne 0) or ((c(5,i)) ne 0) or ((c(4,i)) ne 0) then begi
            data_list(n+1) = "C4 = " + STRING (c(4,i))
            n = n + 1
        endif
        if ((c(7,i)) ne 0) or ((c(6,i)) ne 0) or ((c(5,i)) ne 0) then begin
            data_list(n+1) = "C5 = " + STRING (c(5,i))
            n = n + 1
        endif
        if ((c(7,i)) ne 0) or ((c(6,i)) ne 0) then begin
            data_list(n+1) = "C6 = " + STRING (c(6,i))
            n = n + 1
        endif
        if ((c(7,i)) ne 0) then begin
            data_list(n+1) = "C7 = " + STRING (c(7,i))
            n = n + 1
        endif

      endif else begin

        data_list(n+1)  = "Type = Exponential"
        data_list(n+2)  = "Min DN = " + STRING (min_dn(i))
        data_list(n+3)  = "C0 = " + STRING (c(0,i))
        data_list(n+4)  = "C1 = " + STRING (c(1,i))
        data_list(n+5)  = "C2 = " + STRING (c(2,i))
        n = n + 5

      endelse

    endfor

    DB_GET_LIMITS, tmid, red, yellow, display

    if !db_status gt 0 then begin

      data_list(n+1) = "Limits:"
      data_list(n+2) = "Red Lo = " + STRING (red(0))
      data_list(n+3) = "Yel Lo = " + STRING (yellow(0))
      data_list(n+4) = "Yel Hi = " + STRING (yellow(1))
      data_list(n+5) = "Red Hi = " + STRING (red(1))
      data_list(n+6) = "Display Lo = " + STRING (display(0))
      data_list(n+7) = "Display Hi = " + STRING (display(1))
      n = n + 7

    endif else begin

      n = n + 1
      data_list(n) = "No limits available"

    endelse
```

MCR 229

```
if STRMID (datatype, 1, 1) ne 'D' then begin

  DB_GET_CALIBRATION, tmid, units, ctype, min_dn, c

  if !db_status gt 0 then begin

    data_list(n+1)  = "Units  = " + units
    n = n + 1

    nn = !db_status

    for i = 0, nn-1 do begin

      data_list(n+1)  = "Coefficient Set " + STRING (i+1, FORMAT="(I1)")+ ':'
      n = n + 1

      if (ctype(i) eq 1) then begin

        data_list(n+1)  = "Type = Polynomial"
        data_list(n+2)  = "Min DN = " + STRING (min_dn(i))
        data_list(n+3)  = "C0 = " + STRING (c(0,i))
        data_list(n+4)  = "C1 = " + STRING (c(1,i))
        data_list(n+5)  = "C2 = " + STRING (c(2,i))
        data_list(n+6)  = "C3 = " + STRING (c(3,i))
        data_list(n+7)  = "C4 = " + STRING (c(4,i))
        data_list(n+8)  = "C5 = " + STRING (c(5,i))
        data_list(n+9)  = "C6 = " + STRING (c(6,i))
        data_list(n+10) = "C7 = " + STRING (c(7,i))
        n = n + 10

      endif else begin

        data_list(n+1)  = "Type = Exponential"
        data_list(n+2)  = "Min DN = " + STRING (min_dn(i))
        data_list(n+3)  = "C0 = " + STRING (c(0,i))
        data_list(n+4)  = "C1 = " + STRING (c(1,i))
        data_list(n+5)  = "C2 = " + STRING (c(2,i))
        n = n + 5

      endelse

    endfor

    DB_GET_LIMITS, tmid, red, yellow, display

    if !db_status gt 0 then begin

      data_list(n+1) = "Limits:"
      data_list(n+2) = "Red       Low  = " + STRING (red(0))
      data_list(n+3) = "Yellow  Low  = " + STRING (yellow(0))
      data_list(n+4) = "Yellow  High = " + STRING (yellow(1))
      data_list(n+5) = "Red       High = " + STRING (red(1))
      data_list(n+6) = "Display Low = " + STRING (display(0))
      data_list(n+7) = "Display High= " + STRING (display(1))
      n = n + 7

    endif else begin

      n = n + 1
      data_list(n) = "No limits available"

    endelse
```

MCR 229

```
    endif

  endif else if STRMID (datatype, 1, 1) eq 'D' then begin

    DB_GET_STATES, tmid, value, sval

    if !db_status gt 0 then begin

      nd = N_ELEMENTS (value)

      n = n + 1
      data_list(n) = "States:"

      for i = 0, nd-1 do begin
        n = n + 1
        data_list(n) = STRING(value(i)) + " = " + sval(i)
      endfor

    endif else begin

      n = n + 1
      data_list(n) = "No states available"

    endelse

  endif

  WIDGET_CONTROL, (*display_state).data_widget, SET_VALUE=data_list(0:n)

END

;*****************************************************************************
; TCAD_TMINFO - Create the Telemetry Information Panel
;*****************************************************************************

pro TCAD_TMINFO, GROUP_LEADER=group_leader

  ; Log into the database, if we aren't already

  if not !dbi_connected then DB_CONNECT, "gpb","gravitydb", server="GPB_DATA"

  ; Define the structure used to store the widget IDs and the selections
  ; the user makes

  state = {base:                0L, $
           subsystems_widget:   0L, $
           items_widget:        0L, $
           data_widget:         0L, $
           selected_subsystem:  " ", $
           selected_item:       " ", $
           mnemonic_widget:     0L,$
           mnemonic_window:     0L,$
           mnemonic_button:     0, $
           mnemonic_search:     0L,$
           mnemoniclist_widget: 0L,$
           selected_mnemonic:   0L,$
           data_list:           PTR_NEW() }

  display_state = PTR_NEW(state,/NO_COPY)

  ;Create and initialize the individual widgets, starting with the top-level
  ;base of the widget tree

  (*display_state).base = WIDGET_BASE ( TITLE="GPB Telemetry Information", $
                                        /COLUMN, /TLB_SIZE_EVENTS)

  if KEYWORD_SET (GROUP_LEADER) then begin
    WIDGET_CONTROL, (*display_state).base, GROUP_LEADER=group_leader
  endif

  ; Place the subsystem/items lists on the panel

  TCAD_ITEMS_LIST__DRAW, display_state

  ; Place the output text field on the panel

  base1  = WIDGET_BASE ((*display_state).base, /ROW)

  tmp = WIDGET_LIST (base1, XSIZE = 43, YSIZE = 20)

  (*display_state).data_widget = tmp

  base2 = WIDGET_BASE ((*display_state).base, /ROW, XPAD=250)

  tmp = WIDGET_BUTTON (base2, VALUE="     Close     ", UVALUE="CLOSE")
```


```
    endif

  endif else if STRMID (datatype, 1, 1) eq 'D' then begin

    DB_GET_STATES, tmid, value, sval

    if !db_status gt 0 then begin

      nd = N_ELEMENTS (value)

      n = n + 1
      data_list(n) = "States:"

      for i = 0, nd-1 do begin
        n = n + 1
        data_list(n) = STRING(value(i)) + " = " + sval(i)
      endfor

    endif else begin

      n = n + 1
      data_list(n) = "No states available"

    endelse

  endif

  WIDGET_CONTROL, (*display_state).data_widget, SET_VALUE=data_list(0:n)

END

;*****************************************************************************
; TCAD_TMINFO - Create the Telemetry Information Panel
;*****************************************************************************

pro TCAD_TMINFO, GROUP_LEADER=group_leader

  ; Log into the database, if we aren't already

  if not !dbi_connected then DB_CONNECT, "gpb","gravitydb", server="GPB_DATA"

  ; Define the structure used to store the widget IDs and the selections
  ; the user makes

  state = {base:                0L, $
           subsystems_widget:   0L, $
           items_widget:        0L, $
           data_widget:         0L, $
           selected_subsystem:  " ", $
           selected_item:       " ", $
           mnemonic_widget:     0L,$
           mnemonic_window:     0L,$
           mnemonic_button:     0, $
           mnemonic_search:     0L,$
           mnemoniclist_widget: 0L,$
           selected_mnemonic:   0L,$
           data_list:           PTR_NEW() }

  display_state = PTR_NEW(state,/NO_COPY)

  ;Create and initialize the individual widgets, starting with the top-level
  ;base of the widget tree

  (*display_state).base = WIDGET_BASE ( TITLE="GPB Telemetry Information", $
                                        /COLUMN, /TLB_SIZE_EVENTS)

  if KEYWORD_SET (GROUP_LEADER) then begin
    WIDGET_CONTROL, (*display_state).base, GROUP_LEADER=group_leader
  endif

  ; Place the subsystem/items lists on the panel

  TCAD_ITEMS_LIST__DRAW, display_state

  ; Place the output text field on the panel

  base1  = WIDGET_BASE ((*display_state).base, /ROW)

  tmp = WIDGET_LIST (base1, XSIZE = 43, YSIZE = 20)

  (*display_state).data_widget = tmp

  base2 = WIDGET_BASE ((*display_state).base, /ROW, XPAD=250)

  tmp = WIDGET_BUTTON (base2, VALUE="     Close     ", UVALUE="CLOSE")
```


D#1

4

```
  ; Store the state

  WIDGET_CONTROL, (*display_state).base, SET_UVALUE=display_state
  WIDGET_CONTROL, (*display_state).base, /REALIZE

  ;Hand off control of the panel to the XMANAGER

  XMANAGER, 'TCAD_TMINFO', (*display_state).base

  return

end
```


```
  ; Store the state

  WIDGET_CONTROL, (*display_state).base, SET_UVALUE=display_state
  WIDGET_CONTROL, (*display_state).base, /REALIZE

  ;Hand off control of the panel to the XMANAGER

  XMANAGER, 'TCAD_TMINFO', (*display_state).base

  return

end
```

```
; /apps/supported/lasp-2.3/tcad/analyze/SCCS/s.l1_fmtreport.pro 2.5
; @(#)RELEASE VERSION 2.5
; FILENAME              : l1_fmtreport.pro
; SCCS Delta Date       : 03/15/04,02:45:15
;FILE:  L1_fmtreport.pro
;CREATED:  02-04-04 by Jennifer Spencer, from program by S Patterson.
;DESCRIPTION:  This program spawns a new window from the Analyze list and
;  generates a wrapper around an IDL script for fetching slices of data.
;  The resulting data can be either saved to a file and/or be displayed to
;  the screen.  Shows gaps in Level 1 data processing.

PRO L1_FMTREPORT_EVENT, event

  COMMON TCAD_FORMAT_REPORT_INFO, display_options, time_format, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON TDPPATH_COMMON, gcTDPPATH


  ; Get the widget ID for the Display panel
  base = event.handler
  tmpfile = 'tmpfile_' + GETENV("USER")

  ; Get the user value associated with the format report we're processing and use
  ; it to determine what the user did to generate this format report.

  WIDGET_CONTROL, event.top, GET_UVALUE=display_state
  WIDGET_CONTROL, event.id, GET_UVALUE=uval

  case uval of

   "OKAY": begin
      ;  Clicked on OKAY button.  Run the IDL script and do our output.
      TCAD_INPUT_TIMES__GET, display_state
      if not (*display_state).times_valid then return

      ;Generate the variable string to the IDL script.
      cycle = STRTRIM((*display_state).start_sct.cycle)

      L1_FMTRPT,gcTDPPATH,(*display_state).start_sct,(*display_state).stop_sct

      ;Now let's output our results in all the required ways.
      datapath = GETENV("HOME") + '/FMTRPT/'
      outfile32 = datapath + "l1_fmtrpt_"+ STRCOMPRESS(cycle, /remove_all) + "_32k.sum"
      outfile03 = datapath + "l1_fmtrpt_"+ STRCOMPRESS(cycle, /remove_all) + "_03k.sum"

      ;Now we display according to the user's choices.

      if display_options[0] eq 1 then begin     ;To Screen.

         cmd32= "/usr/openwin/bin/textedit -read_only -Ws 1000 500 " + outfile32 +' &'
         SPAWN, cmd32
         cmd03= "/usr/openwin/bin/textedit -read_only -Ws 1000 500 " + outfile03 +' &'
         SPAWN, cmd03

      endif

      if display_options[1] eq 1 then begin     ;To Printer.
         cmd32= "lpr -P " + printer_list[printer] + " " + outfile32
         SPAWN, cmd32
         cmd03= "lpr -P " + printer_list[printer] + " " + outfile03
         SPAWN, cmd03
      endif

      if display_options[2] eq 1 then begin     ;To File.
         ;cmd32= "cp " + outfile32 + " " + filename
         ;SPAWN, cmd32
         ;cmd03= "cp " + outfile03 + " " + filename
         ;SPAWN, cmd03
      endif

      ;WIDGET_CONTROL, base, /DESTROY

      end
   "CANCEL": begin
      ;Delete our tmp file.  Ignore any error messages.
      ;Force us to use sh so we know 2> is STDERR.
      SPAWN, '/bin/sh -c "rm ' + tmpfile + ' 2> /dev/null"'

      ; Clicked on Close Button.  Erase the panel and return
      WIDGET_CONTROL, base, /DESTROY

      return

      end

   "TIMETYPE":   begin
```

MCR 301

MCR 301

```
; /apps/supported/lasp-2.2.1/tcad/analyze/SCCS/s.l1_fmtreport.pro 2.4
; @(#)RELEASE VERSION 2.4
; FILENAME              : l1_fmtreport.pro
; SCCS Delta Date       : 02/25/04,21:45:55
;FILE:  L1_fmtreport.pro
;CREATED:  02-04-04 by Jennifer Spencer, from program by S Patterson.
;DESCRIPTION:  This program spawns a new window from the Analyze list and
;  generates a wrapper around an IDL script for fetching slices of data.
;  The resulting data can be iether saved to a file and/or be displayed to
;  the screen.  Shows gaps in Level 1 data processing.

pro L1_FMTREPORT_EVENT, event

  COMMON TCAD_FORMAT_REPORT_INFO, display_options, time_format, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON TDPPATH_COMMON, gcTDPPATH


  ; Get the widget ID for the Display panel
  base = event.handler
  tmpfile = 'tmpfile_' + GETENV("USER")

  ; Get the user value associated with the format report we're processing and use
  ; it to determine what the user did to generate this format report.

  WIDGET_CONTROL, event.top, GET_UVALUE=display_state
  WIDGET_CONTROL, event.id, GET_UVALUE=uval

  case uval of

   "OKAY": begin
      ;  Clicked on OKAY button.  Run the IDL script and do our output.
      TCAD_INPUT_TIMES__GET, display_state
      if not (*display_state).times_valid then return

      ;Generate the variable string to the IDL script.
      cycle = STRTRIM((*display_state).start_sct.cycle)

      L1_FMTRPT,gcTDPPATH,(*display_state).start_sct,(*display_state).stop_sct

      ;Now let's output our results in all the required ways.
      datapath = GETENV("HOME") + '/FMTRPT/'
      outfile1 = datapath + "l1_fmtrpt_"+ STRCOMPRESS(cycle, /remove_all) + "_32k.sum"

      ;Now we display according to the user's choices.

      if display_options[0] eq 1 then begin       ;To Screen.

         cmd1= "/usr/openwin/bin/textedit -read_only -Ws 1000 500 " + outfile1 +' &'
;         cmd1 = cmd1 + " & "                ;Run in background.
         SPAWN, cmd1

      endif

      if display_options[1] eq 1 then begin       ;To Printer.
         cmd = "lpr -P " + printer_list[printer] + " " + outfile1
         SPAWN, cmd

      endif

      if display_options[2] eq 1 then begin       ;To File.
         ;cmd = "cp " + outfile1 + " " + filename
         ;SPAWN, cmd

      endif

      ;WIDGET_CONTROL, base, /DESTROY

      end
   "CANCEL": begin
      ;Delete our tmp file.  Ignore any error messages.
      ;Force us to use sh so we know 2> is STDERR.
      SPAWN, '/bin/sh -c "rm ' + tmpfile + ' 2> /dev/null"'

      ; Clicked on Close Button.  Erase the panel and return
      WIDGET_CONTROL, base, /DESTROY

      return

      end

   "TIMETYPE":   begin
```

Attach #1

SU GP.B INSP 31

6

```
      TCAD_DISPLAY__TIMETYPE, display_state
     end
   "DISPLAY_OPTIONS": begin              ;A display option was (un)checked
      display_options[event.value] = event.select      ;Record which one.
     end
   "TIME_FORMAT": begin                  ;Changed time format output.
      time_format = event.value          ;Record which radio button was clicked.
   end
   "PRINTER": begin                      ;A new printer was selected.
      printer = event.index              ;Get the list index of the printer.
   end
   "FILENAME": begin                     ;A filename was entered.
      filename = event.value
   end

  else:BEGIN
    TCAD_AUTOFILL_TIMES, event
   end

  endcase

  return

end


;*************************************************************************
;
; L1_FMTREPORT - Create the Display Data dialog box
;
;*************************************************************************

PRO L1_FMTREPORT, GROUP_LEADER=group_leader
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON TCAD_FORMAT_REPORT_INFO, display_options, time_format, printer

  ;Default our menu settings.
  time_format = "yes"            ;Default to time conversion.
  printer =0                     ;Default to first printer.

  ; Log into the database, if we aren't already
  if not !dbi_connected then DB_CONNECT, "gpb","gravitydb", server="GPB_DATA"

  ; Initialize the printer information, if needed
  if N_ELEMENTS (printer_list) eq 0 then TCAD_INITIAL_PRINTER

  ; Define the structure used to store the panel object's data
  state = {base:                0L, $
           start_time_widget:   0L, $
           stop_time_widget:    0L, $
           timetype_widget:     0L, $
           timetype:             1, $
           changed:              0, $
           time_window:         0L, $
           cycle:                0, $
           start_sct:   NEW_SCT(), $
           stop_sct:    NEW_SCT(), $
           printer_droplist:    0L, $
           times_valid:          0 }

  display_state = PTR_NEW(state,/NO_COPY)        ;Make a common variable.

  ;Create and initialize the individual widgets, starting with the top-level
  ;base of the widget tree

  (*display_state).base = WIDGET_BASE ( TITLE="LEVEL 1 FORMAT REPORT", /COLUMN)

  if KEYWORD_SET (GROUP_LEADER) then begin
    WIDGET_CONTROL, (*display_state).base, GROUP_LEADER=group_leader
  endif

  ; Place the start and stop time input fields on the panel
  TCAD_INPUT_TIMES__DRAW, display_state

  base1 = WIDGET_BASE ((*display_state).base, /ROW)
  base2 = WIDGET_BASE (base1, /ROW, FRAME=2, /ALIGN_LEFT)
  base2a = WIDGET_BASE (base2, /COLUMN, /ALIGN_LEFT)
  display_options = [1,0,0]
  tmp = CW_BGROUP (base2a,["Display to Screen","Send to Printer", $
        "Write to File"], COLUMN=1, LABEL_TOP = "DISPLAY OPTIONS", $
        /RETURN_INDEX, /NONEXCLUSIVE, $
        UVALUE="DISPLAY_OPTIONS", SET_VALUE=display_options, /NO_RELEASE)

  base2b = WIDGET_BASE(base2, /COLUMN, /ALIGN_BOTTOM)
  tmp = WIDGET_DROPLIST(base2b, UVALUE='PRINTER', XSIZE=30, VALUE=printer_list)
  (*display_state).printer_droplist = tmp
```

```
      TCAD_DISPLAY__TIMETYPE, display_state
     end
   "DISPLAY_OPTIONS": begin              ;A display option was (un)checked
      display_options[event.value] = event.select      ;Record which one.
     end
   "TIME_FORMAT": begin                  ;Changed time format output.
      time_format = event.value          ;Record which radio button was clicked.
   end
   "PRINTER": begin                      ;A new printer was selected.
      printer = event.index              ;Get the list index of the printer.
   end
   "FILENAME": begin                     ;A filename was entered.
      filename = event.value
   end

  else:BEGIN
    TCAD_AUTOFILL_TIMES, event
   end

  endcase

  return

end


;*************************************************************************
;
; L1_FMTREPORT - Create the Display Data dialog box
;
;*************************************************************************

pro L1_FMTREPORT, GROUP_LEADER=group_leader
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON TCAD_FORMAT_REPORT_INFO, display_options, time_format, printer

  ;Default our menu settings.
  time_format = "yes"            ;Default to time conversion.
  printer =0                     ;Default to first printer.

  ; Log into the database, if we aren't already
  if not !dbi_connected then DB_CONNECT, "gpb","gravitydb", server="GPB_DATA"

  ; Initialize the printer information, if needed
  if N_ELEMENTS (printer_list) eq 0 then TCAD_INITIAL_PRINTER

  ; Define the structure used to store the panel object's data
  state = {base:                0L, $
           start_time_widget:   0L, $
           stop_time_widget:    0L, $
           timetype_widget:     0L, $
           timetype:             1, $
           changed:              0, $
           time_window:         0L, $
           cycle:                0, $
           start_sct:   NEW_SCT(), $
           stop_sct:    NEW_SCT(), $
           printer_droplist:    0L, $
           times_valid:          0 }

  display_state = PTR_NEW(state,/NO_COPY) ;Make a common variable.

  ;Create and initialize the individual widgets, starting with the top-level
  ;base of the widget tree

  (*display_state).base = WIDGET_BASE ( TITLE="LEVEL 1 FORMAT REPORT", /COLUMN)

  if KEYWORD_SET (GROUP_LEADER) then begin
    WIDGET_CONTROL, (*display_state).base, GROUP_LEADER=group_leader
  endif

  ; Place the start and stop time input fields on the panel
  TCAD_INPUT_TIMES__DRAW, display_state

  base1 = WIDGET_BASE ((*display_state).base, /ROW)
  base2 = WIDGET_BASE (base1, /ROW, FRAME=2, /ALIGN_LEFT)
  base2a = WIDGET_BASE (base2, /COLUMN, /ALIGN_LEFT)
  display_options = [1,0,0]
  tmp = CW_BGROUP (base2a,["Display to Screen","Send to Printer", $
        "Write to File"], COLUMN=1, LABEL_TOP = "DISPLAY OPTIONS", $
        /RETURN_INDEX, /NONEXCLUSIVE, $
        UVALUE="DISPLAY_OPTIONS", SET_VALUE=display_options, /NO_RELEASE)

  base2b = WIDGET_BASE(base2, /COLUMN, /ALIGN_BOTTOM)
  tmp = WIDGET_DROPLIST(base2b, UVALUE='PRINTER', XSIZE=30, VALUE=printer_list)
  (*display_state).printer_droplist = tmp
```


7

```
WIDGET_CONTROL, (*display_state).printer_droplist, SET_DROPLIST_SELECT=plot_printer
tmp = WIDGET_TEXT(base2b, /EDITABLE, UVALUE="FILENAME")

base3  =WIDGET_BASE (base1, /COLUMN )
base3a =WIDGET_BASE (base3, /ROW, FRAME=2)
tmp = CW_BGROUP(base3a, ["Yes","No"], LABEL_TOP = "VTCW time to timestamp", $
      /EXCLUSIVE, UVALUE="TIME_FORMAT", SET_VALUE=0, /RETURN_NAME, /NO_RELEASE)

base3b  =WIDGET_BASE (base3, /ROW, /ALIGN_BOTTOM)
tmp = WIDGET_LABEL(base3b, VALUE="                    ")
tmp = WIDGET_BUTTON (base3b, VALUE="    Okay    ", UVALUE="OKAY")
tmp = WIDGET_BUTTON (base3b, VALUE="   Cancel   ", UVALUE="CANCEL")

; Display the panel
WIDGET_CONTROL, (*display_state).base, SET_UVALUE=display_state
WIDGET_CONTROL, (*display_state).base, /REALIZE

;Hand off control of the dialog box to the XMANAGER
XMANAGER, 'L1_FMTREPORT', (*display_state).base

end
```


```
WIDGET_CONTROL, (*display_state).printer_droplist, SET_DROPLIST_SELECT=plot_printer
tmp = WIDGET_TEXT(base2b, /EDITABLE, UVALUE="FILENAME")

base3  =WIDGET_BASE (base1, /COLUMN )
base3a =WIDGET_BASE (base3, /ROW, FRAME=2)
tmp = CW_BGROUP(base3a, ["Yes","No"], LABEL_TOP = "VTCW time to timestamp", $
      /EXCLUSIVE, UVALUE="TIME_FORMAT", SET_VALUE=0, /RETURN_NAME, /NO_RELEASE)

base3b  =WIDGET_BASE (base3, /ROW, /ALIGN_BOTTOM)
tmp = WIDGET_LABEL(base3b, VALUE="                    ")
tmp = WIDGET_BUTTON (base3b, VALUE="    Okay    ", UVALUE="OKAY")
tmp = WIDGET_BUTTON (base3b, VALUE="   Cancel   ", UVALUE="CANCEL")

; Display the panel
WIDGET_CONTROL, (*display_state).base, SET_UVALUE=display_state
WIDGET_CONTROL, (*display_state).base, /REALIZE

;Hand off control of the dialog box to the XMANAGER
XMANAGER, 'L1_FMTREPORT', (*display_state).base

end
```

db_routines.cp

A#3

SU GP-B INSP 31

P100 FMT_INFO

```
296,299c296,297
<   exec sql declare c_p100_info cursor for
<     select SCT_VTCW, VCID, Format_ID from GPB_L0..Packets
<       where SCT_Cycle = :sct_cycle and SCT_VTCW between :start_vtcw and :end_vtcw
<       at isolation 0;
---
>   sprintf(sqlstmt,"select SCT_VTCW, VCID, Format_ID from GPB_L0..Packets");
>   sprintf(sqlstmt,"%s where SCT_Cycle = %d",sqlstmt,sct_cycle);
301c299,307
<   exec sql open c_p100_info row_count=75; /* Trial and error, do NOT change row_count. */
---
>   if ((start_vtcw | end_vtcw)) {      /*A time range was specified.*/
>      sprintf(sqlstmt,"%s and SCT_VTCW between %d and %d", sqlstmt, start_vtcw, end_vtcw);
>   }
>   strcat(sqlstmt," at isolation 0");
>
>   printf("  SEL: %s\n", sqlstmt);
>   exec sql prepare make_work from :sqlstmt;
>   exec sql declare c_p100_info cursor for make_work;
>   exec sql open c_p100_info;
355,356c361,362
<       memcpy (sct_vtcw_out, sct_vtcw,   (sizeof(long))*numr);
<       memcpy (vcid_out, vcid,           (sizeof(char))*numr);
---
>       memcpy (sct_vtcw_out, sct_vtcw, (sizeof(long))*numr);
>       memcpy (vcid_out, vcid,        (sizeof(char))*numr);
542c548
<   exec sql open c_evt row_count=55; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_evt row_count=60;
663c669
<   exec sql open c_tmd row_count=48; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_tmd row_count=48;
778c784
<   exec sql open c_snd row_count=48; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_snd row_count=48;
917c923
<   error, I've learned, do NOT change ROW_COUNT.  It is optimized for
---
>   error, I've learned, do NOT change row_count.  It is optimized for
923c929
<   exec sql open c_tma row_count=29; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_tma row_count=29;
1043c1049
<   error, I've learned, do NOT change ROW_COUNT.  It is optimized for
---
>   error, I've learned, do NOT change row_count.  It is optimized for
1049c1055
<   exec sql open c_sna row_count=29; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_sna row_count=29;
1169c1175
<   exec sql open c_tmv row_count=14; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_tmv row_count=10;
4219,4226c4225,4228
<     /*select P.SCT_Cycle, P.SCT_VTCW, P.Format_ID                                  */
<     /*from GPB_L0..Packets P, GPB_L0..Formats F                                    */
<     /*where P.SCT_Cycle = :sct_cycle and P.SCT_VTCW between :start_vtcw and :end_vtcw  */
<     /*        and P.Format_ID = F.Format_ID and F.Num_Frames = 100                 */
<             /*and P.Format_ID in (select Format_ID from GPB_L0..Formats where Num_Frames=100) */
<     select SCT_Cycle, SCT_VTCW, Format_ID from GPB_L0..Packets
<       where SCT_Cycle = :sct_cycle and SCT_VTCW between :start_vtcw and :end_vtcw
<               and Format_ID in (select Format_ID from GPB_L0..Formats where Num_Frames=100)
---
>     select P.SCT_Cycle, P.SCT_VTCW, P.Format_ID
>       from GPB_L0..Packets P, GPB_L0..Formats F
>       where P.SCT_Cycle = :sct_cycle and P.SCT_VTCW between :start_vtcw and :end_vtcw
>             and P.Format_ID = F.Format_ID and F.Num_Frames = 100
4229c4231
<   exec sql open c_sffmtid_tm row_count=63; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_sffmtid_tm row_count=10;
4272,4274c4274,4277
<     select SCT_Cycle, SCT_VTCW, Format_ID from GPB_L0..Packets
<       where SCT_Cycle = :sct_cycle and SCT_VTCW between :start_vtcw and :end_vtcw
<               and Format_ID in (select Format_ID from GPB_L0..Formats where Num_Frames<>100)
---
>     select P.SCT_Cycle, P.SCT_VTCW, P.Format_ID
>       from GPB_L0..Packets P, GPB_L0..Formats F
>       where P.SCT_Cycle = :sct_cycle and P.SCT_VTCW between :start_vtcw and :end_vtcw
>             and P.Format_ID = F.Format_ID and F.Num_Frames<>100
```

MCR 306

select 301 expect records

Comment only

306+ 307 Efficiency

N/C

Comment only

306 Format fetch

total first fetch

306+ 307

R/O cursor

MCR 306

total first fetch

W# 3

SU GP-B INSP 31

```
4277c4280
<   exec sql open c_sffmtid_sn row_count=63; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_sffmtid_sn row_count=10;
4433,4435c4436,4439
<     select SCT_Cycle, SCT_VTCW, Frame_Counter from GPB_L0..Packets
<       where SCT_Cycle = :sct_cycle and SCT_VTCW between :start_vtcw and :end_vtcw
<               and Format_ID in (select Format_ID from GPB_L0..Formats where Num_Frames=100)
---
>     select P.SCT_Cycle, P.SCT_VTCW, P.Frame_Counter
>       from GPB_L0..Packets P, GPB_L0..Formats F
>       where P.SCT_Cycle = :sct_cycle and P.SCT_VTCW between :start_vtcw and :end_vtcw
>             and P.Format_ID = F.Format_ID and F.Num_Frames = 100
4438c4442
<   exec sql open c_sfframecnt_tm row_count=63; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_sfframecnt_tm row_count=10;
4536,4538c4540,4543
<     select SCT_Cycle, SCT_VTCW, Frame_Counter from GPB_L0..Packets
<       where SCT_Cycle = :sct_cycle and SCT_VTCW between :start_vtcw and :end_vtcw
<               and Format_ID in (select Format_ID from GPB_L0..Formats where Num_Frames<>100)
---
>     select P.SCT_Cycle, P.SCT_VTCW, P.Frame_Counter
>       from GPB_L0..Packets P, GPB_L0..Formats F
>       where P.SCT_Cycle = :sct_cycle and P.SCT_VTCW between :start_vtcw and :end_vtcw
>             and P.Format_ID = F.Format_ID and F.Num_Frames<>100
4541c4546
<   exec sql open c_sfframecnt_sn row_count=63; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_sfframecnt_sn row_count=10;
4643c4648
<   exec sql open c_tlm_snapdata row_count=63; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_tlm_snapdata row_count=10;
4746c4751
<   exec sql open c_tlm_mrodata row_count=55; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_tlm_mrodata row_count=10;
4848c4853
<   exec sql open c_tlm_event_app_data row_count=63; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_tlm_event_app_data row_count=10;
4947c4952
<   exec sql open c_tlm_event_evt_data row_count=63; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_tlm_event_evt_data row_count=10;
5007a5013
>
5048c5054
<   exec sql open c_tlm_dbro_app_data row_count=3; /* Trial and error, do NOT change row_count. */
---
>   exec sql open c_tlm_dbro_app_data row_count=10;
5071a5078
>   unsigned char tmp;
5134,5205d5140
<
< /*************************************************************************/
< /* Routine for selecting TMID OCCURRENCEs from GPB_L1..TMdecom           */
< /*************************************************************************/
< long db_sel_tlmfmts (int argc, void *argv[])
< {
<   void error_handler ();
<   void warning_handler ();
<
<   short *tmid_in;
<
<   exec sql begin declare section;
<     CS_SMALLINT tmid;
<   exec sql end declare section;
<
<   tmid_in = (short *)argv[0];
<   tmid    = *tmid_in;
<
<   exec sql whenever SQLERROR   continue;
<   exec sql whenever SQLWARNING call warning_handler();
<   exec sql whenever NOT FOUND  continue;
<
<   exec sql close c_tlmfmts;
<
<   exec sql whenever SQLERROR   goto error;
<
<   exec sql declare c_tlmfmts cursor for
<       select L1d.Format_ID, L0f.Num_Frames, convert(binary(100),L1d.Word_Number)
<       from GPB_L1..TMdecom L1d, GPB_L0..Formats L0f
<       where L1d.TMID = :tmid and L1d.MSSID = (select MAX(MSSID) from GPB_L1..TMdecom)
```

306+ 307 R/O cursor

306 tmd fmt Tm

306+ 307 Efficiency R/O cursors

MCR 229 Format info for mnemonics New tmd fmt info

Att 3

SU GP-B INSP 31

MCR 229 (Cont)

```
<          and L1d.Format_ID = L0f.Format_ID and L0f.Num_Frames > 0   at isolation 0;
<   exec sql open c_tlmfmts row_count=3; /* Trial and error, do NOT change row_count. */
<   return(1);
< error:
<   error_handler ();
<   return(-1);
< }
<
< long db_get_tlmfmts (int argc, void *argv[])
< {
<   void   error_handler();
<   void   warning_handler();
<
<   char   *fmt_id_out;
<   char   *fmt_mf_out;
<   char   *mfbyte_out;
<
<   exec sql begin declare section;
<     CS_TINYINT  fmt_id;
<     CS_TINYINT  fmt_mf;
<     CS_BINARY   mfbyte[100];
<   exec sql end declare section;
<
<   exec sql whenever SQLERROR   goto error;
<   exec sql whenever SQLWARNING call warning_handler();
<   exec sql whenever NOT FOUND  continue;
<
<   fmt_id_out    = (char   *)argv[0];
<   fmt_mf_out    = (char   *)argv[1];
<   mfbyte_out    = (char   *)argv[2];
<
<   exec sql fetch norebind c_tlmfmts  into :fmt_id, :fmt_mf, :mfbyte;
<   if (sqlca.sqlcode == 100) return(0);
<
<   *fmt_id_out    = fmt_id;
<   *fmt_mf_out    = fmt_mf;
<   memcpy (mfbyte_out, mfbyte,100);
<   return(1);
< error:
<   error_handler ();
<   return(-1);
< }
```

```
;FILE:   snapshot.pro
;DATE:   10-05-03
;AUTHOR: Karl Stahl
;DESCRIPTION:  This program spawns a new window from the Analyze list and
;  generates a wrapper around a C script that retrieves snapshot data.
;  The resulting data can be either saved to a file, displayed to the screen,
;  or printed.

; Modified IDL code from Samantha Patterson
; Implements C code from Vladimir Solomonik

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PRO PROCESS_SNAPSHOT_FILE
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mode, sna
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tre_snap_
    GSSData, snap_id_id, do_all, did_output
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON FIRST_AND_LAST_SNAPSHOTS, first_snapshot_id, last_snapshot_id
  COMMON NODATA, found_no_data

  found_no_data = 0

  CASE sub_system OF
      0:  begin                  ; SRE
          if squid_number eq 0 then begin
              min = 1
              max = 4
          endif else begin
              min = squid_number
              max = squid_number
          endelse

          ;SquidData = PTRARR(1, max) ;pointer array since there are different numbers of snapsh   |

          ;nSnapshots = lonarr(1, max) ;array containing number of snapshots for each squid        |

          for squidnum = min, max do begin
              file = strcompress(GETENV("HOME") + '/temp.' + string(squidnum), /REMOVE_ALL)

              OPENR, lun, file, /GET_LUN
              OneSnapshot = {VehicleTime:   0L,$
                              Decimation:    0,$
                              Values:        intarr(1,4096)}

              num = 0L   ;set num to be a Long integer, so this is what gets read
              READU, lun, num
              if num eq 0 then CONTINUE ;no snapshots for this squid, so continue to next one

              nSnapshots[squidnum-1] = num ;num = number of Snapshots

              EachSquidData = REPLICATE(OneSnapshot, num)

              for i = 0, num-1 do begin
                  READU, lun, OneSnapshot
                  EachSquidData[i] = OneSnapshot
              endfor

              FREE_LUN, lun

              SquidData[squidnum-1] = ptr_new(EachSquidData)
                                  ;make a pointer to each of these (since they have varying length
                                  ;if they were all the same length could use 2D array of structur
                                  ;i.e. SquidData = replicate(OneSnapshot, 4, somelength)

              spawn, 'rm ' + file   ;clean up temporary files

          endfor

          if FIX(TOTAL(nSnapshots)) eq 0 then begin
              tmp = DIALOG_MESSAGE('No snapshots found.', /INFORMATION)
              ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
              ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
              found_no_data = 1
              return;     Procedures can't return values (only functions can do that).
          endif
      end

      1:  begin                  ; TRE

          file = strcompress(GETENV("HOME") + '/temp.' + string(tre_snap_type), /REMOVE_ALL)

          nSnapshots = 0L
          OPENR, lun, file, /GET_LUN
          READU, lun, nSnapshots

          if nSnapshots eq 0 then begin
```

```
;FILE:   snapshot.pro
;DATE:   10-05-03
;AUTHOR: Karl Stahl
;DESCRIPTION:  This program spawns a new window from the Analyze list and
;  generates a wrapper around a C script that retrieves snapshot data.
;  The resulting data can be either saved to a file, displayed to the screen,
;  or printed.

; Modified IDL code from Samantha Patterson
; Implements C code from Vladimir Solomonik

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
PRO PROCESS_SNAPSHOT_FILE
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mo
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tr
    GSSData, snap_id_id, do_all, did_output
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON FIRST_AND_LAST_SNAPSHOTS, first_snapshot_id, last_snapshot_id
  COMMON NODATA, found_no_data

  found_no_data = 0

  CASE sub_system OF
      0:  begin                  ; SRE
          if squid_number eq 0 then begin
              min = 1
              max = 4
          endif else begin
              min = squid_number
              max = squid_number
          endelse

          SquidData = PTRARR(1, max) ;pointer array since there are different numbers of

          nSnapshots = lonarr(1, max) ;array containing number of snapshots for each squi

          for squidnum = min, max do begin
              file = strcompress(GETENV("HOME") + '/temp.' + string(squidnum), /REMOVE_AL

              OPENR, lun, file, /GET_LUN
              OneSnapshot = {VehicleTime:   0L,$
                              Decimation:    0,$
                              Values:        intarr(1,4096)}

              num = 0L   ;set num to be a Long integer, so this is what gets read
              READU, lun, num
              if num eq 0 then CONTINUE ;no snapshots for this squid, so continue to next

              nSnapshots[squidnum-1] = num ;num = number of Snapshots

              EachSquidData = REPLICATE(OneSnapshot, num)

              for i = 0, num-1 do begin
                  READU, lun, OneSnapshot
                  EachSquidData[i] = OneSnapshot
              endfor

              FREE_LUN, lun

              SquidData[squidnum-1] = ptr_new(EachSquidData)
                                  ;make a pointer to each of these (since they have varying
                                  ;if they were all the same length could use 2D array of s
                                  ;i.e. SquidData = replicate(OneSnapshot, 4, somelength)

              spawn, 'rm ' + file   ;clean up temporary files

          endfor

          if FIX(TOTAL(nSnapshots)) eq 0 then begin
              tmp = DIALOG_MESSAGE('No snapshots found.', /INFORMATION)
              ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
              ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
              found_no_data = 1
              return;     Procedures can't return values (only functions can do that).
          endif
      end

      1:  begin                  ; TRE

          file = strcompress(GETENV("HOME") + '/temp.' + string(tre_snap_type), /REMOVE_A

          nSnapshots = 0L
          OPENR, lun, file, /GET_LUN
          READU, lun, nSnapshots

          if nSnapshots eq 0 then begin
```

MCR 312

12

```
        tmp = DIALOG_MESSAGE('No snapshots found', /INFORMATION)
        ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
        ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
        found_no_data = 1
        return
    endif

    SWITCH tre_snap_type OF
        1:
        2:
        3:  begin
            nPoints = 330
            data_type = 0
            nChannels = 8
        end
    ENDSWITCH

    SWITCH tre_snap_type OF
        4:
        5:  begin
            nPoints = 330
            data_type = 0
            nChannels = 1
        end
    ENDSWITCH

    SWITCH tre_snap_type OF
        6:  begin
            nPoints = 100
            data_type = 1
            nChannels = 8
        end
    ENDSWITCH

    n = nPoints * nChannels

    OneSnapshot = create_struct('VehicleTime',      0L,$
                                'Channel',          0)
    if data_type eq 0 then begin
        OneSnapshot = create_struct(OneSnapshot, 'AllColumnsData', INTARR(nChannels, nPoin
    endif else begin
        OneSnapshot = create_struct(OneSnapshot, 'AllColumnsData', LONARR(nChannels, nPoin
    endelse

    TREdata = REPLICATE(OneSnapshot, nSnapshots)

    ; read everything
    for i = 0, nSnapshots - 1 do begin
        READU, lun, OneSnapshot
        TREdata[i] = OneSnapshot
    endfor

    FREE_LUN, lun

    spawn, 'rm ' + file   ;clean up temporary files
end

2:  begin                 ; GSS

    CASE gss_snap_type OF
        1:  begin    ;suspension

            nSamples = 256

            OneSnapshot = {VehicleTime:              0L,$  ;L means it's a long integer (32
                           hDragFreeDataSource:      0L,$
                           hControlMode:             0L,$
                           hCommandsEnabled:         0L,$
                           hIOMode:                  0L,$
                           hStatus:                  0L,$
                           hFeedForwardMode:         0L,$
                           hDragFreeInjection:       0L,$
                           hDragFreeStatus:          0L,$
                           hEvent:                   0L,$
                           hSpare:                   0L,$
                           hControlPhase:            0L,$
                           h10HzIndex:               0L,$
                           hMultiplicityIndex:       0L,$
                           hMultiplicityCount:       0L,$
                           aAxisPosition:            FLTARR(1, nSamples),$
                           aAxisControlEffort:       FLTARR(1, nSamples),$
                           VaPlus:                   FLTARR(1, nSamples),$
                           VaMinus:                  FLTARR(1, nSamples),$
                           a_hat:                    FLTARR(1, nSamples),$
                           a_hat_dot:                FLTARR(1, nSamples),$
                           ControllerBandwidth:      FLTARR(1, nSamples),$
```

```
        tmp = DIALOG_MESSAGE('No snapshots found', /INFORMATION)
        ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
        ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
        found_no_data = 1
        return
    endif

    SWITCH tre_snap_type OF
        1:
        2:
        3:  begin
            nPoints = 330
            data_type = 0
            nChannels = 8
        end
    ENDSWITCH

    SWITCH tre_snap_type OF
        4:
        5:  begin
            nPoints = 330
            data_type = 0
            nChannels = 1
        end
    ENDSWITCH

    SWITCH tre_snap_type OF
        6:  begin
            nPoints = 100
            data_type = 1
            nChannels = 8
        end
    ENDSWITCH

    n = nPoints * nChannels

    OneSnapshot = create_struct('VehicleTime',      0L,$
                                'Channel',          0)
    if data_type eq 0 then begin
        OneSnapshot = create_struct(OneSnapshot, 'AllColumnsData', INTARR(nChannels
    endif else begin
        OneSnapshot = create_struct(OneSnapshot, 'AllColumnsData', LONARR(nChannels
    endelse

    TREdata = REPLICATE(OneSnapshot, nSnapshots)

    ; read everything
    for i = 0, nSnapshots - 1 do begin
        READU, lun, OneSnapshot
        TREdata[i] = OneSnapshot
    endfor

    FREE_LUN, lun

    spawn, 'rm ' + file   ;clean up temporary files
end

2:  begin                 ; GSS

    CASE gss_snap_type OF
        1:  begin    ;suspension

            nSamples = 256

            OneSnapshot = {VehicleTime:              0L,$  ;L means it's a long inte
                           hDragFreeDataSource:      0L,$
                           hControlMode:             0L,$
                           hCommandsEnabled:         0L,$
                           hIOMode:                  0L,$
                           hStatus:                  0L,$
                           hFeedForwardMode:         0L,$
                           hDragFreeInjection:       0L,$
                           hDragFreeStatus:          0L,$
                           hEvent:                   0L,$
                           hSpare:                   0L,$
                           hControlPhase:            0L,$
                           h10HzIndex:               0L,$
                           hMultiplicityIndex:       0L,$
                           hMultiplicityCount:       0L,$
                           aAxisPosition:            FLTARR(1, nSamples),$
                           aAxisControlEffort:       FLTARR(1, nSamples),$
                           VaPlus:                   FLTARR(1, nSamples),$
                           VaMinus:                  FLTARR(1, nSamples),$
                           a_hat:                    FLTARR(1, nSamples),$
                           a_hat_dot:                FLTARR(1, nSamples),$
                           ControllerBandwidth:      FLTARR(1, nSamples),$
```

D44

13

```
                bAxisPosition:          FLTARR(1, nSamples),$
                bAxisControlEffort:     FLTARR(1, nSamples),$
                VbPlus:                 FLTARR(1, nSamples),$
                VbMinus:                FLTARR(1, nSamples),$
                b_hat:                  FLTARR(1, nSamples),$
                b_hat_dot:              FLTARR(1, nSamples),$
                AftMuxIndex:            FLTARR(1, nSamples),$
                AftMuxValue:            FLTARR(1, nSamples),$
                cAxisPosition:          FLTARR(1, nSamples),$
                cAxisControlEffort:     FLTARR(1, nSamples),$
                VcPlus:                 FLTARR(1, nSamples),$
                VcMinus:                FLTARR(1, nSamples),$
                c_hat:                  FLTARR(1, nSamples),$
                c_hat_dot:              FLTARR(1, nSamples),$
                FwdMuxIndex:            FLTARR(1, nSamples),$
                FwdMuxValue:            FLTARR(1, nSamples)}
    end

    2:  begin    ;scheduler

        nSamples = 252

        OneSnapshot = {VehicleTime:             0L,$
                aPrimIdleTime:          FLTARR(1, nSamples),$
                aSecondIdleTime:        FLTARR(1, nSamples),$
                aTenHzIndex:            FLTARR(1, nSamples),$
                aFiveHzIndex:           FLTARR(1, nSamples),$
                aOneHzIndex:            FLTARR(1, nSamples),$
                aPrimTo10HzCounter:     FLTARR(1, nSamples),$
                aOneTo10HzCounter:      FLTARR(1, nSamples),$
                aTenHzSyncFailCounter:  FLTARR(1, nSamples),$
                aOneHzSyncFailCounter:  FLTARR(1, nSamples),$
                aDSPIntCounter:         FLTARR(1, nSamples),$
                bPrimIdleTime:          FLTARR(1, nSamples),$
                bSecondIdleTime:        FLTARR(1, nSamples),$
                bTenHzIndex:            FLTARR(1, nSamples),$
                bFiveHzIndex:           FLTARR(1, nSamples),$
                bOneHzIndex:            FLTARR(1, nSamples),$
                bPrimTo10HzCounter:     FLTARR(1, nSamples),$
                bOneto10HzCounter:      FLTARR(1, nSamples),$
                bTenHzSyncFailCounter:  FLTARR(1, nSamples),$
                bOneHzSyncFailCounter:  FLTARR(1, nSamples),$
                bDSPIntCounter:         FLTARR(1, nSamples),$
                cPrimIdleTime:          FLTARR(1, nSamples),$
                cSecondIdleTime:        FLTARR(1, nSamples),$
                cTenHzIndex:            FLTARR(1, nSamples),$
                cFiveHzIndex:           FLTARR(1, nSamples),$
                cOneHzIndex:            FLTARR(1, nSamples),$
                cPrimTo10HzCounter:     FLTARR(1, nSamples),$
                cOneto10HzCounter:      FLTARR(1, nSamples),$
                cTenHzSyncFailCounter:  FLTARR(1, nSamples),$
                cOneHzSyncFailCounter:  FLTARR(1, nSamples),$
                cDSPIntCounter:         FLTARR(1, nSamples)}
    end
ENDCASE

if squid_number eq 0 then begin
    min = 1
    max = 4
endif else begin
    min = squid_number
    max = squid_number
endelse

;GSSData = PTRARR(1, max) ;pointer array since there are different numbers of snapshot

;nSnapshots = lonarr(1, 4) ;array containing number of snapshots for each squid

for squidnum = min, max do begin
    file = strcompress(GETENV("HOME") + '/temp.' + string(squidnum), /REMOVE_ALL)

    OPENR, lun, file, /GET_LUN

    num = 0L
    READU, lun, num

    if num eq 0 then CONTINUE  ;no snapshots for this GSS, so continue to next one

    nSnapshots[squidnum-1] = num ;num = number of Snapshots

    EachGSSData = REPLICATE(OneSnapshot, num)

    for i = 0, num-1 do begin
        READU, lun, OneSnapshot
        EachGSSData[i] = OneSnapshot
    endfor
```

MCR 3/2

```
                bAxisPosition:          FLTARR(1, nSamples),$
                bAxisControlEffort:     FLTARR(1, nSamples),$
                VbPlus:                 FLTARR(1, nSamples),$
                VbMinus:                FLTARR(1, nSamples),$
                b_hat:                  FLTARR(1, nSamples),$
                b_hat_dot:              FLTARR(1, nSamples),$
                AftMuxIndex:            FLTARR(1, nSamples),$
                AftMuxValue:            FLTARR(1, nSamples),$
                cAxisPosition:          FLTARR(1, nSamples),$
                cAxisControlEffort:     FLTARR(1, nSamples),$
                VcPlus:                 FLTARR(1, nSamples),$
                VcMinus:                FLTARR(1, nSamples),$
                c_hat:                  FLTARR(1, nSamples),$
                c_hat_dot:              FLTARR(1, nSamples),$
                FwdMuxIndex:            FLTARR(1, nSamples),$
                FwdMuxValue:            FLTARR(1, nSamples)}
    end

    2:  begin    ;scheduler

        nSamples = 252

        OneSnapshot = {VehicleTime:             0L,$
                aPrimIdleTime:          FLTARR(1, nSamples),$
                aSecondIdleTime:        FLTARR(1, nSamples),$
                aTenHzIndex:            FLTARR(1, nSamples),$
                aFiveHzIndex:           FLTARR(1, nSamples),$
                aOneHzIndex:            FLTARR(1, nSamples),$
                aPrimTo10HzCounter:     FLTARR(1, nSamples),$
                aOneTo10HzCounter:      FLTARR(1, nSamples),$
                aTenHzSyncFailCounter:  FLTARR(1, nSamples),$
                aOneHzSyncFailCounter:  FLTARR(1, nSamples),$
                aDSPIntCounter:         FLTARR(1, nSamples),$
                bPrimIdleTime:          FLTARR(1, nSamples),$
                bSecondIdleTime:        FLTARR(1, nSamples),$
                bTenHzIndex:            FLTARR(1, nSamples),$
                bFiveHzIndex:           FLTARR(1, nSamples),$
                bOneHzIndex:            FLTARR(1, nSamples),$
                bPrimTo10HzCounter:     FLTARR(1, nSamples),$
                bOneto10HzCounter:      FLTARR(1, nSamples),$
                bTenHzSyncFailCounter:  FLTARR(1, nSamples),$
                bOneHzSyncFailCounter:  FLTARR(1, nSamples),$
                bDSPIntCounter:         FLTARR(1, nSamples),$
                cPrimIdleTime:          FLTARR(1, nSamples),$
                cSecondIdleTime:        FLTARR(1, nSamples),$
                cTenHzIndex:            FLTARR(1, nSamples),$
                cFiveHzIndex:           FLTARR(1, nSamples),$
                cOneHzIndex:            FLTARR(1, nSamples),$
                cPrimTo10HzCounter:     FLTARR(1, nSamples),$
                cOneto10HzCounter:      FLTARR(1, nSamples),$
                cTenHzSyncFailCounter:  FLTARR(1, nSamples),$
                cOneHzSyncFailCounter:  FLTARR(1, nSamples),$
                cDSPIntCounter:         FLTARR(1, nSamples)}
    end
ENDCASE

if squid_number eq 0 then begin
    min = 1
    max = 4
endif else begin
    min = squid_number
    max = squid_number
endelse

GSSData = PTRARR(1, max) ;pointer array since there are different numbers of sn

nSnapshots = lonarr(1, max) ;array containing number of snapshots for each squi

for squidnum = min, max do begin
    file = strcompress(GETENV("HOME") + '/temp.' + string(squidnum), /REMOVE_AL

    OPENR, lun, file, /GET_LUN

    num = 0L
    READU, lun, num

    if num eq 0 then CONTINUE  ;no snapshots for this GSS, so continue to next

    nSnapshots[squidnum-1] = num ;num = number of Snapshots

    EachGSSData = REPLICATE(OneSnapshot, num)

    for i = 0, num-1 do begin
        READU, lun, OneSnapshot
        EachGSSData[i] = OneSnapshot
    endfor
```

SU Gp-D MSP 31

14

```
            FREE_LUN, lun

            GSSData[squidnum-1] = ptr_new(EachGSSData)
                                 ;make a pointer to each of these (since they have varying length
                                 ;if they were all the same length could use 2D array of structur
                                 ;i.e. GSSData = replicate(OneSnapshot,
                                 ;4, somelength)

            spawn, 'rm ' + file   ;clean up temporary files

        endfor

        if FIX(TOTAL(nSnapshots)) eq 0 then begin
            tmp = DIALOG_MESSAGE('No snapshots found.', /INFORMATION)
            ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
            ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
            found_no_data = 1
            return
        endif
     end
  ENDCASE
END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PRO SNAPSHOT_SUBSYSTEM_COLUMN

  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mode, sna
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tre_snap_
    GSSData, snap_id_id, do_all, did_output
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON WIDGET_IDS, subsystem_id, squid_number_id, sre_snap_type_id, squid_mode_id, tre_snap_ty

  WIDGET_CONTROL, base4, /DESTROY

  CASE sub_system OF
      0:  begin   ; SRE
          base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2) ;Snapshot Options Column

          base4a = WIDGET_BASE (base4, /ROW)
          tmp = WIDGET_LABEL(base4a, VALUE = 'SQUID Number')
          squid_number = 0    ;default value for squid_number
          squid_number_id= WIDGET_DROPLIST(base4a, UVALUE = 'SQUIDNUMBER', VALUE = ['0 (All)','1

          base4b = WIDGET_BASE(base4, /ROW)
          tmp = WIDGET_LABEL(base4b, VALUE = 'Snapshot Type')
          sre_snap_type = 20      ;default value for sre_snap_type
          sre_snap_type_id = CW_BGROUP(base4b, ['20 Hz', '1 kHz'], /EXCLUSIVE, UVALUE = 'SRE_SNA

          base4c = WIDGET_BASE(base4, /ROW)
          tmp = WIDGET_LABEL(base4c, VALUE = 'SQUID Mode')
          squid_mode = 1      ;default value for squid_mode
          squid_mode_id = CW_BGROUP(base4c, ['Default','Locked'], /EXCLUSIVE, UVALUE = 'SQUIDMOD
      end

      1:  begin   ; TRE
          base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2) ;Snapshot Options Column

          base4a = WIDGET_BASE (base4, /ROW, /ALIGN_LEFT)
          tmp = WIDGET_LABEL(base4a, VALUE = 'Snapshot Type')
          tre_snap_type = 1
          tre_snap_type_id = WIDGET_DROPLIST(base4a, UVALUE = 'TRE_SNAPTYPE', VALUE = ['1', '2',
      end

      2:  begin   ; GSS
          base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2) ;Snapshot Options Column

          base4a = WIDGET_BASE (base4, /ROW, /ALIGN_LEFT)
          tmp = WIDGET_LABEL(base4a, VALUE = 'Snapshot Type')
          gss_snap_type = 1
          gss_snap_type_id = CW_BGROUP(base4a, ['Suspension', 'Scheduler'], /EXCLUSIVE, UVALUE =

          base4b = WIDGET_BASE (base4, /ROW, /ALIGN_LEFT)
          tmp = WIDGET_LABEL(base4b, VALUE = 'GSS Number: ')
          squid_number = 0  ;Note: instead of defining a "gss_number", I'll just use squid_numbe

          ; NB: Used the same code for gss number as for squidnumber (originally defined for SRE
          ; are identical and it would be superfluous to repeat the code.
          squid_number_id = WIDGET_DROPLIST(base4b, UVALUE = 'SQUIDNUMBER', VALUE = ['0 (All)',
      end

;      3:  begin   ; Other
;          base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2) ;Snapshot Options Column

;          base4a = WIDGET_BASE (base4, /ROW, /ALIGN_LEFT)
```

```
            FREE_LUN, lun

            GSSData[squidnum-1] = ptr_new(EachGSSData)
                                 ;make a pointer to each of these (since they have varying
                                 ;if they were all the same length could use 2D array of s
                                 ;i.e. GSSData = replicate(OneSnapshot,
                                 ;4, somelength)

            spawn, 'rm ' + file   ;clean up temporary files

        endfor

        if FIX(TOTAL(nSnapshots)) eq 0 then begin
            tmp = DIALOG_MESSAGE('No snapshots found.', /INFORMATION)
            ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
            ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
            found_no_data = 1
            return
        endif
     end
  ENDCASE
END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PRO SNAPSHOT_SUBSYSTEM_COLUMN

  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mo
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tr
    GSSData, snap_id_id, do_all, did_output
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON WIDGET_IDS, subsystem_id, squid_number_id, sre_snap_type_id, squid_mode_id, tre_

  WIDGET_CONTROL, base4, /DESTROY

  CASE sub_system OF
      0:  begin   ; SRE
          base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2) ;Snapshot Options

          base4a = WIDGET_BASE (base4, /ROW)
          tmp = WIDGET_LABEL(base4a, VALUE = 'SQUID Number')
          squid_number = 0    ;default value for squid_number
          squid_number_id= WIDGET_DROPLIST(base4a, UVALUE = 'SQUIDNUMBER', VALUE = ['0 (A

          base4b = WIDGET_BASE(base4, /ROW)
          tmp = WIDGET_LABEL(base4b, VALUE = 'Snapshot Type')
          sre_snap_type = 20      ;default value for sre_snap_type
          sre_snap_type_id = CW_BGROUP(base4b, ['20 Hz', '1 kHz'], /EXCLUSIVE, UVALUE = '

          base4c = WIDGET_BASE(base4, /ROW)
          tmp = WIDGET_LABEL(base4c, VALUE = 'SQUID Mode')
          squid_mode = 1      ;default value for squid_mode
          squid_mode_id = CW_BGROUP(base4c, ['Default','Locked'], /EXCLUSIVE, UVALUE = 'S
      end

      1:  begin   ; TRE
          base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2) ;Snapshot Options

          base4a = WIDGET_BASE (base4, /ROW, /ALIGN_LEFT)
          tmp = WIDGET_LABEL(base4a, VALUE = 'Snapshot Type')
          tre_snap_type = 1
          tre_snap_type_id = WIDGET_DROPLIST(base4a, UVALUE = 'TRE_SNAPTYPE', VALUE = ['1
      end

      2:  begin   ; GSS
          base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2) ;Snapshot Options

          base4a = WIDGET_BASE (base4, /ROW, /ALIGN_LEFT)
          tmp = WIDGET_LABEL(base4a, VALUE = 'Snapshot Type')
          gss_snap_type = 1
          gss_snap_type_id = CW_BGROUP(base4a, ['Suspension', 'Scheduler'], /EXCLUSIVE, U

          base4b = WIDGET_BASE (base4, /ROW, /ALIGN_LEFT)
          tmp = WIDGET_LABEL(base4b, VALUE = 'GSS Number: ')
          squid_number = 0  ;Note: instead of defining a "gss_number", I'll just use squi

          ; NB: Used the same code for gss number as for squidnumber (originally defined
          ; are identical and it would be superfluous to repeat the code.
          squid_number_id = WIDGET_DROPLIST(base4b, UVALUE = 'SQUIDNUMBER', VALUE = ['0 (
      end

;      3:  begin   ; Other
;          base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2) ;Snapshot Options

;          base4a = WIDGET_BASE (base4, /ROW, /ALIGN_LEFT)
```

SU GP-B
INSP 31

15

```
;           tmp = WIDGET_LABEL(base4a, VALUE = 'OTHER STUFF HERE')

;           base4b = WIDGET_BASE(base4, /ROW, /ALIGN_LEFT)
;           tmp = WIDGET_LABEL(base4b, VALUE = 'Snapshot ID: ')
;           tmp = WIDGET_DROPLIST(base4b, UVALUE = 'SNAPID', VALUE = ['136', '137', '138', '139',

;           base4c = WIDGET_BASE(base4, /ROW, /ALIGN_CENTER)
;           tmp = WIDGET_BUTTON(base4c, VALUE = 'Format...', UVALUE = 'FORMAT')
;       end
  ENDCASE
END


FUNCTION ARRAY2STR, array
  s = string(array[0])

  x = fix(max(size(array,/dimension)))  ; note that you cannot call another function from within
                                        ; function (e.g. cannot use 'length' function here)

  for i = 1,x[0]-1 do begin
      s = s + string(array[i])
  endfor

  return, s
END


FUNCTION LENGTH, array   ; find the length of a vector
  return, fix(max(size(array,/dimension))) ;  'max' gets biggest dimension, 'fix' converts to in
END


FUNCTION ISINARRAY, number, array
  for i = 0, length(array)-1 do begin
      if array[i] eq number then return, 1
  endfor
  return, 0
END

FUNCTION GETNUMEXP, number
  i = 0
  ; DAMN! Stupid, stupid, stupid IDL assumes that "number" (which is passed as a parameter!)
  ; is actually a pointer.  This means that if I say number/10, the value to which
  ; "number" points gets altered!!!  So I have to create a new variable.
  num = number
  num = num/10
  while num gt 1 do begin
      num = num/10
      i = i+1
  endwhile
  return, i
END

FUNCTION REMOVE_INF, a, b  ;a = t, b = y

  ; inefficient but functional
  L = length(a)
  for i = 0, L-1 do begin
      if (not FINITE(a[i])) or (not FINITE(b[i])) then begin
          for j = i, L-2 do begin  ;shift up
              a[i] = a[i+1]
              b[i] = b[i+1]
          endfor
          L = L-1
      endif
  endfor
  tmp = {t: a[0:L-1], y: b[0:L-1]}
  return, tmp

END


FUNCTION GET_LOG_LOG_PLOT_DATA, x_min, x_max, minor_ticks_per_major_tick, add_space

;  minor_ticks_per_major_tick = 5

  m = 0
  interval = double(10-1)/double(minor_ticks_per_major_tick)
  tick = DBLARR(1, minor_ticks_per_major_tick*interval + 1)
  for i = 1, minor_ticks_per_major_tick do begin
      tick[m] = ALOG10(i*interval + 1)
```


```
;           tmp = WIDGET_LABEL(base4a, VALUE = 'OTHER STUFF HERE')

;           base4b = WIDGET_BASE(base4, /ROW, /ALIGN_LEFT)
;           tmp = WIDGET_LABEL(base4b, VALUE = 'Snapshot ID: ')
;           tmp = WIDGET_DROPLIST(base4b, UVALUE = 'SNAPID', VALUE = ['136', '137', '138',

;           base4c = WIDGET_BASE(base4, /ROW, /ALIGN_CENTER)
;           tmp = WIDGET_BUTTON(base4c, VALUE = 'Format...', UVALUE = 'FORMAT')
;       end
  ENDCASE
END


FUNCTION ARRAY2STR, array
  s = string(array[0])

  x = fix(max(size(array,/dimension)))  ; note that you cannot call another function from
                                        ; function (e.g. cannot use 'length' function her

  for i = 1,x[0]-1 do begin
      s = s + string(array[i])
  endfor

  return, s
END


FUNCTION LENGTH, array   ; find the length of a vector
  return, fix(max(size(array,/dimension))) ;  'max' gets biggest dimension, 'fix' convert
END


FUNCTION ISINARRAY, number, array
  for i = 0, length(array)-1 do begin
      if array[i] eq number then return, 1
  endfor
  return, 0
END

FUNCTION GETNUMEXP, number
  i = 0
  ; DAMN! Stupid, stupid, stupid IDL assumes that "number" (which is passed as a paramete
  ; is actually a pointer.  This means that if I say number/10, the value to which
  ; "number" points gets altered!!!  So I have to create a new variable.
  num = number
  num = num/10
  while num gt 1 do begin
      num = num/10
      i = i+1
  endwhile
  return, i
END

FUNCTION REMOVE_INF, a, b  ;a = t, b = y

  ; inefficient but functional
  L = length(a)
  for i = 0, L-1 do begin
      if (not FINITE(a[i])) or (not FINITE(b[i])) then begin
          for j = i, L-2 do begin  ;shift up
              a[i] = a[i+1]
              b[i] = b[i+1]
          endfor
          L = L-1
      endif
  endfor
  tmp = {t: a[0:L-1], y: b[0:L-1]}
  return, tmp

END


FUNCTION GET_LOG_LOG_PLOT_DATA, x_min, x_max, minor_ticks_per_major_tick, add_space

;  minor_ticks_per_major_tick = 5

  m = 0
  interval = double(10-1)/double(minor_ticks_per_major_tick)
  tick = DBLARR(1, minor_ticks_per_major_tick*interval + 1)
  for i = 1, minor_ticks_per_major_tick do begin
      tick[m] = ALOG10(i*interval + 1)
```

```
    m = m+1
  endfor

  k = 0
  axis_tick_font = OBJ_NEW('IDLgrFont', 'Times', size = 10)
  major = x_max-x_min
  minor = major*minor_ticks_per_major_tick + 1
  tick_text = OBJARR(1, (major + minor))
  tick_location = DBLARR(1, (major + minor))
  for i = x_min, x_max do begin

      tick_location[k] = i
      tmp = '10!U' + strtrim(i, 2)
      if add_space then tmp = tmp + ' '
      tick_text[k] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, /ENABLE_FORMATTING)

      if i ne x_max then begin
          m = k
          for j = 0, minor_ticks_per_major_tick-1 do begin
              tick_location[m + j + 1] = i + tick[j]
              tick_text[m + j + 1] = OBJ_NEW('IDLgrText') ;Blank
              k = k+1
          endfor
      endif

      k = k+1
  end

  tmp = {tick_location: tick_location ,$
         tick_text: tick_text         ,$
         num_ticks: k                 }

;;;; Here's a nice, clear way of viewing things: You provide major tick locations and their
;;;; corresponding values, then you provide minor tick locations and their corresponding values.
;;;; Alas, that is not the IDL way.  This is my nice code that uses that mistaken notion.
;   for i = x_min, x_max do begin

;       major_tick_location[k] = i
;       tmp = '10!U' + strtrim(i, 2)
;       major_tick_text[k] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, /ENABLE_FORMATTING

;       for j = 0, minor_ticks_per_major_tick-1 do begin
;           minor_tick_location[j + k*minor_ticks_per_major_tick] = i + tick[j]
;       endfor

;       k = k+1
;   end

;   tmp = {major_tick_location: major_tick_location ,$
;          major_tick_text: major_tick_text         ,$
;          minor_tick_location: minor_tick_location }

  return, tmp
END


FUNCTION MAKE_FFT_VIEW, t, y

  fft_array = CALCULATE_FFT(t/1000, y)

  t = ALOG10(fft_array.x)
  y = ALOG10(fft_array.y)

  tmp = REMOVE_INF(t, y)
  t = tmp.t
  y = tmp.y

;  data_array = DBLARR(4, length(y))
;  data_array = [TRANSPOSE(t), TRANSPOSE(y), TRANSPOSE(fft_array.x), TRANSPOSE(fft_array.y)]
;  openw, lun, 'log_data.dat', /get_lun
;  printf, lun, data_array, format = '(d14.6, d14.1, d14.1, d14.1)'
;  free_lun, lun

  max_t = CEIL(MAX(t))
  min_t = FLOOR(MIN(t))
  max_y = CEIL(MAX(y))
  min_y = FLOOR(MIN(y))
  llx = min_t
  lly = min_y
  width = double(max_t) - double(min_t)
  height = double(max_y) - double(min_y)
  x_margin = .13*width
  y_margin = .2*height

  plot = OBJ_NEW('IDLgrPlot', t, y, COLOR = [0, 0, 255])
```

```
    m = m+1
  endfor

  k = 0
  axis_tick_font = OBJ_NEW('IDLgrFont', 'Times', size = 10)
  major = x_max-x_min
  minor = major*minor_ticks_per_major_tick + 1
  tick_text = OBJARR(1, (major + minor))
  tick_location = DBLARR(1, (major + minor))
  for i = x_min, x_max do begin

      tick_location[k] = i
      tmp = '10!U' + strtrim(i, 2)
      if add_space then tmp = tmp + ' '
      tick_text[k] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, /ENABLE_FORMATTING)

      if i ne x_max then begin
          m = k
          for j = 0, minor_ticks_per_major_tick-1 do begin
              tick_location[m + j + 1] = i + tick[j]
              tick_text[m + j + 1] = OBJ_NEW('IDLgrText') ;Blank
              k = k+1
          endfor
      endif

      k = k+1
  end

  tmp = {tick_location: tick_location ,$
         tick_text: tick_text         ,$
         num_ticks: k                 }

;;;; Here's a nice, clear way of viewing things: You provide major tick locations and the
;;;; corresponding values, then you provide minor tick locations and their corresponding
;;;; Alas, that is not the IDL way.  This is my nice code that uses that mistaken notion.
;   for i = x_min, x_max do begin

;       major_tick_location[k] = i
;       tmp = '10!U' + strtrim(i, 2)
;       major_tick_text[k] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, /ENABLE_FOR

;       for j = 0, minor_ticks_per_major_tick-1 do begin
;           minor_tick_location[j + k*minor_ticks_per_major_tick] = i + tick[j]
;       endfor

;       k = k+1
;   end

;   tmp = {major_tick_location: major_tick_location ,$
;          major_tick_text: major_tick_text         ,$
;          minor_tick_location: minor_tick_location }

  return, tmp
END


FUNCTION MAKE_FFT_VIEW, t, y

  fft_array = CALCULATE_FFT(t/1000, y)

  t = ALOG10(fft_array.x)
  y = ALOG10(fft_array.y)

  tmp = REMOVE_INF(t, y)
  t = tmp.t
  y = tmp.y

;  data_array = DBLARR(4, length(y))
;  data_array = [TRANSPOSE(t), TRANSPOSE(y), TRANSPOSE(fft_array.x), TRANSPOSE(fft_array.
;  openw, lun, 'log_data.dat', /get_lun
;  printf, lun, data_array, format = '(d14.6, d14.1, d14.1, d14.1)'
;  free_lun, lun

  max_t = CEIL(MAX(t))
  min_t = FLOOR(MIN(t))
  max_y = CEIL(MAX(y))
  min_y = FLOOR(MIN(y))
  llx = min_t
  lly = min_y
  width = double(max_t) - double(min_t)
  height = double(max_y) - double(min_y)
  x_margin = .13*width
  y_margin = .2*height

  plot = OBJ_NEW('IDLgrPlot', t, y, COLOR = [0, 0, 255])
```


17

```
  view = OBJ_NEW('IDLgrView', UNITS = 3, $
                 VIEWPLANE_RECT = [llx - 1.1*x_margin, lly - y_margin, width + 2*x_margin, heigh
                 LOCATION = [0.0, 0.06], dimensions = [1.0, 0.47], $
                 COLOR = [255, 255, 255])

  title = 'FFT'
  title_font = OBJ_NEW('IDLgrFont', 'Times*Bold', size = 14)
  plot_title = OBJ_NEW('IDLgrText', title, LOCATION = [.25*width, 1.1*max_y], $
                       FONT = title_font)



  log_log_t = GET_LOG_LOG_PLOT_DATA(min_t, max_t, 6, 0)
  log_log_y = GET_LOG_LOG_PLOT_DATA(min_y, max_y, 5, 1)



  axis_title_font = OBJ_NEW('IDLgrFont', 'Times*Italic', size = 12)
  x_axis_title = OBJ_NEW('IDLgrText', 'Hz', FONT = axis_title_font)

  x_axis = OBJ_NEW('IDLgrAxis', 0, RANGE = [min_t, max_t], LOCATION = [min_t, min_y], /EXACT, $
                   MAJOR = log_log_t.num_ticks, TICKVALUES = log_log_t.tick_location, $
                   TICKTEXT = log_log_t.tick_text, $
                   TICKLEN = .03*height, TITLE = x_axis_title)

  y_axis = OBJ_NEW('IDLgrAxis', 1, RANGE = [min_y, max_y], LOCATION = [min_t, min_y], /EXACT, $
                   MAJOR = log_log_y.num_ticks, TICKVALUES = log_log_y.tick_location, $
                  TICKTEXT = log_log_y.tick_text, $
                   TICKLEN = .01*width)

                 ;  MAJOR = num_ticks, TICKLEN = .01*width, TEXTALIGNMENTS = [1, 0.5], $
                 ;  TICKVALUES = y_ticks_location, TICKTEXT = y_ticks_vector, /EXACT)

  divider_line = OBJ_NEW('IDLgrPolyline', [llx-2*x_margin, llx + width + x_margin], $
                         [lly - y_margin, lly - y_margin]);, THICK = 2)


  divider_line2 = OBJ_NEW('IDLgrPolyline', [llx-2*x_margin, llx + width + x_margin], $
                          [lly - 0.94*y_margin, lly - 0.94*y_margin])

;  array2print = [TRANSPOSE(fft_array.x), TRANSPOSE(fft_array.y)]
;  print, array2print[*, 1:100]

;plot, fft_array.x, fft_array.y, /XLOG, /YLOG  ;, XRANGE = [1/10^2, 10^4], YRANGE = [1/10^4, 10^

  model = OBJ_NEW('IDLgrModel')
  model -> ADD, [plot, x_axis, y_axis, plot_title, divider_line, divider_line2] ;, divider_line3
  view -> ADD, model
  return, view
END
```

```
  view = OBJ_NEW('IDLgrView', UNITS = 3, $
                 VIEWPLANE_RECT = [llx - 1.1*x_margin, lly - y_margin, width + 2*x_margin
                 LOCATION = [0.0, 0.06], dimensions = [1.0, 0.47], $
                 COLOR = [255, 255, 255])

  title = 'FFT'
  title_font = OBJ_NEW('IDLgrFont', 'Times*Bold', size = 14)
  plot_title = OBJ_NEW('IDLgrText', title, LOCATION = [.25*width, 1.1*max_y], $
                       FONT = title_font)



  log_log_t = GET_LOG_LOG_PLOT_DATA(min_t, max_t, 6, 0)
  log_log_y = GET_LOG_LOG_PLOT_DATA(min_y, max_y, 5, 1)



  axis_title_font = OBJ_NEW('IDLgrFont', 'Times*Italic', size = 12)
  x_axis_title = OBJ_NEW('IDLgrText', 'Hz', FONT = axis_title_font)

  x_axis = OBJ_NEW('IDLgrAxis', 0, RANGE = [min_t, max_t], LOCATION = [min_t, min_y], /EX
                   MAJOR = log_log_t.num_ticks, TICKVALUES = log_log_t.tick_location, $
                   TICKTEXT = log_log_t.tick_text, $
                   TICKLEN = .03*height, TITLE = x_axis_title)

  y_axis = OBJ_NEW('IDLgrAxis', 1, RANGE = [min_y, max_y], LOCATION = [min_t, min_y], /EX
                   MAJOR = log_log_y.num_ticks, TICKVALUES = log_log_y.tick_location, $
                  TICKTEXT = log_log_y.tick_text, $
                   TICKLEN = .01*width)

                 ;  MAJOR = num_ticks, TICKLEN = .01*width, TEXTALIGNMENTS = [1, 0.5], $
                 ;  TICKVALUES = y_ticks_location, TICKTEXT = y_ticks_vector, /EXACT)

  divider_line = OBJ_NEW('IDLgrPolyline', [llx-2*x_margin, llx + width + x_margin], $
                         [lly - y_margin, lly - y_margin]);, THICK = 2)


  divider_line2 = OBJ_NEW('IDLgrPolyline', [llx-2*x_margin, llx + width + x_margin], $
                          [lly - 0.94*y_margin, lly - 0.94*y_margin])

;  array2print = [TRANSPOSE(fft_array.x), TRANSPOSE(fft_array.y)]
;  print, array2print[*, 1:100]

;plot, fft_array.x, fft_array.y, /XLOG, /YLOG  ;, XRANGE = [1/10^2, 10^4], YRANGE = [1/10

  model = OBJ_NEW('IDLgrModel')
  model -> ADD, [plot, x_axis, y_axis, plot_title, divider_line, divider_line2] ;, divide
  view -> ADD, model
  return, view
END
```

```
FUNCTION MAKE_SRE_VIEW, t, y ;SquidData, squid, snap

  plot = OBJ_NEW('IDLgrPlot', t, y, COLOR = [255, 0, 0])
  model = OBJ_NEW('IDLgrModel')
  axis_title_font = OBJ_NEW('IDLgrFont', 'Times*Italic', size = 12)
  axis_tick_font = OBJ_NEW('IDLgrFont', 'Times', size = 10)
  llx = MIN(t)
  lly = MIN(y)
  max_t = MAX(t)
  min_t = MIN(t)
  max_y = MAX(y)
  min_y = MIN(y)

  width = double(max_t) - double(min_t)
  height = double(max_y) - double(min_y)
  x_margin = .13*width
  y_margin = .2*height

;  title = 'SRE Snapshot'
;  plot_title = OBJ_NEW('IDLgrText', title)
;  title_font = OBJ_NEW('IDLgrFont', 'Times*Bold', size = 12)
;  plot_title -> SetProperty, font = title_font
;  plot_title -> SetProperty, LOCATION = [.35*width, 1.2*max_y]

  x = DBLARR(1, 100)
```

```
FUNCTION MAKE_SRE_VIEW, t, y ;SquidData, squid, snap

  plot = OBJ_NEW('IDLgrPlot', t, y, COLOR = [255, 0, 0])
  model = OBJ_NEW('IDLgrModel')
  axis_title_font = OBJ_NEW('IDLgrFont', 'Times*Italic', size = 12)
  axis_tick_font = OBJ_NEW('IDLgrFont', 'Times', size = 10)
  llx = MIN(t)
  lly = MIN(y)
  max_t = MAX(t)
  min_t = MIN(t)
  max_y = MAX(y)
  min_y = MIN(y)

  width = double(max_t) - double(min_t)
  height = double(max_y) - double(min_y)
  x_margin = .13*width
  y_margin = .2*height

;  title = 'SRE Snapshot'
;  plot_title = OBJ_NEW('IDLgrText', title)
;  title_font = OBJ_NEW('IDLgrFont', 'Times*Bold', size = 12)
;  plot_title -> SetProperty, font = title_font
;  plot_title -> SetProperty, LOCATION = [.35*width, 1.2*max_y]

  x = DBLARR(1, 100)
```

18

```
  i = 0
  t_step = 500
  x[0] = t_step
  while x[i] lt max_t do begin
      x[i + 1] = x[i] + t_step
      i = i+1
  endwhile
  num_ticks = i

  x_ticks = DBLARR(1, num_ticks)
  for i = 0, num_ticks-1 do x_ticks[i] = x[i]
  x_axis_title = OBJ_NEW('IDLgrText', 'msec', FONT = axis_title_font)
  x_axis = OBJ_NEW('IDLgrAxis', 0, RANGE = [min_t, max_t], LOCATION = [min_t, min_y], $
                   MAJOR = num_ticks, TICKVALUES = x_ticks, $
                   TICKLEN = .01*height, TITLE = x_axis_title, /EXACT)
  x_axis -> GetProperty, TICKTEXT = x_ticktext     ; FONT keyword not allowed in IDLgrAxis objec
  x_ticktext -> SetProperty, FONT = axis_tick_font ; the only way to change the font of the tick

  y_location = DBLARR(1,100)
  i = 0
  exponent = GetNumExp(max_y)
  y_step = 10^exponent
  y_location[0] = min_y
  while y_location[i] lt max_y do begin
      y_location[i + 1] = y_location[i] + y_step
      i = i+1
  endwhile
  num_ticks = i
  y_ticks_location = DBLARR(1, num_ticks)
  y_ticks_vector = OBJARR(1, num_ticks)
  for i = 0, num_ticks-1 do begin
      y_ticks_location[i] = y_location[i]
      tmp = strtrim(string(y_location[i]/y_step, FORMAT = '(d10.1)'), 2) + ' '
      y_ticks_vector[i] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font)
  endfor
  y_axis = OBJ_NEW('IDLgrAxis', 1, RANGE = [min_y, max_y], LOCATION = [min_t, min_y], $
                   MAJOR = num_ticks, TICKLEN = .01*width, TEXTALIGNMENTS = [1, 0.5], $
                   TICKVALUES = y_ticks_location, TICKTEXT = y_ticks_vector, /EXACT)

  tmp = 'x10!U' + strtrim(exponent, 2)
  exponent_text = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, /ENABLE_FORMATTING, $
                          LOCATION = [min_t, 1.05*max_y])

  divider_line = OBJ_NEW('IDLgrPolyline', [llx-2*x_margin, llx + width + x_margin], $
                         [lly - .96*y_margin, lly - .96*y_margin]);, THICK = 2)


  data_view = OBJ_NEW('IDLgrView', UNITS = 3, $
                VIEWPLANE_RECT = [llx - 1.1*x_margin, lly - y_margin, width + 2*x_margin, heigh
                LOCATION = [0.0, 0.53], dimensions = [1.0, 0.47], $
                COLOR = [255, 255, 255])

  model -> ADD, [plot, x_axis, y_axis, exponent_text, divider_line]
  data_view -> ADD, model
  return, data_view

END


FUNCTION MAKE_TRE_VIEW, t, y, column

  if column gt 3 then color = [255, 0, 0] else color = [255, 0, 255]

  plot = OBJ_NEW('IDLgrPlot', t, y, COLOR = color)
  model = OBJ_NEW('IDLgrModel')
  axis_title_font = OBJ_NEW('IDLgrFont', 'Times*Italic', size = 10)
  axis_tick_font = OBJ_NEW('IDLgrFont', 'Times', size = 10)

  max_t = MAX(t)
  min_t = MIN(t)
  max_y = MAX(y)
  min_y = MIN(y)
  llx = min_t
  lly = min_y
  width = double(max_t) - double(min_t)
  height = double(max_y) - double(min_y)
  x_margin = .1*width
  y_margin = .3*height

  case column of
      0: title = 'Channel: X + A'
      1: title = 'Channel: X - A'
      2: title = 'Channel: Y + A'
      3: title = 'Channel: Y - A'
      4: title = 'Channel: X + B'
```

```
  i = 0
  t_step = 500
  x[0] = t_step
  while x[i] lt max_t do begin
      x[i + 1] = x[i] + t_step
      i = i+1
  endwhile
  num_ticks = i

  x_ticks = DBLARR(1, num_ticks)
  for i = 0, num_ticks-1 do x_ticks[i] = x[i]
  x_axis_title = OBJ_NEW('IDLgrText', 'msec', FONT = axis_title_font)
  x_axis = OBJ_NEW('IDLgrAxis', 0, RANGE = [min_t, max_t], LOCATION = [min_t, min_y], $
                   MAJOR = num_ticks, TICKVALUES = x_ticks, $
                   TICKLEN = .01*height, TITLE = x_axis_title, /EXACT)
  x_axis -> GetProperty, TICKTEXT = x_ticktext     ; FONT keyword not allowed in IDLgrAxi
  x_ticktext -> SetProperty, FONT = axis_tick_font ; the only way to change the font of t

  y_location = DBLARR(1,100)
  i = 0
  exponent = GetNumExp(max_y)
  y_step = 10^exponent
  y_location[0] = min_y
  while y_location[i] lt max_y do begin
      y_location[i + 1] = y_location[i] + y_step
      i = i+1
  endwhile
  num_ticks = i
  y_ticks_location = DBLARR(1, num_ticks)
  y_ticks_vector = OBJARR(1, num_ticks)
  for i = 0, num_ticks-1 do begin
      y_ticks_location[i] = y_location[i]
      tmp = strtrim(string(y_location[i]/y_step, FORMAT = '(d10.1)'), 2) + ' '
      y_ticks_vector[i] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font)
  endfor
  y_axis = OBJ_NEW('IDLgrAxis', 1, RANGE = [min_y, max_y], LOCATION = [min_t, min_y], $
                   MAJOR = num_ticks, TICKLEN = .01*width, TEXTALIGNMENTS = [1, 0.5], $
                   TICKVALUES = y_ticks_location, TICKTEXT = y_ticks_vector, /EXACT)

  tmp = 'x10!U' + strtrim(exponent, 2)
  exponent_text = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, /ENABLE_FORMATTING, $
                          LOCATION = [min_t, 1.05*max_y])

  divider_line = OBJ_NEW('IDLgrPolyline', [llx-2*x_margin, llx + width + x_margin], $
                         [lly - .96*y_margin, lly - .96*y_margin]);, THICK = 2)


  data_view = OBJ_NEW('IDLgrView', UNITS = 3, $
                VIEWPLANE_RECT = [llx - 1.1*x_margin, lly - y_margin, width + 2*x_margin
                LOCATION = [0.0, 0.53], dimensions = [1.0, 0.47], $
                COLOR = [255, 255, 255])

  model -> ADD, [plot, x_axis, y_axis, exponent_text, divider_line]
  data_view -> ADD, model
  return, data_view

END


FUNCTION MAKE_TRE_VIEW, t, y, column

  if column gt 3 then color = [255, 0, 0] else color = [255, 0, 255]

  plot = OBJ_NEW('IDLgrPlot', t, y, COLOR = color)
        model = OBJ_NEW('IDLgrModel')
  axis_title_font = OBJ_NEW('IDLgrFont', 'Times*Italic', size = 10)
  axis_tick_font = OBJ_NEW('IDLgrFont', 'Times', size = 10)

  max_t = MAX(t)
  min_t = MIN(t)
  max_y = MAX(y)
  min_y = MIN(y)
  llx = min_t
  lly = min_y
  width = double(max_t) - double(min_t)
  height = double(max_y) - double(min_y)
  x_margin = .1*width
  y_margin = .3*height

  case column of
          0: title = 'Channel: X + A'
          1: title = 'Channel: X - A'
          2: title = 'Channel: Y + A'
          3: title = 'Channel: Y - A'
          4: title = 'Channel: X + B'
```


19

```
    5: title = 'Channel: X - B'
    6: title = 'Channel: Y + B'
    7: title = 'Channel: Y - B'
    else: title = 'TRE Snapshot'
endcase


plot_title = OBJ_NEW('IDLgrText', title)
title_font = OBJ_NEW('IDLgrFont', 'Times*Bold', size = 10)
plot_title -> SetProperty, font = title_font
                              ;title_buffer = OBJ_NEW('IDLgrBuffer'
                              ;tmp = GetTextDimensions(plot_title)
plot_title -> SetProperty, LOCATION = [.35*width, 1.2*max_y]
                              ;exponent_text = OBJ_NEW('IDLgrText', 'e4', FONT = axis_tick_fon
                              ;                          LOCATION = [min_t, 1.05*max_y])


x = DBLARR(1, 100)
i = 0
t_step = 20
x[0] = t_step
while x[i] lt max_t do begin
    x[i + 1] = x[i] + t_step
    i = i+1
endwhile
num_ticks = i
x_ticks = DBLARR(1, num_ticks)
for i = 0, num_ticks-1 do x_ticks[i] = x[i]
x_axis_title = OBJ_NEW('IDLgrText', 'msec', FONT = axis_title_font)
x_axis = OBJ_NEW('IDLgrAxis', 0, RANGE = [min_t, max_t], LOCATION = [min_t, min_y], $
                 MAJOR = num_ticks, TICKVALUES = x_ticks, $
                 TICKLEN = .01*height, TITLE = x_axis_title, /EXACT)

y_location = DBLARR(1,100)
i = 0
exponent = GetNumExp(height)
y_step = 10^exponent

y_location[0] = min_y
while y_location[i] lt max_y do begin
    y_location[i + 1] = y_location[i] + y_step
    i = i+1
endwhile
num_ticks = i
y_ticks_location = DBLARR(1, num_ticks)
y_ticks_vector = OBJARR(1, num_ticks)
for i = 0, num_ticks-1 do begin
    y_ticks_location[i] = y_location[i]
    tmp = strtrim(string(y_location[i]/y_step, FORMAT = '(d10.1)'), 2) + ' '
    y_ticks_vector[i] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font)
endfor
y_axis = OBJ_NEW('IDLgrAxis', 1, RANGE = [min_y, max_y], LOCATION = [min_t, min_y], $
                 MAJOR = num_ticks, TICKLEN = .01*width, TEXTALIGNMENTS = [1, 0.5], $
                 TICKVALUES = y_ticks_location, TICKTEXT = y_ticks_vector, /EXACT)

x_axis -> GetProperty, ticktext = x_ticktext
x_ticktext -> SetProperty, font = axis_tick_font

tmp = 'e' + strtrim(exponent, 2) ;  '!Utmp'
exponent_text = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, $
                        LOCATION = [min_t, 1.05*max_y])
if column eq -1 then begin
    x_loc = 0.0
    y_loc = 0.0
    dims = [1.0, 1.0]
endif else begin
    y_loc = 1.0 - (column MOD 4 + 1)*.23
    dims = [0.5, 0.23]
    if column gt 3 then x_loc = 0.5 else x_loc = 0.0
endelse

view = OBJ_NEW('IDLgrView', UNITS = 3, $
               VIEWPLANE_RECT = [llx - 2*x_margin, lly - y_margin, width + 3*x_margin, height
               LOCATION = [x_loc, y_loc], dimensions = dims, $
               COLOR = [255, 255, 255])
model -> ADD, [plot, x_axis, y_axis, plot_title, exponent_text]
view -> ADD, model
return, view
END

FUNCTION MAKE_TRE_DIFF_VIEW, t, xy_diff, k

if k gt 1 then color = [255, 0, 0] else color = [255, 0, 255]

; Massage data so that outliers are removed
L = length(xy_diff)
```

plotting in TRE MAR 3/12

```
    5: title = 'Channel: X - B'
    6: title = 'Channel: Y + B'
    7: title = 'Channel: Y - B'
        else: title = 'TRE Snapshot'
endcase


plot_title = OBJ_NEW('IDLgrText', title)
title_font = OBJ_NEW('IDLgrFont', 'Times*Bold', size = 10)
plot_title -> SetProperty, font = title_font
;title_buffer = OBJ_NEW('IDLgrBuffer'
;tmp = GetTextDimensions(plot_title)
plot_title -> SetProperty, LOCATION = [.35*width, 1.2*max_y]
;exponent_text = OBJ_NEW('IDLgrText', 'e4', FONT = axis_tick_font, $
;                 LOCATION = [min_t, 1.05*max_y])


x = DBLARR(1, 100)
i = 0
t_step = 20
x[0] = t_step
while x[i] lt max_t do begin
        x[i + 1] = x[i] + t_step
        i = i+1
endwhile
num_ticks = i
x_ticks = DBLARR(1, num_ticks)
for i = 0, num_ticks-1 do x_ticks[i] = x[i]
x_axis_title = OBJ_NEW('IDLgrText', 'msec', FONT = axis_title_font)
x_axis = OBJ_NEW('IDLgrAxis', 0, RANGE = [min_t, max_t], LOCATION = [min_t, min_y], $
                 MAJOR = num_ticks, TICKVALUES = x_ticks, $
                 TICKLEN = .01*height, TITLE = x_axis_title, /EXACT)

y_location = DBLARR(1,100)
i = 0
      exponent = GetNumExp(height)
      y_step = 10^exponent

y_location[0] = min_y
while y_location[i] lt max_y do begin
        y_location[i + 1] = y_location[i] + y_step
        i = i+1
endwhile
num_ticks = i
y_ticks_location = DBLARR(1, num_ticks)
y_ticks_vector = OBJARR(1, num_ticks)
for i = 0, num_ticks-1 do begin
        y_ticks_location[i] = y_location[i]
        tmp = strtrim(string(y_location[i]/y_step, FORMAT = '(d10.1)'), 2) + ' '
        y_ticks_vector[i] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font)
endfor
y_axis = OBJ_NEW('IDLgrAxis', 1, RANGE = [min_y, max_y], LOCATION = [min_t, min_y], $
                 MAJOR = num_ticks, TICKLEN = .01*width, TEXTALIGNMENTS = [1, 0.5], $
                 TICKVALUES = y_ticks_location, TICKTEXT = y_ticks_vector, /EXACT)

x_axis -> GetProperty, ticktext = x_ticktext
x_ticktext -> SetProperty, font = axis_tick_font

      tmp = 'e' + strtrim(exponent, 2)  ;  '!Utmp'
      exponent_text = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, $
                              LOCATION = [min_t, 1.05*max_y])
      if column eq -1 then begin
          x_loc = 0.0
          y_loc = 0.0
          dims = [1.0, 1.0]
      endif else begin
          y_loc = 1.0 - (column MOD 4 + 1)*.23
          dims = [0.5, 0.23]
          if column gt 3 then x_loc = 0.5 else x_loc = 0.0
      endelse

      view = OBJ_NEW('IDLgrView', UNITS = 3, $
                 VIEWPLANE_RECT = [llx - 2*x_margin, lly - y_margin, width + 3*x_margin,
                      LOCATION = [x_loc, y_loc], dimensions = dims, $
                                COLOR = [255, 255, 255])
      model -> ADD, [plot, x_axis, y_axis, plot_title, exponent_text]
      view -> ADD, model
return, view
  END
```


Ref 4

20

```
average_value = TOTAL(xy_diff)/L
deviation = average_value - xy_diff  ;hopefully this will output a vector - it does!
average_deviation = TOTAL(deviation)/L
for i = 0, L-1 do begin
    if deviation[i] gt 2*average_deviation then begin
        xy_diff[i] = average_value   ; instead of removing the outlier and shifting all the da
                                     ; with average value
    end
endfor

plot = OBJ_NEW('IDLgrPlot', t, xy_diff, COLOR = color)
model = OBJ_NEW('IDLgrModel')
axis_title_font = OBJ_NEW('IDLgrFont', 'Times*Italic', size = 10)
axis_tick_font = OBJ_NEW('IDLgrFont', 'Times', size = 10)

max_t = MAX(t)
min_t = MIN(t)
max_y = MAX(xy_diff)
min_y = MIN(xy_diff)
llx = min_t
lly = min_y
width = double(max_t) - double(min_t)
height = double(max_y) - double(min_y)
x_margin = .1*width
y_margin = .3*height

case k of
    0: title = 'A Side:  |X_plus - X_minus|'
    1: title = 'A Side:  |Y_plus - Y_minus|'
    2: title = 'B Side:  |X_plus - X_minus|'
    3: title = 'B Side:  |Y_plus - Y_minus|'
    else: title = 'TRE Snapshot'
endcase

plot_title = OBJ_NEW('IDLgrText', title)
title_font = OBJ_NEW('IDLgrFont', 'Times*Bold', size = 10)
plot_title -> SetProperty, font = title_font
;print, 'max_y', max_y
plot_title -> SetProperty, LOCATION = [min_t + .15*width, min_y + 1.2*height] ;.15*width, .5*m

x = DBLARR(1, 100)
i = 0
t_step = 20
x[0] = t_step
while x[i] lt max_t do begin
    x[i + 1] = x[i] + t_step
    i = i+1
endwhile
num_ticks = i
x_ticks = DBLARR(1, num_ticks)
for i = 0, num_ticks-1 do x_ticks[i] = x[i]
x_axis_title = OBJ_NEW('IDLgrText', 'msec', FONT = axis_title_font)
x_axis = OBJ_NEW('IDLgrAxis', 0, RANGE = [min_t, max_t], LOCATION = [min_t, min_y], $
                 MAJOR = num_ticks, TICKVALUES = x_ticks, $
                 TICKLEN = .01*height, TITLE = x_axis_title, /EXACT)

y_location = DBLARR(1,100)
i = 0
exponent = GetNumExp(height)
y_step = 10^exponent

y_location[0] = min_y
while y_location[i] lt max_y do begin
    y_location[i + 1] = y_location[i] + y_step
    i = i+1
endwhile
num_ticks = i
y_ticks_location = DBLARR(1, num_ticks)
y_ticks_vector = OBJARR(1, num_ticks)
for i = 0, num_ticks-1 do begin
    y_ticks_location[i] = y_location[i]
    tmp = strtrim(string(y_location[i]/y_step, FORMAT = '(d10.1)'), 2) + ' '
    y_ticks_vector[i] = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font)
endfor
y_axis = OBJ_NEW('IDLgrAxis', 1, RANGE = [min_y, max_y], LOCATION = [min_t, min_y], $
                 MAJOR = num_ticks, TICKLEN = .01*width, TEXTALIGNMENTS = [1, 0.5], $
                 TICKVALUES = y_ticks_location, TICKTEXT = y_ticks_vector, /EXACT)

x_axis -> GetProperty, ticktext = x_ticktext
x_ticktext -> SetProperty, font = axis_tick_font

tmp = 'e' + strtrim(exponent, 2) ;  '!Utmp'
exponent_text = OBJ_NEW('IDLgrText', tmp, FONT = axis_tick_font, $
                        LOCATION = [min_t, 1.05*max_y])

if k eq -1 then begin
```

TRE Plot
MCR 37

SU GP-B
INSP 31

Rt 1

21

```
      x_loc = 0.0                                                                <
      y_loc = 0.0                                                                <
      dims = [1.0, 1.0]                                                          <
  endif else begin                                                               <
      y_loc = 1.0 - (k MOD 2 + 1)*.46                                            <
      dims = [0.5, 0.46]                                                         <
      if k gt 1 then x_loc = 0.5 else x_loc = 0.0                                <
  endelse                                                                        <
                                                                                 <
  view = OBJ_NEW('IDLgrView', UNITS = 3, $                                       <
               VIEWPLANE_RECT = [llx - 2*x_margin, lly - y_margin, width + 3*x_margin, height   <
               LOCATION = [x_loc, y_loc], dimensions = dims, $                   <
               COLOR = [255, 255, 255])                                          <
  model -> ADD, [plot, x_axis, y_axis, plot_title, exponent_text]                <
  view -> ADD, model                                                             <
  return, view                                                                   <
END                                                                              <
                                                                                 <
FUNCTION RANGEISGOOD, first, last
  if (ARRAY_EQUAL(first, '') and ARRAY_EQUAL(last, '')) then begin
      tmp = DIALOG_MESSAGE('Please enter range under "Show Snapshots: "')
      return, 0
  endif

  return, 1
END

PRO PRINT_PLOT, image_to_print
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list

  file = 'temp'
  entry_device = !d.name
  tvlct, r, g, b, /get
  loadct, 0
  set_plot, 'PS'
  device, /color, bits_per_pixel=1, filename=file, ysize=10.5, yoffset=0.25, /inches, /portrait
  tv, image_to_print, 0, 0, xsize = !d.x_vsize, ysize = !d.y_vsize, true = 1
  device, /close_file
  set_plot, entry_device
  tvlct, r, g, b

  wait, 1.5

  SPAWN,"lp -d" + printer_list(printer) + " " + file  ; 'end31p'
  SPAWN, "rm -f " + file
END

FUNCTION CONVERT_IMAGE_TO_BW, image
  ; Converts image to black and white.
  ; Note: This algorithm takes advantage of the fact that these are BYTE arrays,
  ; so 0B/255B --> 0B, 1B/255B --> 0B, 3B/255B --> 0B, etc.
  ; Adding together the Red, Green, and Blue arrays and performing the array operations
  ; below converts all non-white values (white = [255, 255, 255]) to black (= [0, 0, 0]).

  temp_array = image[0, *, *] + image[1, *, *] + image[2, *, *]
  temp_array = ((temp_array + 2B)/255B)*255B
  image[0, *, *] = temp_array
  image[1, *, *] = temp_array
  image[2, *, *] = temp_array

  return, image
END

PRO RIGHT_CLICK_EVENT, event
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON PRINTBUFFER, print_buffer
  ;COMMON BUTTON, button_active, print_button
  COMMON RIGHTMENU, button_active, popup, lookup_table                          |

  ; should make a common block with just the data ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   <
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mode, sre   <
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tre_snap_   <
    GSSData, snap_id_id, do_all, did_output                                     <
                                                                                 <
  WIDGET_CONTROL, popup, GET_UVAL = i
  button = event.value

  WIDGET_CONTROL, popup, /DESTROY
  button_active = 0

  ;WIDGET_CONTROL, event.top, GET_UVAL = i
  ;WIDGET_CONTROL, event.id, /DESTROY
```

MCK
3/2

```
FUNCTION RANGEISGOOD, first, last
  if (ARRAY_EQUAL(first, '') and ARRAY_EQUAL(last, '')) then begin
      tmp = DIALOG_MESSAGE('Please enter range under "Show Snapshots: "')
      return, 0
  endif

  return, 1
END

PRO PRINT_PLOT, image_to_print
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list

  file = 'temp'
  entry_device = !d.name
  tvlct, r, g, b, /get
  loadct, 0
  set_plot, 'PS'
  device, /color, bits_per_pixel=1, filename=file, ysize=10.5, yoffset=0.25, /inches, /po
  tv, image_to_print, 0, 0, xsize = !d.x_vsize, ysize = !d.y_vsize, true = 1
  device, /close_file
  set_plot, entry_device
  tvlct, r, g, b

  wait, 1.5

  SPAWN,"lp -d" + printer_list(printer) + " " + file  ; 'end31p'
  SPAWN, "rm -f " + file
END

FUNCTION CONVERT_IMAGE_TO_BW, image
  ; Converts image to black and white.
  ; Note: This algorithm takes advantage of the fact that these are BYTE arrays,
  ; so 0B/255B --> 0B, 1B/255B --> 0B, 3B/255B --> 0B, etc.
  ; Adding together the Red, Green, and Blue arrays and performing the array operations
  ; below converts all non-white values (white = [255, 255, 255]) to black (= [0, 0, 0]).

  temp_array = image[0, *, *] + image[1, *, *] + image[2, *, *]
  temp_array = ((temp_array + 2B)/255B)*255B
  image[0, *, *] = temp_array
  image[1, *, *] = temp_array
  image[2, *, *] = temp_array

  return, image
END

PRO RIGHT_CLICK_EVENT, event
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON PRINTBUFFER, print_buffer
  ;COMMON BUTTON, button_active, print_button
  COMMON RIGHTMENU, button_active, popup
```

MCK
3/2

```
  WIDGET_CONTROL, popup, GET_UVAL = i
  button = event.value

  WIDGET_CONTROL, popup, /DESTROY
  button_active = 0

  ;WIDGET_CONTROL, event.top, GET_UVAL = i
  ;WIDGET_CONTROL, event.id, /DESTROY
```

SU GP-B INSP 31

A47

22

```
print_buffer[i] -> GetProperty, IMAGE_DATA = image_to_print

CASE button OF
    1:  begin  ; this plot, B&W
        image_to_print = CONVERT_IMAGE_TO_BW(image_to_print)
        PRINT_PLOT, image_to_print
    end
    2:  begin  ; this plot, Color
        PRINT_PLOT, image_to_print
    end
    4:  begin  ; print all open plots, B&W
        active_widgets = WIDGET_INFO(/MANAGED)
        for i = 2, length(active_widgets)-1 do begin            ; skip first two elements, si
            WIDGET_CONTROL, active_widgets[i], GET_UVAL = j     ; and the Snapshot main windo
            print_buffer[j] -> GetProperty, IMAGE_DATA = image_to_print
            image_to_print = CONVERT_IMAGE_TO_BW(image_to_print)
            PRINT_PLOT, image_to_print
        endfor
    end
    5:  begin  ; print all open plots, Color
        active_widgets = WIDGET_INFO(/MANAGED)
        for i = 2, length(active_widgets)-1 do begin            ; skip first two elements, si
            WIDGET_CONTROL, active_widgets[i], GET_UVAL = j     ; and the Snapshot main windo
            print_buffer[j] -> GetProperty, IMAGE_DATA = image_to_print
            PRINT_PLOT, image_to_print
        endfor
    end
    6:  begin  ; display tabulated numerical data
        ;tmp = DIALOG_MESSAGE('Not yet implemented', /INFO)
        CREATE_TABLE   ; (lookup_table[i])  CURRENTLY DISPLAYS ALL DATA, NOT JUST THIS PLOT'S
        ;spawn, '/usr/openwin/bin/textedit -read_only -Ws 1000 500 ' + outputfile + ' &'
    end
    7:  begin  ; close all open plots
        active_widgets = WIDGET_INFO(/MANAGED)
        for i = 2, length(active_widgets)-1 do begin            ; skip first two elements, si
            WIDGET_CONTROL, active_widgets[i], /DESTROY         ; and the Snapshot main windo
        endfor
    end
    else: print, 'Error in RIGHT_CLICK_EVENT'
ENDCASE


END


PRO DO_OUTPUT_EVENT, event

  ;COMMON BUTTON, button_active, print_button
  COMMON RIGHTMENU, button_active, popup, lookup_table
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list

  if event.type eq 4 then begin

      kid = WIDGET_INFO(event.top, /CHILD)
      WIDGET_CONTROL, kid, GET_VALUE = plot_window
      WIDGET_CONTROL, kid, GET_UVAL = viewgroup



      plot_window -> DRAW, viewgroup
  endif

  active = 0 ; button_active is necessary (?) because the first time this function is called, po
  if button_active then active = WIDGET_INFO(popup, /MANAGED)

  if (event.release ne 4) and active then begin
      ;WIDGET_CONTROL, print_button, /DESTROY
      WIDGET_CONTROL, popup, /DESTROY
      button_active = 0
  endif

  if event.press eq 4 then begin
      WIDGET_CONTROL, event.top, GET_UVAL = i
      WIDGET_CONTROL, event.top, TLB_GET_SIZE = size

      options = [ '1\Print This Plot (' + printer_list(printer) + ')' , $
                  '0\Black and White' , $
                  '2\Color' , $
                  '1\Print All Open Plots (' + printer_list(printer) + ')' , $
                  '0\Black and White' , $
                  '2\Color' , $
                  '0\Display Tabulated Data' ,$
                  '0\Close All Open Plots' ]

      popup = WIDGET_BASE(TITLE = 'Print', XOFFSET = event.x, YOFFSET = size[1] - event.y, UVALU
```

```
print_buffer[i] -> GetProperty, IMAGE_DATA = image_to_print

CASE button OF
    1:  begin  ; this plot, B&W
        image_to_print = CONVERT_IMAGE_TO_BW(image_to_print)
        PRINT_PLOT, image_to_print
    end
    2:  begin  ; this plot, Color
        PRINT_PLOT, image_to_print
    end
    4:  begin  ; print all open plots, B&W
        active_widgets = WIDGET_INFO(/MANAGED)
        for i = 2, length(active_widgets)-1 do begin            ; skip first two eleme
            WIDGET_CONTROL, active_widgets[i], GET_UVAL = j     ; and the Snapshot mai
            print_buffer[j] -> GetProperty, IMAGE_DATA = image_to_print
            image_to_print = CONVERT_IMAGE_TO_BW(image_to_print)
            PRINT_PLOT, image_to_print
        endfor
    end
    5:  begin  ; print all open plots, Color
        active_widgets = WIDGET_INFO(/MANAGED)
        for i = 2, length(active_widgets)-1 do begin            ; skip first two eleme
            WIDGET_CONTROL, active_widgets[i], GET_UVAL = j     ; and the Snapshot mai
            print_buffer[j] -> GetProperty, IMAGE_DATA = image_to_print
            PRINT_PLOT, image_to_print
        endfor
    end

    6:  begin  ; close all open plots
<
<
<
<
        active_widgets = WIDGET_INFO(/MANAGED)
        for i = 2, length(active_widgets)-1 do begin            ; skip first two eleme
            WIDGET_CONTROL, active_widgets[i], /DESTROY         ; and the Snapshot mai
        endfor
    end
    else: print, 'Error in RIGHT_CLICK_EVENT'
ENDCASE


END


PRO DO_OUTPUT_EVENT, event

  ;COMMON BUTTON, button_active, print_button
  COMMON RIGHTMENU, button_active, popup
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list

  if event.type eq 4 then begin
>     print, 'event.top', event.top
      kid = WIDGET_INFO(event.top, /CHILD)
      WIDGET_CONTROL, kid, GET_VALUE = plot_window
      WIDGET_CONTROL, kid, GET_UVAL = viewgroup
> print, 'view', viewgroup
> ;print, kid
> ;viewgroup = plot_window -> GET(ISA = 'IDLgrViewgroup')
      plot_window -> DRAW, viewgroup
  endif

  active = 0 ; button_active is necessary (?) because the first time this function is cal
  if button_active then active = WIDGET_INFO(popup, /MANAGED)

  if (event.release ne 4) and active then begin
      ;WIDGET_CONTROL, print_button, /DESTROY
      WIDGET_CONTROL, popup, /DESTROY
      button_active = 0
  endif

  if event.press eq 4 then begin
      WIDGET_CONTROL, event.top, GET_UVAL = i
      WIDGET_CONTROL, event.top, TLB_GET_SIZE = size

      options = [ '1\Print This Plot (' + printer_list(printer) + ')' , $
                  '0\Black and White' , $
                  '2\Color' , $
                  '1\Print All Open Plots (' + printer_list(printer) + ')' , $
                  '0\Black and White' , $
                  '2\Color' , $

                  '0\Close All Open Plots' ]

      popup = WIDGET_BASE(TITLE = 'Print', XOFFSET = event.x, YOFFSET = size[1] - event.y
```

TREplot NCR 3/2

TREplot NCR 3/2

TRE plot options NCR 3/2

SU GP-B INSP 31

23

```
    print_button = CW_PDMENU(popup, options, /COLUMN, /RETURN_INDEX)

    ;print_button = WIDGET_BUTTON(event.top, VALUE = 'Print this plot', XOFFSET = event.x,$
    ;                              YOFFSET = size[1] - event.y, EVENT_PRO = 'RIGHT_CLICK_EVENT'
    button_active = 1

    ;popup_struct = { popup:popup, print_button:print_button, options:options, button_active:b
    ;WIDGET_CONTROL, popup, SET_UVALUE=i   ;popup_struct
    WIDGET_CONTROL, popup, /REALIZE

    XMANAGER, 'RIGHT_CLICK', popup,  EVENT_HANDLER='RIGHT_CLICK_EVENT', /NO_BLOCK
  endif
END

FUNCTION SNAPSHOTINRANGE, snapshot, squid, nSnapshots
  if snapshot lt nSnapshots[squid] then return, 1 else return, 0
END


PRO DO_OUTPUT
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mode, sre
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tre_snap_
    GSSData, snap_id_id, do_all, did_output
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON PRINTBUFFER, print_buffer

  ;COMMON BUTTON, button_active, print_button
  COMMON RIGHTMENU, button_active, popup, lookup_table

  button_active = 0

  ;if not do_all then begin
  ;    first_snapshot = fix(first_snapshot[0]) ;get first (and only) value of string array, then
  ;    last_snapshot = fix(last_snapshot[0]) ; Ah! Now I see this could be done with a CW_FIELD
  ;endif


  first_snapshot = fix(first_snapshot[0]) ;get first (and only) value of string array, then turn
  last_snapshot = fix(last_snapshot[0])

  if display_options[0] eq 1 then begin ;Display to screen

      CASE sub_system OF
          0:  begin              ;SRE
              if squid_number eq 0 then begin
                  min = 0
                  max = 3
              endif else begin
                  min = squid_number-1
                  max = min
              endelse

                          ; Check to see if snapshots are in range
              for squid = min, max do begin
                  if not SnapshotInRange(last_snapshot-1, squid, nSnapshots) then begin
                      tmp = 'Snapshot out of range. Squid ' + strtrim(squid + 1, 2) + ' has ' +
                        strtrim(nSnapshots[squid], 2) + ' available.'
                      tmp = DIALOG_MESSAGE(tmp, /INFORMATION)



                      return
                  endif
              endfor

                          ; Plot stuff using object graphics
              i = 0
              print_buffer = OBJARR(1,(max - min + 1)*((last_snapshot-first_snapshot) + 1))
              DEVICE, GET_SCREEN_SIZE = screen_size

              ; lookup table to associate plot window index with the squid/snapshot number
              lookup_table = INTARR(3, 20) ; capped max number of plots at 20 elsewhere (should


              for squid = min, max do begin
                  for snap = first_snapshot-1, last_snapshot-1 do begin

                      lookup_table[i] = [sub_system, squid, snap]
                      tmp = 'SRE Snapshot ' + strtrim(snap + 1, 2) + ' SQUID ' + strtrim(squid +
                      base_widget = WIDGET_BASE(TITLE = tmp, UVAL = i)
                      draw_widget = WIDGET_DRAW(base_widget, xsize=600, ysize=700, color_model =
                                          RETAIN = 0, /EXPOSE_EVENTS, /BUTTON_EVENTS)  ; 4

                      viewgroup = OBJ_NEW('IDLgrViewgroup')
```

TRE print

NCR 3/2

```
    print_button = CW_PDMENU(popup, options, /COLUMN, /RETURN_INDEX)

    ;print_button = WIDGET_BUTTON(event.top, VALUE = 'Print this plot', XOFFSET = event
    ;                              YOFFSET = size[1] - event.y, EVENT_PRO = 'RIGHT_CLICK
    button_active = 1

    ;popup_struct = { popup:popup, print_button:print_button, options:options, button_a
    ;WIDGET_CONTROL, popup, SET_UVALUE=i   ;popup_struct
    WIDGET_CONTROL, popup, /REALIZE

    XMANAGER, 'RIGHT_CLICK', popup,  EVENT_HANDLER='RIGHT_CLICK_EVENT', /NO_BLOCK
  endif
END

FUNCTION SNAPSHOTINRANGE, snapshot, squid, nSnapshots
  if snapshot lt nSnapshots[squid] then return, 1 else return, 0
END


PRO DO_OUTPUT
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mo
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tr
    GSSData, snap_id_id, do_all, did_output
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON PRINTBUFFER, print_buffer

  ;COMMON BUTTON, button_active, print_button
  COMMON RIGHTMENU, button_active, popup

  button_active = 0

  ;if not do_all then begin
  ;    first_snapshot = fix(first_snapshot[0]) ;get first (and only) value of string arra
  ;    last_snapshot = fix(last_snapshot[0]) ; Ah! Now I see this could be done with a CW
  ;endif


  first_snapshot = fix(first_snapshot[0]) ;get first (and only) value of string array, th
  last_snapshot = fix(last_snapshot[0])

  if display_options[0] eq 1 then begin ;Display to screen

      CASE sub_system OF
          0:  begin              ;SRE
              if squid_number eq 0 then begin
                  min = 0
                  max = 3
              endif else begin
                  min = squid_number-1
                  max = min
              endelse

                          ; Check to see if snapshots are in range
              for squid = min, max do begin
                  if not SnapshotInRange(last_snapshot-1, squid, nSnapshots) then begin
                      tmp = 'Snapshot out of range. Squid ' + strtrim(squid + 1, 2) + ' h
                        strtrim(nSnapshots[squid], 2) + ' available.'
                      tmp = DIALOG_MESSAGE(tmp, /INFORMATION)
                      ;print, 'squid', squid
                      ;print, 'last_snapshot - 1', last_snapshot -1
                      ;print, 'nSnapshots', nSnapshots
                      return
                  endif
              endfor

                          ; Plot stuff using object graphics
              i = 0
              print_buffer = OBJARR(1,(max - min + 1)*((last_snapshot-first_snapshot) + 1
              DEVICE, GET_SCREEN_SIZE = screen_size



              for squid = min, max do begin
                  for snap = first_snapshot-1, last_snapshot-1 do begin

                      tmp = 'SRE Snapshot ' + strtrim(snap + 1, 2) + ' SQUID ' + strtrim(
                      base_widget = WIDGET_BASE(TITLE = tmp, UVAL = i)
                      draw_widget = WIDGET_DRAW(base_widget, xsize=600, ysize=700, color_
                                          RETAIN = 0, /EXPOSE_EVENTS, /BUTTON_EVENT

                      viewgroup = OBJ_NEW('IDLgrViewgroup')
```

cleanup

24

SU GP-B INSP 31

```
        ;border = OBJ_NEW('IDLgrPolyline', [-1, 1, 1, -1], $
        ;                  [-1, -1, 1, 1], THICK = 5)

        background = OBJ_NEW('IDLgrView', COLOR = [200, 200, 200])
        model = OBJ_NEW('IDLgrModel') ; useless, but IDL requires it
        tmp = 'SQUID ' + strtrim(squid + 1, 2) + '.  Snapshot ' + strtrim(snap + 1
              'Type ' + strtrim(sre_snap_type, 2) + '.  Cycle ' + strtrim(cycle, 2
              '.  Start Time: ' + strtrim((*SquidData[squid])[snap].VehicleTime, 2
        caption = OBJ_NEW('IDLgrText', tmp, LOCATION = [-0.95, -0.97])
        model -> ADD, [caption] ;, border]
        background -> ADD, model
        viewgroup -> ADD, background

        y = (*SquidData[squid])[snap].values
        HowMany = length(y)
        t = DBLARR(1, HowMany)
        timestep = double(1)/2200*1000
        t[0] = 0  ; start at 0 so it looks better
        for k = 1, HowMany-1 do begin
            t[k] = t[k-1] + timestep
        endfor


        SRE_view = MAKE_SRE_VIEW(t, y)
        FFT_view = MAKE_FFT_VIEW(t, y)
        viewgroup -> ADD, [SRE_view, FFT_view]

        WIDGET_CONTROL, draw_widget, SET_UVALUE=viewgroup
        WIDGET_CONTROL, base_widget, /REALIZE
        WIDGET_CONTROL, draw_widget, GET_VALUE=plot_window
        XMANAGER, 'DO_OUTPUT', base_widget
        plot_window -> Draw, viewgroup

        ;DPI = 100.0  ; default is 72 DPI and increasing this only makes printed i
        ;res = 1/(DPI/2.54)

        ;bw_palette = OBJ_NEW('IDLgrPalette', [0, 255], [0, 255], [0, 255])

        print_buffer[i] = OBJ_NEW('IDLgrBuffer') ;, PALETTE = bw_palette, $
                                ; COLOR_MODEL = 1, N_COLORS = 2, $
                                ; RESOLUTION = [res, res])
        print_buffer[i] -> SetProperty, DIMENSIONS=screen_size
        print_buffer[i] -> Draw, viewgroup

        i = i+1

      endfor
    endfor
```

MCR 372

```
;*****************************************************************************
; OBSOLETE DIRECT GRAPHICS OUTPUT
;             if squid_number eq 0 then begin
;                 min = 0
;                 max = 3
;             endif else begin
;                 min = squid_number-1
;                 max = squid_number-1
;             endelse

;             window_index = 1
;             for squid = min, max do begin

;                 if squid_number eq 0 then begin
;                     first_snapshot = 1 ;
;                     last_snapshot = nSnapshots[squid] ;
;                 endif

;                 window, window_index
;                 window_index = window_index + 1
;                 !p.multi = [0, 2, 2] ; 2x2 plots per page

;                 for snap = first_snapshot-1, last_snapshot-1 do begin
;                                 ; Make a time vector (we're given just
;                                 ; one time value, and we know the step
;                                 ; size

;                     HowMany = length((*SquidData[squid])[snap].values)

;                     times = DBLARR(1, HowMany)
;                     timestep = double(1)/2200*1000

;                     times[0] = 0    ; start at 0 so it looks better (*SquidData[squid])[snap
;                     for i = 1, HowMany-1 do begin
;                         times[i] = times[i-1] + timestep
```

```
> print, draw_widget

| print, viewgroup
```


```
        ;border = OBJ_NEW('IDLgrPolyline', [-1, 1, 1, -1], $
        ;                  [-1, -1, 1, 1], THICK = 5)

        background = OBJ_NEW('IDLgrView', COLOR = [200, 200, 200])
        model = OBJ_NEW('IDLgrModel') ; useless, but IDL requires it
        tmp = 'SQUID ' + strtrim(squid + 1, 2) + '.  Snapshot ' + strtrim(s
              'Type ' + strtrim(sre_snap_type, 2) + '.  Cycle ' + strtrim(c
              '.  Start Time: ' + strtrim((*SquidData[squid])[snap].Vehicle
        caption = OBJ_NEW('IDLgrText', tmp, LOCATION = [-0.95, -0.97])
        model -> ADD, [caption] ;, border]
        background -> ADD, model
        viewgroup -> ADD, background

        y = (*SquidData[squid])[snap].values
        HowMany = length(y)
        t = DBLARR(1, HowMany)
        timestep = double(1)/2200*1000
        t[0] = 0  ; start at 0 so it looks better
        for k = 1, HowMany-1 do begin
            t[k] = t[k-1] + timestep
        endfor


        SRE_view = MAKE_SRE_VIEW(t, y)
        FFT_view = MAKE_FFT_VIEW(t, y)
        viewgroup -> ADD, [SRE_view, FFT_view]

        WIDGET_CONTROL, draw_widget, SET_UVALUE=viewgroup
        WIDGET_CONTROL, base_widget, /REALIZE
        WIDGET_CONTROL, draw_widget, GET_VALUE=plot_window
        XMANAGER, 'DO_OUTPUT', base_widget
        plot_window -> Draw, viewgroup

        ;DPI = 100.0  ; default is 72 DPI and increasing this only makes pr
        ;res = 1/(DPI/2.54)

        ;bw_palette = OBJ_NEW('IDLgrPalette', [0, 255], [0, 255], [0, 255])

        print_buffer[i] = OBJ_NEW('IDLgrBuffer') ;, PALETTE = bw_palette, $
                                ; COLOR_MODEL = 1, N_COLORS = 2, $
                                ; RESOLUTION = [res, res])
        print_buffer[i] -> SetProperty, DIMENSIONS=screen_size
        print_buffer[i] -> Draw, viewgroup

        i = i+1

      endfor
    endfor
```

```
;*****************************************************************************
; OBSOLETE DIRECT GRAPHICS OUTPUT
;             if squid_number eq 0 then begin
;                 min = 0
;                 max = 3
;             endif else begin
;                 min = squid_number-1
;                 max = squid_number-1
;             endelse

;             window_index = 1
;             for squid = min, max do begin

;                 if squid_number eq 0 then begin
;                     first_snapshot = 1 ;
;                     last_snapshot = nSnapshots[squid] ;
;                 endif

;                 window, window_index
;                 window_index = window_index + 1
;                 !p.multi = [0, 2, 2] ; 2x2 plots per page

;                 for snap = first_snapshot-1, last_snapshot-1 do begin
;                                 ; Make a time vector (we're given just
;                                 ; one time value, and we know the step
;                                 ; size

;                     HowMany = length((*SquidData[squid])[snap].values)

;                     times = DBLARR(1, HowMany)
;                     timestep = double(1)/2200*1000

;                     times[0] = 0    ; start at 0 so it looks better (*SquidData[squid
;                     for i = 1, HowMany-1 do begin
;                         times[i] = times[i-1] + timestep
```


MHC

```
;                   endfor

;                   array = DBLARR(2, HowMany)
;                   array = [times, (*SquidData[squid])[snap].values]
;                             ;first number is squidnumber-1, second
;                             ;number is snapshotnumber-1

;                   if (snap+1) MOD 4 ne 0 then begin ;make a new window every 4 plots
;                       plot, times, (*SquidData[squid])[snap].values
;                       title = 'this' ;string('SQUID ', squid, ' Snapshot ', snap)
;                   endif else begin
;                       plot, times, (*SquidData[squid])[snap].values
;                       window, window_index ;set index of window to unique value
;                       window_index = window_index + 1
;                       !p.multi = [0, 2, 2] ;new plot window
;                   endelse

;               endfor

;           endfor
;*****************************************************************************************
;*****************************************************************************************
;*****************************************************************************************

        end

        1:  begin     ;TRE     ;Display to Screen

                                ; Make time vector
            time_scale = [2.2, 2.2/4, 2.2, 2.2, 2.2, 0.01]
            nSamples = [330, 330, 330, 330, 330, 100]
            t = INDGEN(1, nSamples[tre_snap_type - 1], /DOUBLE) / time_scale[tre_snap_type - 1

            if tre_snap_type eq 4 or tre_snap_type eq 5 then nColumns = 1 $
            else nColumns = 8

                                ; Plot stuff using object graphics
            i = 0
            j = 100  ; start second tally for difference plots                          |
                     ; at 100 ('i' should never get near 100, so right-click stuff should stil  <
            print_buffer = OBJARR(1, 200)  ; (last_snapshot-first_snapshot) + 1)        <
            DEVICE, GET_SCREEN_SIZE = screen_size

            lookup_table = INTARR(3, 20) ; capped max number of plots at 20 elsewhere (should  <
                                                                                        <
            for snap = first_snapshot-1, last_snapshot-1 do begin
                                                                                        >

                lookup_table[i] = [sub_system, 0, snap]                                 <
                                                                                        <
                base_widget = WIDGET_BASE(TITLE = 'TRE Snapshot ' + strtrim(snap+1,2), UVAL =  <
                                                                                        <
                draw_widget = WIDGET_DRAW(base_widget, xsize=800, ysize=800, color_model = 0,
                                          /BUTTON_EVENTS) ;, /MOTION_EVENTS)

                viewgroup = OBJ_NEW('IDLgrViewgroup')
                background = OBJ_NEW('IDLgrView')
                model = OBJ_NEW('IDLgrModel')      ; useless, but IDL requires it
                tmp = 'TRE Snapshot ' + strtrim(snap + 1, 2) + '. Type ' + $
                      strtrim(tre_snap_type, 2) + '.  Cycle ' + strtrim(cycle, 2)
                caption = OBJ_NEW('IDLgrText', tmp, LOCATION = [-0.95, -0.95])
                model -> ADD, caption
                background -> ADD, model
                viewgroup -> ADD, background

                if nColumns eq 8 then begin
                    for column = 0, 7 do begin
                        y = TREData[snap].AllColumnsData[column, *]
                        view = MAKE_TRE_VIEW(t, y, column)
                        viewgroup -> ADD, view
                    endfor

                endif else begin
                    y = TREData[snap].AllColumnsData[0, *]
                    view = MAKE_TRE_VIEW(t, y, -1)
                    viewgroup -> ADD, view
                endelse

                                                                                        <
                                                                                        <
                diff_base_widget = WIDGET_BASE(TITLE = 'TRE Snapshot ' + strtrim(snap+1,2) + '  <
                diff_draw_widget = WIDGET_DRAW(diff_base_widget, xsize = 700, ysize = 700, col  <
                                               graphics_level = 0, /BUTTON_EVENTS)      <
                diff_viewgroup = OBJ_NEW('IDLgrViewgroup')                              <
                diff_background = OBJ_NEW('IDLgrView')                                  <
```

MCR 312

MCR 312

```
;                   endfor

;                   array = DBLARR(2, HowMany)
;                   array = [times, (*SquidData[squid])[snap].values]
;                             ;first number is squidnumber-1, second
;                             ;number is snapshotnumber-1

;                   if (snap+1) MOD 4 ne 0 then begin ;make a new window every 4 plot
;                       plot, times, (*SquidData[squid])[snap].values
;                       title = 'this' ;string('SQUID ', squid, ' Snapshot ', snap)
;                   endif else begin
;                       plot, times, (*SquidData[squid])[snap].values
;                       window, window_index ;set index of window to unique value
;                       window_index = window_index + 1
;                       !p.multi = [0, 2, 2] ;new plot window
;                   endelse

;               endfor

;           endfor
;*****************************************************************************************
;*****************************************************************************************
;*****************************************************************************************

        end

        1:  begin     ;TRE     ;Display to Screen

                                ; Make time vector
            time_scale = [2.2, 2.2/4, 2.2, 2.2, 2.2, 0.01]
            nSamples = [330, 330, 330, 330, 330, 100]
            t = INDGEN(1, nSamples[tre_snap_type - 1], /DOUBLE) / time_scale[tre_snap_t

            if tre_snap_type eq 4 or tre_snap_type eq 5 then nColumns = 1 $
            else nColumns = 8

                                ; Plot stuff using object graphics
            i = 0
            print_buffer = OBJARR(1, (last_snapshot-first_snapshot) + 1)


            DEVICE, GET_SCREEN_SIZE = screen_size



            for snap = first_snapshot-1, last_snapshot-1 do begin
                base_widget = WIDGET_BASE(TITLE = 'TRE Snapshots', UVAL = i)




                draw_widget = WIDGET_DRAW(base_widget, xsize=800, ysize=800, color_mode
                                          /BUTTON_EVENTS) ;, /MOTION_EVENTS)

                viewgroup = OBJ_NEW('IDLgrViewgroup')
                background = OBJ_NEW('IDLgrView')
                model = OBJ_NEW('IDLgrModel')      ; useless, but IDL requires it
                tmp = 'TRE Snapshot ' + strtrim(snap + 1, 2) + '. Type ' + $
                      strtrim(tre_snap_type, 2) + '.  Cycle ' + strtrim(cycle, 2)
                caption = OBJ_NEW('IDLgrText', tmp, LOCATION = [-0.95, -0.95])
                model -> ADD, caption
                background -> ADD, model
                viewgroup -> ADD, background

                if nColumns eq 8 then begin
                    for column = 0, 7 do begin
                        y = TREData[snap].AllColumnsData[column, *]
                        view = MAKE_TRE_VIEW(t, y, column)
                        viewgroup -> ADD, view
                    endfor

                endif else begin
                    y = TREData[snap].AllColumnsData[0, *]
                    view = MAKE_TRE_VIEW(t, y, -1)
                    viewgroup -> ADD, view
                endelse
```

SU GP-B INSP 31

26

```
                  diff_model = OBJ_NEW('IDLgrModel')                                               <
                  tmp = 'TRE Snapshot ' + strtrim(snap + 1, 2) + '. Type ' + $                     <
                    strtrim(tre_snap_type, 2) + '.  Cycle ' + strtrim(cycle, 2)                    <
                  diff_caption = OBJ_NEW('IDLgrText', tmp, LOCATION = [-0.95, -0.95])              <
                  diff_model -> ADD, diff_caption                                                  <
                  diff_background -> ADD, diff_model                                               <
                  diff_viewgroup -> ADD, diff_background                                           <
                                                                                                   <
                  for k = 0,3 do begin                                                             <
                      xy_diff = ABS(TREData[snap].AllColumnsData[2*k, *] - TREData[snap].AllColu   <
                      if nColumns ne 8 then k = -1                                                 <
                      diff_view = MAKE_TRE_DIFF_VIEW(t, xy_diff, k)                                <
                      diff_viewgroup -> ADD, diff_view                                             <
                  endfor                                                                           <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                  WIDGET_CONTROL, base_widget, /REALIZE                                                              WIDGET_CONTROL, base_widget, /REALIZE
                  WIDGET_CONTROL, draw_widget, GET_VALUE=plot_window                                                 WIDGET_CONTROL, draw_widget, GET_VALUE=plot_window
                  XMANAGER, 'DO_OUTPUT', base_widget                                                                 XMANAGER, 'DO_OUTPUT', base_widget
                  plot_window -> Draw, viewgroup                                                                     plot_window -> Draw, viewgroup

                  print_buffer[i] = OBJ_NEW('IDLgrBuffer')                                                           print_buffer[i] = OBJ_NEW('IDLgrBuffer')
                  print_buffer[i] -> SetProperty, DIMENSIONS=screen_size                                             print_buffer[i] -> SetProperty, DIMENSIONS=screen_size
                  print_buffer[i] -> Draw, viewgroup                                                                 print_buffer[i] -> Draw, viewgroup

                                                                                                   <
                                                                                                   <
                  WIDGET_CONTROL, diff_base_widget, /REALIZE                                       <
                  WIDGET_CONTROL, diff_draw_widget, GET_VALUE = plot_window                        <
                  XMANAGER, 'DO_OUTPUT', diff_base_widget                                          <
                  plot_window -> Draw, diff_viewgroup                                              <
                                                                                                   <
                  print_buffer[j] = OBJ_NEW('IDLgrBuffer')                                         <
                  print_buffer[j] -> SetProperty, DIMENSIONS=screen_size                           <
                  print_buffer[j] -> Draw, diff_viewgroup                                          <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                  i = i+1                                                                                            i = i+1
                  j = j+1                                                                          <
              endfor                                                                                             endfor

          end                                                                                                end

          2:  begin     ;GSS   ;Display to Screen                                                            2:  begin     ;GSS   ;Display to Screen
              CREATE_TABLE                                                                         |             GOTO, JUMP_TO_PRINT
              ; GOTO, JUMP_TO_PRINT                                                                <
              ; no graphical display yet                                                                         ; no graphical display yet
          end                                                                                                end

      ENDCASE                   ;Display to screen (all)                                                 ENDCASE                   ;Display to screen (all)
  endif                         ;Display to screen selected                                          endif                         ;Display to screen selected


  if display_options[1] eq 1 then begin    ;Send to file                                           |   if display_options[1] eq 1 then begin ;Send to printer
      CREATE_TABLE                                                                                 |
                                                                                                   >       CASE sub_system OF
                                                                                                   >           0:  begin             ;SRE
                                                                                                   >               GOTO, JUMP_TO_PRINT
  end                                                                                                          end

did_output = 1                                                                                     |           1:  begin             ;TRE
                                                                                                   >               GOTO, JUMP_TO_PRINT
                                                                                                   >           end

END                                                                                                |           2:  begin             ;GSS
                                                                                                   >               GOTO, JUMP_TO_PRINT
                                                                                                   >           end
                                                                                                   >       ENDCASE
                                                                                                   >   endif

                                                                                                   |   if display_options[2] eq 1 then begin ;Print to file
;;;FIX;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;   | JUMP_TO_PRINT:
;; There needs to be a function to create one table at a time ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;  <
                                                                                                   <
PRO CREATE_TABLE                                                                                   <
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mode, sre   <
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tre_snap_   <
    GSSData, snap_id_id, do_all, did_output                                                        <
  COMMON SNAPSHOT_INFO, display_options, printer, filename                                         <
                                                                                                   <
                                                                                                   <
  CASE sub_system OF                                                                                     CASE sub_system OF
      0:  begin                 ;SRE                                                                         0:  begin             ;SRE
```

TRE printing
MCE 3/12

SU GP-B
INSP 31

A774

27

```
if squid_number eq 0 then begin
    min = 0
    max = 3
endif else begin
    min = squid_number-1
    max = squid_number-1
endelse

for squid = min, max do begin

    if nSnapshots[squid] eq 0 then CONTINUE

    if do_all then begin
        first_snapshot = 1
        last_snapshot = nSnapshots[squid]
        end

        for snap = first_snapshot-1, last_snapshot-1 do begin

            HowMany = length((*SquidData[squid])[snap].values)

                      ; Make a time vector (we're given just one time value, and we kn

            times = DBLARR(1, HowMany)
            timestep = double(1)/2200*1000

            times[0] = (*SquidData[squid])[snap].VehicleTime
            for i = 1, HowMany-1 do begin
                times[i] = times[i-1] + timestep
            endfor

            array = DBLARR(2, HowMany)
            array = [times, (*SquidData[squid])[snap].values] ;first number is squidnu
                      ; second number is snapshotnumber-1

                      ; Print to a file

            WIDGET_CONTROL, filename_text_id, GET_VALUE = tmp

            if array2str(tmp) ne '' then begin
                outputfile = array2str(tmp) ;if the user typed in a filename, use it,
            endif else begin
                outputfile = strcompress(string('Squid', squid+1, '_Snap', snap+1, '_C
            endelse

            outputfile = GETENV("HOME") + '/OUTPUT/' + outputfile

            CATCH, no_dir_present

            if no_dir_present ne 0 then begin
                spawn, 'mkdir ' + GETENV("HOME") + '/OUTPUT/'
            endif

            OPENW, lun, outputfile, /get_lun


            CATCH, data_out_of_range
            if data_out_of_range ne 0 then begin
                FREE_LUN, lun
                spawn, 'rm ' + outputfile
                break
            endif

            printf, lun, 'Decimation: ', (*SquidData[squid])[snap].Decimation
            printf, lun, ''
            printf, lun, ' Vehicle Time    ', '       Value'
            printf, lun, array, format = '(d14.3, i14)'
            FREE_LUN, lun

            CATCH, /CANCEL

            if display_options[0] eq 1 then begin
                spawn, '/usr/openwin/bin/textedit -read_only -Ws 1000 500 ' + outputfi
            endif

            ; This next bit is needed since this currently gets run if display to scre
            ;if display_options[1] eq 1 then begin
            ;    cmd = 'lpr -P ' + printer_list[printer] + ' ' + outputfile
            ;    SPAWN, cmd
            ;endif


                      ; get rid of file we just created if user didn't ask for a file
            if display_options[1] ne 1 then begin
                spawn, 'rm ' + outputfile
```

MCA 372

```
if squid_number eq 0 then begin
    min = 0
    max = 3
endif else begin
    min = squid_number-1
    max = squid_number-1
endelse

for squid = min, max do begin

    if nSnapshots[squid] eq 0 then CONTINUE

    if do_all then begin
        first_snapshot = 1
        last_snapshot = nSnapshots[squid]
    end

    for snap = first_snapshot-1, last_snapshot-1 do begin

        HowMany = length((*SquidData[squid])[snap].values)

        ; Make a time vector (we're given just one time value, and we know

        times = DBLARR(1, HowMany)
        timestep = double(1)/2200*1000

        times[0] = (*SquidData[squid])[snap].VehicleTime
        for i = 1, HowMany-1 do begin
            times[i] = times[i-1] + timestep
        endfor

        array = DBLARR(2, HowMany)
        array = [times, (*SquidData[squid])[snap].values] ;first number is
                  ; second number is snapshotnumber-1

                  ; Print to a file

        WIDGET_CONTROL, filename_text_id, GET_VALUE = tmp

        if array2str(tmp) ne '' then begin
            outputfile = array2str(tmp) ;if the user typed in a filename, u
        endif else begin
            outputfile = strcompress(string('Squid', squid+1, '_Snap', snap
        endelse

        outputfile = GETENV("HOME") + '/OUTPUT/' + outputfile

        CATCH, no_dir_present

        if no_dir_present ne 0 then begin
            spawn, 'mkdir ' + GETENV("HOME") + '/OUTPUT/'
        endif

        OPENW, lun, outputfile, /get_lun


        CATCH, data_out_of_range
        if data_out_of_range ne 0 then begin
            FREE_LUN, lun
            spawn, 'rm ' + outputfile
            break
        endif

        printf, lun, 'Decimation: ', (*SquidData[squid])[snap].Decimation
        printf, lun, ''
        printf, lun, ' Vehicle Time    ', '       Value'
        printf, lun, array, format = '(d14.3, i14)'
        FREE_LUN, lun

        CATCH, /CANCEL

        if display_options[0] eq 1 then begin
            spawn, '/usr/openwin/bin/textedit -read_only -Ws 1000 500 ' + o
        endif

        ; This next bit is needed since this currently gets run if display
        if display_options[1] eq 1 then begin
            cmd = 'lpr -P ' + printer_list[printer] + ' ' + outputfile
            SPAWN, cmd
        endif


                  ; get rid of file we just created if user didn't ask for
        if display_options[2] ne 1 then begin
            spawn, 'rm ' + outputfile
```

SU GP-B INSP 31

28

```
            endif

        endfor
    endfor
end

1:  begin               ;TRE
    for snap = first_snapshot-1, last_snapshot-1 do begin

        if TREdata[snap].VehicleTime eq 0 then CONTINUE

        WIDGET_CONTROL, filename_text_id, GET_VALUE = tmp

        if array2str(tmp) ne '' then begin
            outputfile = array2str(tmp) ;if the user typed in a filename, use it, othe
        endif else begin
            outputfile = strcompress(string('TRE', '_Snap', snap+1, '_Cycle',$
                                            cycle, '.txt'), /REMOVE_ALL)
        endelse

        outputfile = GETENV("HOME") + '/OUTPUT/' + outputfile

        CATCH, no_dir_present

        if no_dir_present ne 0 then begin
            spawn, 'mkdir ' + GETENV("HOME") + '/OUTPUT/'
        endif

        OPENW, lun, outputfile, /get_lun

        CATCH, data_out_of_range

        if data_out_of_range ne 0 then begin
            FREE_LUN, lun
            spawn, 'rm ' + outputfile
            break
        endif

        printf, lun, 'Vehicle Time: ', TREdata[snap].VehicleTime
        printf, lun, ''
        printf, lun, 'Channel: ', TREdata[snap].Channel
        printf, lun, ''
        printf, lun, '      X+A       X-A       Y+A       Y-A       X+B       X-B
        printf, lun, ''
        printf, lun, TREdata[snap].AllColumnsData, format = '(i10, i10, i10, i10, i10,
        FREE_LUN, lun

        CATCH, /CANCEL

        if display_options[0] eq 1 then begin
            spawn, '/usr/openwin/bin/textedit -read_only -Ws 1000 500 ' + outputfile +
        endif;

             ; This next bit is needed since this currently gets run if display to scr
        ;if display_options[1] eq 1 then begin
        ;    cmd = 'lpr -P ' + printer_list[printer] + ' ' + outputfile
        ;    SPAWN, cmd
        ;endif


             ; get rid of file we just created if user didn't ask for a file to be mad
        if display_options[1] ne 1 then begin
            spawn, 'rm ' + outputfile
        endif


    endfor
end

2:  begin               ;GSS

    if squid_number eq 0 then begin
        min = 0
        max = 3
    endif else begin
        min = squid_number-1
        max = squid_number-1
    endelse

    for squid = min, max do begin

        if nSnapshots[squid] eq 0 then CONTINUE

        if do_all then begin
            first_snapshot = 1
```

```
            endif

        endfor
    endfor
end

1:  begin               ;TRE
    for snap = first_snapshot-1, last_snapshot-1 do begin

        if TREdata[snap].VehicleTime eq 0 then CONTINUE

        WIDGET_CONTROL, filename_text_id, GET_VALUE = tmp

        if array2str(tmp) ne '' then begin
            outputfile = array2str(tmp) ;if the user typed in a filename, use i
        endif else begin
            outputfile = strcompress(string('TRE', '_Snap', snap+1, '_Cycle',$
                                            cycle, '.txt'), /REMOVE_ALL)
        endelse

        outputfile = GETENV("HOME") + '/OUTPUT/' + outputfile

        CATCH, no_dir_present

        if no_dir_present ne 0 then begin
            spawn, 'mkdir ' + GETENV("HOME") + '/OUTPUT/'
        endif

        OPENW, lun, outputfile, /get_lun

        CATCH, data_out_of_range

        if data_out_of_range ne 0 then begin
            FREE_LUN, lun
            spawn, 'rm ' + outputfile
            break
        endif

        printf, lun, 'Vehicle Time: ', TREdata[snap].VehicleTime
        printf, lun, ''
        printf, lun, 'Channel: ', TREdata[snap].Channel
        printf, lun, ''
        printf, lun, '      X+A       X-A       Y+A       Y-A       X+B       X
        printf, lun, ''
        printf, lun, TREdata[snap].AllColumnsData, format = '(i10, i10, i10, i1
        FREE_LUN, lun

        CATCH, /CANCEL

        if display_options[0] eq 1 then begin
            spawn, '/usr/openwin/bin/textedit -read_only -Ws 1000 500 ' + outpu
        endif

             ; This next bit is needed since this currently gets run if display
        if display_options[1] eq 1 then begin
            cmd = 'lpr -P ' + printer_list[printer] + ' ' + outputfile
            SPAWN, cmd
        endif


             ; get rid of file we just created if user didn't ask for a file to
        if display_options[2] ne 1 then begin
            spawn, 'rm ' + outputfile
        endif


    endfor
end

2:  begin               ;GSS

    if squid_number eq 0 then begin
        min = 0
        max = 3
    endif else begin
        min = squid_number-1
        max = squid_number-1
    endelse

    for squid = min, max do begin

        if nSnapshots[squid] eq 0 then CONTINUE

        if do_all then begin
            first_snapshot = 1
```

MCL 312

SU GP-B INSP 31

```
    last_snapshot = nSnapshots[squid]
end

for snap = first_snapshot-1, last_snapshot-1 do begin

            ; Print to a file

    WIDGET_CONTROL, filename_text_id, GET_VALUE = tmp

    if array2str(tmp) ne '' then begin
        outputfile = array2str(tmp) ;if the user typed in a filename, use it,
    endif else begin
        outputfile = strcompress(string('GSS', squid+1, '_Snap', snap+1, '_Cyc
                                        cycle, '_Type', gss_snap_type, '.txt')
    endelse

    outputfile = GETENV("HOME") + '/OUTPUT/' + outputfile

    CATCH, no_dir_present

    if no_dir_present ne 0 then begin
        spawn, 'mkdir ' + GETENV("HOME") + '/OUTPUT/'
    endif

    OPENW, lun, outputfile, /get_lun

    CATCH, data_out_of_range

    if data_out_of_range ne 0 then begin
        FREE_LUN, lun
        spawn, 'rm ' + outputfile
        break
    endif

    CASE gss_snap_type OF
        1:  begin ;suspension

            printf, lun, 'Vehicle Time: ', (*GSSData[squid])[snap].VehicleTime
            printf, lun, 'Drag-Free Data Source: ', (*GSSData[squid])[snap].hD
            printf, lun, 'Control Mode: ', (*GSSData[squid])[snap].hControlMod
            printf, lun, 'Commands Enabled: ', (*GSSData[squid])[snap].hComman
            printf, lun, 'IO Mode: ', (*GSSData[squid])[snap].hIOMode, format
            printf, lun, 'Status: ', (*GSSData[squid])[snap].hStatus, format =
            printf, lun, 'Feed-Forward Mode: ', (*GSSData[squid])[snap].hFeedF
            printf, lun, 'Drag-Free Injection: ', (*GSSData[squid])[snap].hDra
            printf, lun, 'Drag-Free Status: ', (*GSSData[squid])[snap].hDragFr
            printf, lun, 'Event: ', (*GSSData[squid])[snap].hEvent, format = '
            printf, lun, 'Spare: ', (*GSSData[squid])[snap].hSpare, format = '
            printf, lun, 'Control Phase: ', (*GSSData[squid])[snap].hControlPh
            printf, lun, '10Hz Index: ', (*GSSData[squid])[snap].h10HzIndex, f
            printf, lun, 'Multiplicity Index: ', (*GSSData[squid])[snap].hMult
            printf, lun, 'Multiplicity Count: ', (*GSSData[squid])[snap].hMult
            printf, lun, ''
            printf, lun, 'aAxisPosition: '
            printf, lun, (*GSSData[squid])[snap].aAxisPosition
            printf, lun, ''
            printf, lun, 'aAxisControlEffort: '
            printf, lun, (*GSSData[squid])[snap].aAxisControlEffort
            printf, lun, ''
            printf, lun, 'VaPlus: '
            printf, lun, (*GSSData[squid])[snap].VaPlus
            printf, lun, ''
            printf, lun, 'VaMinus: '
            printf, lun, (*GSSData[squid])[snap].VaMinus
            printf, lun, ''
            printf, lun, 'a_hat: '
            printf, lun, (*GSSData[squid])[snap].a_hat
            printf, lun, ''
            printf, lun, 'a_hat_dot: '
            printf, lun, (*GSSData[squid])[snap].a_hat_dot
            printf, lun, ''
            printf, lun, 'ControllerBandwidth: '
            printf, lun, (*GSSData[squid])[snap].ControllerBandwidth
            printf, lun, ''
            printf, lun, 'bAxisPosition: '
            printf, lun, (*GSSData[squid])[snap].bAxisPosition
            printf, lun, ''
            printf, lun, 'bAxisControlEffort: '
            printf, lun, (*GSSData[squid])[snap].bAxisControlEffort
            printf, lun, ''
            printf, lun, 'VbPlus: '
            printf, lun, (*GSSData[squid])[snap].VbPlus
            printf, lun, ''
            printf, lun, 'VbMinus: '
            printf, lun, (*GSSData[squid])[snap].VbMinus
            printf, lun, ''
```

```
    last_snapshot = nSnapshots[squid]
end

for snap = first_snapshot-1, last_snapshot-1 do begin

            ; Print to a file

    WIDGET_CONTROL, filename_text_id, GET_VALUE = tmp

    if array2str(tmp) ne '' then begin
        outputfile = array2str(tmp) ;if the user typed in a filename, u
    endif else begin
        outputfile = strcompress(string('GSS', squid+1, '_Snap', snap+1
                                        cycle, '_Type', gss_snap_type,
    endelse

    outputfile = GETENV("HOME") + '/OUTPUT/' + outputfile

    CATCH, no_dir_present

    if no_dir_present ne 0 then begin
        spawn, 'mkdir ' + GETENV("HOME") + '/OUTPUT/'
    endif

    OPENW, lun, outputfile, /get_lun

    CATCH, data_out_of_range

    if data_out_of_range ne 0 then begin
        FREE_LUN, lun
        spawn, 'rm ' + outputfile
        break
    endif

    CASE gss_snap_type OF
        1:  begin ;suspension

            printf, lun, 'Vehicle Time: ', (*GSSData[squid])[snap].Vehi
            printf, lun, 'Drag-Free Data Source: ', (*GSSData[squid])[s
            printf, lun, 'Control Mode: ', (*GSSData[squid])[snap].hCon
            printf, lun, 'Commands Enabled: ', (*GSSData[squid])[snap].
            printf, lun, 'IO Mode: ', (*GSSData[squid])[snap].hIOMode,
            printf, lun, 'Status: ', (*GSSData[squid])[snap].hStatus, f
            printf, lun, 'Feed-Forward Mode: ', (*GSSData[squid])[snap]
            printf, lun, 'Drag-Free Injection: ', (*GSSData[squid])[sna
            printf, lun, 'Drag-Free Status: ', (*GSSData[squid])[snap].
            printf, lun, 'Event: ', (*GSSData[squid])[snap].hEvent, for
            printf, lun, 'Spare: ', (*GSSData[squid])[snap].hSpare, for
            printf, lun, 'Control Phase: ', (*GSSData[squid])[snap].hCo
            printf, lun, '10Hz Index: ', (*GSSData[squid])[snap].h10HzI
            printf, lun, 'Multiplicity Index: ', (*GSSData[squid])[snap
            printf, lun, 'Multiplicity Count: ', (*GSSData[squid])[snap
            printf, lun, ''
            printf, lun, 'aAxisPosition: '
            printf, lun, (*GSSData[squid])[snap].aAxisPosition
            printf, lun, ''
            printf, lun, 'aAxisControlEffort: '
            printf, lun, (*GSSData[squid])[snap].aAxisControlEffort
            printf, lun, ''
            printf, lun, 'VaPlus: '
            printf, lun, (*GSSData[squid])[snap].VaPlus
            printf, lun, ''
            printf, lun, 'VaMinus: '
            printf, lun, (*GSSData[squid])[snap].VaMinus
            printf, lun, ''
            printf, lun, 'a_hat: '
            printf, lun, (*GSSData[squid])[snap].a_hat
            printf, lun, ''
            printf, lun, 'a_hat_dot: '
            printf, lun, (*GSSData[squid])[snap].a_hat_dot
            printf, lun, ''
            printf, lun, 'ControllerBandwidth: '
            printf, lun, (*GSSData[squid])[snap].ControllerBandwidth
            printf, lun, ''
            printf, lun, 'bAxisPosition: '
            printf, lun, (*GSSData[squid])[snap].bAxisPosition
            printf, lun, ''
            printf, lun, 'bAxisControlEffort: '
            printf, lun, (*GSSData[squid])[snap].bAxisControlEffort
            printf, lun, ''
            printf, lun, 'VbPlus: '
            printf, lun, (*GSSData[squid])[snap].VbPlus
            printf, lun, ''
            printf, lun, 'VbMinus: '
            printf, lun, (*GSSData[squid])[snap].VbMinus
            printf, lun, ''
```

```
    printf, lun, 'b_hat: '
    printf, lun, (*GSSData[squid])[snap].b_hat
    printf, lun, ''
    printf, lun, 'b_hat_dot: '
    printf, lun, (*GSSData[squid])[snap].b_hat_dot
    printf, lun, ''
    printf, lun, 'AftMuxIndex: '
    printf, lun, (*GSSData[squid])[snap].AftMuxIndex
    printf, lun, ''
    printf, lun, 'AftMuxValue: '
    printf, lun, (*GSSData[squid])[snap].AftMuxValue
    printf, lun, ''
    printf, lun, 'cAxisPosition: '
    printf, lun, (*GSSData[squid])[snap].cAxisPosition
    printf, lun, ''
    printf, lun, 'cAxisControlEffort: '
    printf, lun, (*GSSData[squid])[snap].cAxisControlEffort
    printf, lun, ''
    printf, lun, 'VcPlus: '
    printf, lun, (*GSSData[squid])[snap].VcPlus
    printf, lun, ''
    printf, lun, 'VcMinus: '
    printf, lun, (*GSSData[squid])[snap].VcMinus
    printf, lun, ''
    printf, lun, 'c_hat: '
    printf, lun, (*GSSData[squid])[snap].c_hat
    printf, lun, ''
    printf, lun, 'c_hat_dot: '
    printf, lun, (*GSSData[squid])[snap].c_hat_dot
    printf, lun, ''
    printf, lun, 'FwdMuxIndex: '
    printf, lun, (*GSSData[squid])[snap].FwdMuxIndex
    printf, lun, ''
    printf, lun, 'FwdMuxValue: '
    printf, lun, (*GSSData[squid])[snap].FwdMuxValue
end

2:  begin ;scheduler

    printf, 'Vehicle Time: ', (*GSSData[squid])[snap].VehicleTime
    printf, lun, ''
    printf, 'aPrimIdleTime: '
    printf, (*GSSData[squid])[snap].aPrimIdleTime
    printf, lun, ''
    printf, 'aSecondIdleTime: '
    printf, (*GSSData[squid])[snap].aSecondIdleTime
    printf, lun, ''
    printf, 'aTenHzIndex: '
    printf, (*GSSData[squid])[snap].aTenHzIndex
    printf, lun, ''
    printf, 'aFiveHzIndex: '
    printf, (*GSSData[squid])[snap].aFiveHzIndex
    printf, lun, ''
    printf, 'aOneHzIndex: '
    printf, (*GSSData[squid])[snap].aOneHzIndex
    printf, lun, ''
    printf, 'aPrimTo10HzCounter: '
    printf, (*GSSData[squid])[snap].aPrimTo10HzCounter
    printf, lun, ''
    printf, 'aOneTo10HzCounter: '
    printf, (*GSSData[squid])[snap].aOneTo10HzCounter
    printf, lun, ''
    printf, 'aTenHzSyncFailCounter: '
    printf, (*GSSData[squid])[snap].aTenHzSyncFailCounter
    printf, lun, ''
    printf, 'aOneHzSyncFailCounter: '
    printf, (*GSSData[squid])[snap].aOneHzSyncFailCounter
    printf, lun, ''
    printf, 'aDSPIntCounter: '
    printf, (*GSSData[squid])[snap].aDSPIntCounter
    printf, lun, ''
    printf, 'bSecondIdleTime: '
    printf, (*GSSData[squid])[snap].bSecondIdleTime
    printf, lun, ''
    printf, 'bTenHzIndex: '
    printf, (*GSSData[squid])[snap].bTenHzIndex
    printf, lun, ''
    printf, 'bFiveHzIndex: '
    printf, (*GSSData[squid])[snap].bFiveHzIndex
    printf, lun, ''
    printf, 'bOneHzIndex: '
    printf, (*GSSData[squid])[snap].bOneHzIndex
    printf, lun, ''
    printf, 'bPrimTo10HzCounter: '
    printf, (*GSSData[squid])[snap].bPrimTo10HzCounter
    printf, lun, ''
```

```
    printf, lun, 'b_hat: '
    printf, lun, (*GSSData[squid])[snap].b_hat
    printf, lun, ''
    printf, lun, 'b_hat_dot: '
    printf, lun, (*GSSData[squid])[snap].b_hat_dot
    printf, lun, ''
    printf, lun, 'AftMuxIndex: '
    printf, lun, (*GSSData[squid])[snap].AftMuxIndex
    printf, lun, ''
    printf, lun, 'AftMuxValue: '
    printf, lun, (*GSSData[squid])[snap].AftMuxValue
    printf, lun, ''
    printf, lun, 'cAxisPosition: '
    printf, lun, (*GSSData[squid])[snap].cAxisPosition
    printf, lun, ''
    printf, lun, 'cAxisControlEffort: '
    printf, lun, (*GSSData[squid])[snap].cAxisControlEffort
    printf, lun, ''
    printf, lun, 'VcPlus: '
    printf, lun, (*GSSData[squid])[snap].VcPlus
    printf, lun, ''
    printf, lun, 'VcMinus: '
    printf, lun, (*GSSData[squid])[snap].VcMinus
    printf, lun, ''
    printf, lun, 'c_hat: '
    printf, lun, (*GSSData[squid])[snap].c_hat
    printf, lun, ''
    printf, lun, 'c_hat_dot: '
    printf, lun, (*GSSData[squid])[snap].c_hat_dot
    printf, lun, ''
    printf, lun, 'FwdMuxIndex: '
    printf, lun, (*GSSData[squid])[snap].FwdMuxIndex
    printf, lun, ''
    printf, lun, 'FwdMuxValue: '
    printf, lun, (*GSSData[squid])[snap].FwdMuxValue
end

2:  begin ;scheduler

    printf, 'Vehicle Time: ', (*GSSData[squid])[snap].VehicleTi
    printf, lun, ''
    printf, 'aPrimIdleTime: '
    printf, (*GSSData[squid])[snap].aPrimIdleTime
    printf, lun, ''
    printf, 'aSecondIdleTime: '
    printf, (*GSSData[squid])[snap].aSecondIdleTime
    printf, lun, ''
    printf, 'aTenHzIndex: '
    printf, (*GSSData[squid])[snap].aTenHzIndex
    printf, lun, ''
    printf, 'aFiveHzIndex: '
    printf, (*GSSData[squid])[snap].aFiveHzIndex
    printf, lun, ''
    printf, 'aOneHzIndex: '
    printf, (*GSSData[squid])[snap].aOneHzIndex
    printf, lun, ''
    printf, 'aPrimTo10HzCounter: '
    printf, (*GSSData[squid])[snap].aPrimTo10HzCounter
    printf, lun, ''
    printf, 'aOneTo10HzCounter: '
    printf, (*GSSData[squid])[snap].aOneTo10HzCounter
    printf, lun, ''
    printf, 'aTenHzSyncFailCounter: '
    printf, (*GSSData[squid])[snap].aTenHzSyncFailCounter
    printf, lun, ''
    printf, 'aOneHzSyncFailCounter: '
    printf, (*GSSData[squid])[snap].aOneHzSyncFailCounter
    printf, lun, ''
    printf, 'aDSPIntCounter: '
    printf, (*GSSData[squid])[snap].aDSPIntCounter
    printf, lun, ''
    printf, 'bSecondIdleTime: '
    printf, (*GSSData[squid])[snap].bSecondIdleTime
    printf, lun, ''
    printf, 'bTenHzIndex: '
    printf, (*GSSData[squid])[snap].bTenHzIndex
    printf, lun, ''
    printf, 'bFiveHzIndex: '
    printf, (*GSSData[squid])[snap].bFiveHzIndex
    printf, lun, ''
    printf, 'bOneHzIndex: '
    printf, (*GSSData[squid])[snap].bOneHzIndex
    printf, lun, ''
    printf, 'bPrimTo10HzCounter: '
    printf, (*GSSData[squid])[snap].bPrimTo10HzCounter
    printf, lun, ''
```

```
                printf, 'bOneTo10HzCounter: '
                printf, (*GSSData[squid])[snap].bOneTo10HzCounter
                printf, lun, ''
                printf, 'bTenHzSyncFailCounter: '
                printf, (*GSSData[squid])[snap].bTenHzSyncFailCounter
                printf, lun, ''
                printf, 'bOneHzSyncFailCounter: '
                printf, (*GSSData[squid])[snap].bOneHzSyncFailCounter
                printf, lun, ''
                printf, 'bDSPIntCounter: '
                printf, (*GSSData[squid])[snap].bDSPIntCounter
                printf, lun, ''
                printf, 'cSecondIdleTime: '
                printf, (*GSSData[squid])[snap].cSecondIdleTime
                printf, lun, ''
                printf, 'cTenHzIndex: '
                printf, (*GSSData[squid])[snap].cTenHzIndex
                printf, lun, ''
                printf, 'cFiveHzIndex: '
                printf, (*GSSData[squid])[snap].cFiveHzIndex
                printf, lun, ''
                printf, 'cOneHzIndex: '
                printf, (*GSSData[squid])[snap].cOneHzIndex
                printf, lun, ''
                printf, 'cPrimTo10HzCounter: '
                printf, (*GSSData[squid])[snap].cPrimTo10HzCounter
                printf, lun, ''
                printf, 'cOneTo10HzCounter: '
                printf, (*GSSData[squid])[snap].cOneTo10HzCounter
                printf, lun, ''
                printf, 'cTenHzSyncFailCounter: '
                printf, (*GSSData[squid])[snap].cTenHzSyncFailCounter
                printf, lun, ''
                printf, 'cOneHzSyncFailCounter: '
                printf, (*GSSData[squid])[snap].cOneHzSyncFailCounter
                printf, lun, ''
                printf, 'cDSPIntCounter: '
                printf, (*GSSData[squid])[snap].cDSPIntCounter
            end
        ENDCASE

        FREE_LUN, lun

        if display_options[0] eq 1 then begin
            spawn, '/usr/openwin/bin/textedit -read_only -Ws 1000 500 ' + outputfi
        endif

        ;if display_options[1] eq 1 then begin
        ;    cmd = 'lpr -P ' + printer_list[printer] + ' ' + outputfile
        ;    SPAWN, cmd
        ;endif

        if display_options[2] ne 1 then begin  ;get rid of file we just created if
            spawn, 'rm ' + outputfile
        endif

        CATCH, /CANCEL

      endfor
    endfor
  end

ENDCASE
END


PRO USE_SNAP_ID
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mode, sre
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tre_snap_
    GSSData, snap_id_id, do_all, did_output
  COMMON WIDGET_IDS, subsystem_id, squid_number_id, sre_snap_type_id, squid_mode_id, tre_snap_ty

  bad_id = 0

  WIDGET_CONTROL, snap_id_id, GET_VALUE = snap_id
  snap_id = fix(array2str(snap_id), TYPE = 2)

  if snap_id ge 1 and snap_id le 18 then begin                    ;SRE
      sub_system = 0
      WIDGET_CONTROL, subsystem_id, SET_DROPLIST_SELECT = sub_system
      SNAPSHOT_SUBSYSTEM_COLUMN
      if snap_id ge 1 and snap_id le 8 then begin   ;1kHz
          sre_snap_type = 1
```

```
                printf, 'bOneTo10HzCounter: '
                printf, (*GSSData[squid])[snap].bOneTo10HzCounter
                printf, lun, ''
                printf, 'bTenHzSyncFailCounter: '
                printf, (*GSSData[squid])[snap].bTenHzSyncFailCounter
                printf, lun, ''
                printf, 'bOneHzSyncFailCounter: '
                printf, (*GSSData[squid])[snap].bOneHzSyncFailCounter
                printf, lun, ''
                printf, 'bDSPIntCounter: '
                printf, (*GSSData[squid])[snap].bDSPIntCounter
                printf, lun, ''
                printf, 'cSecondIdleTime: '
                printf, (*GSSData[squid])[snap].cSecondIdleTime
                printf, lun, ''
                printf, 'cTenHzIndex: '
                printf, (*GSSData[squid])[snap].cTenHzIndex
                printf, lun, ''
                printf, 'cFiveHzIndex: '
                printf, (*GSSData[squid])[snap].cFiveHzIndex
                printf, lun, ''
                printf, 'cOneHzIndex: '
                printf, (*GSSData[squid])[snap].cOneHzIndex
                printf, lun, ''
                printf, 'cPrimTo10HzCounter: '
                printf, (*GSSData[squid])[snap].cPrimTo10HzCounter
                printf, lun, ''
                printf, 'cOneTo10HzCounter: '
                printf, (*GSSData[squid])[snap].cOneTo10HzCounter
                printf, lun, ''
                printf, 'cTenHzSyncFailCounter: '
                printf, (*GSSData[squid])[snap].cTenHzSyncFailCounter
                printf, lun, ''
                printf, 'cOneHzSyncFailCounter: '
                printf, (*GSSData[squid])[snap].cOneHzSyncFailCounter
                printf, lun, ''
                printf, 'cDSPIntCounter: '
                printf, (*GSSData[squid])[snap].cDSPIntCounter
            end
        ENDCASE

        FREE_LUN, lun

        if display_options[0] eq 1 then begin
            spawn, '/usr/openwin/bin/textedit -read_only -Ws 1000 500 ' + o
        endif
        if display_options[1] eq 1 then begin
            cmd = 'lpr -P ' + printer_list[printer] + ' ' + outputfile
            SPAWN, cmd
        endif


        if display_options[2] ne 1 then begin  ;get rid of file we just cre
            spawn, 'rm ' + outputfile
        endif

        CATCH, /CANCEL

      endfor
    endfor
  end

ENDCASE
endif                          ;if print to file selected...

did_output = 1

END

PRO USE_SNAP_ID
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mo
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tr
    GSSData, snap_id_id, do_all, did_output
  COMMON WIDGET_IDS, subsystem_id, squid_number_id, sre_snap_type_id, squid_mode_id, tre_

  bad_id = 0

  WIDGET_CONTROL, snap_id_id, GET_VALUE = snap_id
  snap_id = fix(array2str(snap_id), TYPE = 2)

  if snap_id ge 1 and snap_id le 18 then begin                    ;SRE
      sub_system = 0
      WIDGET_CONTROL, subsystem_id, SET_DROPLIST_SELECT = sub_system
      SNAPSHOT_SUBSYSTEM_COLUMN
      if snap_id ge 1 and snap_id le 8 then begin   ;1kHz
          sre_snap_type = 1
```

cmel 3/2

SU GP-B INSP 31

474

32

```
        WIDGET_CONTROL, SRE_SNAP_TYPE_ID, SET_VALUE = 1
    endif else begin
        if snap_id ge 11 and snap_id le 18 then begin  ;20Hz
            sre_snap_type = 20
            WIDGET_CONTROL, SRE_SNAP_TYPE_ID, SET_VALUE = 0
        endif
    endelse
    if (snap_id ge 1 and snap_id le 4) or (snap_id ge 11 and snap_id le 14) then begin
        squid_mode = 1  ;default
        WIDGET_CONTROL, SQUID_MODE_ID, SET_VALUE = 0
    endif else begin
        if (snap_id ge 5 and snap_id le 8) or (snap_id ge 15 and snap_id le 18) then begin
            squid_mode = 2  ;locked
            WIDGET_CONTROL, SQUID_MODE_ID, SET_VALUE = 1
        endif
    endelse
    if snap_id eq 1 or snap_id eq 5 or snap_id eq 11 or snap_id eq 15 then begin
        squid_number = 1
        WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 1
    endif else begin
        if snap_id eq 2 or snap_id eq 6 or snap_id eq 12 or snap_id eq 16 then begin
            squid_number = 2
            WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 2
        endif else begin
            if snap_id eq 3 or snap_id eq 7 or snap_id eq 13 or snap_id eq 17 then begin
                squid_number = 3
                WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 3
            endif else begin
                if snap_id eq 4 or snap_id eq 8 or snap_id eq 14 or snap_id eq 18 then begin
                    squid_number = 4
                    WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 4
                endif
            endelse
        endelse
    endelse

endif else begin
    if snap_id ge 51 and snap_id le 56 then begin                ;TRE
        sub_system = 1
        WIDGET_CONTROL, subsystem_id, SET_DROPLIST_SELECT = sub_system
        SNAPSHOT_SUBSYSTEM_COLUMN
        if snap_id eq 51 then begin
            tre_snap_type = 1
            WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 0
        endif else begin
            if snap_id eq 52 then begin
                tre_snap_type = 2
                WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 1
            endif else begin
                if snap_id eq 53 then begin
                    tre_snap_type = 3
                    WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 2
                endif else begin
                    if snap_id eq 54 then begin
                        tre_snap_type = 4
                        WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 3
                    endif else begin
                        if snap_id eq 55 then begin
                            tre_snap_type = 5
                            WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 4
                        endif else begin
                            if snap_id eq 56 then begin
                                tre_snap_type = 6
                                WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 5
                            endif
                        endelse
                    endelse
                endelse
            endelse
        endelse

    endif else begin
        if snap_id ge 101 and snap_id le 135 then begin          ;GSS
            sub_system = 2
            WIDGET_CONTROL, subsystem_id, SET_DROPLIST_SELECT = sub_system
            SNAPSHOT_SUBSYSTEM_COLUMN
            if (snap_id ge 101 and snap_id le 103) or (snap_id ge 110 and snap_id le 112) $
              or (snap_id ge 120 and snap_id le 122) or (snap_id ge 130 and snap_id le 132) th
                gss_snap_type = 1
                WIDGET_CONTROL, GSS_SNAP_TYPE_ID, SET_VALUE = 0
            endif else begin
                gss_snap_type = 2
                WIDGET_CONTROL, GSS_SNAP_TYPE_ID, SET_VALUE = 1
            endelse
            if snap_id ge 101 and snap_id le 106 then begin
                squid_number = 1
```

```
        WIDGET_CONTROL, SRE_SNAP_TYPE_ID, SET_VALUE = 1
    endif else begin
        if snap_id ge 11 and snap_id le 18 then begin  ;20Hz
            sre_snap_type = 20
            WIDGET_CONTROL, SRE_SNAP_TYPE_ID, SET_VALUE = 0
        endif
    endelse
    if (snap_id ge 1 and snap_id le 4) or (snap_id ge 11 and snap_id le 14) then begin
        squid_mode = 1  ;default
        WIDGET_CONTROL, SQUID_MODE_ID, SET_VALUE = 0
    endif else begin
        if (snap_id ge 5 and snap_id le 8) or (snap_id ge 15 and snap_id le 18) then be
            squid_mode = 2  ;locked
            WIDGET_CONTROL, SQUID_MODE_ID, SET_VALUE = 1
        endif
    endelse
    if snap_id eq 1 or snap_id eq 5 or snap_id eq 11 or snap_id eq 15 then begin
        squid_number = 1
        WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 1
    endif else begin
        if snap_id eq 2 or snap_id eq 6 or snap_id eq 12 or snap_id eq 16 then begin
            squid_number = 2
            WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 2
        endif else begin
            if snap_id eq 3 or snap_id eq 7 or snap_id eq 13 or snap_id eq 17 then begi
                squid_number = 3
                WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 3
            endif else begin
                if snap_id eq 4 or snap_id eq 8 or snap_id eq 14 or snap_id eq 18 then
                    squid_number = 4
                    WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 4
                endif
            endelse
        endelse
    endelse

endif else begin
    if snap_id ge 51 and snap_id le 56 then begin                ;TRE
        sub_system = 1
        WIDGET_CONTROL, subsystem_id, SET_DROPLIST_SELECT = sub_system
        SNAPSHOT_SUBSYSTEM_COLUMN
        if snap_id eq 51 then begin
            tre_snap_type = 1
            WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 0
        endif else begin
            if snap_id eq 52 then begin
                tre_snap_type = 2
                WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 1
            endif else begin
                if snap_id eq 53 then begin
                    tre_snap_type = 3
                    WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 2
                endif else begin
                    if snap_id eq 54 then begin
                        tre_snap_type = 4
                        WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 3
                    endif else begin
                        if snap_id eq 55 then begin
                            tre_snap_type = 5
                            WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT = 4
                        endif else begin
                            if snap_id eq 56 then begin
                                tre_snap_type = 6
                                WIDGET_CONTROL, TRE_SNAP_TYPE_ID, SET_DROPLIST_SELECT =
                            endif
                        endelse
                    endelse
                endelse
            endelse
        endelse

    endif else begin
        if snap_id ge 101 and snap_id le 135 then begin          ;GSS
            sub_system = 2
            WIDGET_CONTROL, subsystem_id, SET_DROPLIST_SELECT = sub_system
            SNAPSHOT_SUBSYSTEM_COLUMN
            if (snap_id ge 101 and snap_id le 103) or (snap_id ge 110 and snap_id le 11
              or (snap_id ge 120 and snap_id le 122) or (snap_id ge 130 and snap_id le
                gss_snap_type = 1
                WIDGET_CONTROL, GSS_SNAP_TYPE_ID, SET_VALUE = 0
            endif else begin
                gss_snap_type = 2
                WIDGET_CONTROL, GSS_SNAP_TYPE_ID, SET_VALUE = 1
            endelse
            if snap_id ge 101 and snap_id le 106 then begin
                squid_number = 1
```

```
                  WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 1
              endif else begin
                  if snap_id ge 110 and snap_id le 115 then begin
                      squid_number = 2
                      WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 2
                  endif else begin
                      if snap_id ge 120 and snap_id le 125 then begin
                          squid_number = 3
                          WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 3
                      endif else begin
                          if snap_id ge 130 and snap_id le 135 then begin
                              squid_number = 4
                              WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 4
                          endif
                      endelse
                  endelse
              endelse
          endif else begin
                  tmp = DIALOG_MESSAGE('Snapshot ID is out of range.', /INFORMATION)
                  bad_id = 1
                  return
              endelse
      endelse
  endelse

END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PRO SNAPSHOT_EVENT, event

  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mode, sre
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tre_snap_
    GSSData, snap_id_id, do_all, did_output
  COMMON FIRST_AND_LAST_SNAPSHOTS, first_snapshot_id, last_snapshot_id
  COMMON WIDGET_IDS, subsystem_id, squid_number_id, sre_snap_type_id, squid_mode_id, tre_snap_ty
  COMMON NODATA, found_no_data

  ; Get the widget ID for the Display panel
  base = event.handler
  tmpfile = GETENV("HOME") + '/temp'

  ; Get the user value associated with the event we're processing and use
  ; it to determine what the user did to generate this event

  WIDGET_CONTROL, event.id, GET_UVALUE=uval
  WIDGET_CONTROL, event.top, GET_UVALUE=display_state


    case uval of

        "CANCEL": begin

            spawn, 'rm ' + GETENV("HOME") + '/temp.*'  ; Get rid of my junk files from last run
             ; Clicked on Close Button.  Erase the panel and return
            WIDGET_CONTROL, base, /DESTROY

            return
        end

        "TIMETYPE":   begin
            TCAD_DISPLAY__TIMETYPE, display_state
            accessed_squid = [-1]
        end

        "DISPLAY_OPTIONS": begin ;A display option was (un)checked
            display_options[event.value] = event.select ;Record which one.
        end

        "PRINTER": begin        ;A new printer was selected.
            printer = event.index ;Get the list index of the printer.
        end

        "FILENAME": begin       ;A filename was entered.
            WIDGET_CONTROL, filename_text_id, GET_VALUE = tmp

            filename = array2str(tmp)

            if (filename eq '') then begin
                tmp = DIALOG_MESSAGE('No filename provided', /INFORMATION)
                return
            endif
        end
```

```
                  WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 1
              endif else begin
                  if snap_id ge 110 and snap_id le 115 then begin
                      squid_number = 2
                      WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 2
                  endif else begin
                      if snap_id ge 120 and snap_id le 125 then begin
                          squid_number = 3
                          WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 3
                      endif else begin
                          if snap_id ge 130 and snap_id le 135 then begin
                              squid_number = 4
                              WIDGET_CONTROL, SQUID_NUMBER_ID, SET_DROPLIST_SELECT = 4
                          endif
                      endelse
                  endelse
              endelse
          endif else begin
                  tmp = DIALOG_MESSAGE('Snapshot ID is out of range.', /INFORMATION)
                  bad_id = 1
                  return
              endelse
      endelse
  endelse

END

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

PRO SNAPSHOT_EVENT, event

  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mo
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tr
    GSSData, snap_id_id, do_all, did_output
  COMMON FIRST_AND_LAST_SNAPSHOTS, first_snapshot_id, last_snapshot_id
  COMMON WIDGET_IDS, subsystem_id, squid_number_id, sre_snap_type_id, squid_mode_id, tre_
  COMMON NODATA, found_no_data

  ; Get the widget ID for the Display panel
  base = event.handler
  tmpfile = GETENV("HOME") + '/temp'

  ; Get the user value associated with the event we're processing and use
  ; it to determine what the user did to generate this event

  WIDGET_CONTROL, event.id, GET_UVALUE=uval
  WIDGET_CONTROL, event.top, GET_UVALUE=display_state


    case uval of

        "CANCEL": begin

            spawn, 'rm ' + GETENV("HOME") + '/temp.*'  ; Get rid of my junk files from la
             ; Clicked on Close Button.  Erase the panel and return
            WIDGET_CONTROL, base, /DESTROY

            return
        end

        "TIMETYPE":   begin
            TCAD_DISPLAY__TIMETYPE, display_state
            accessed_squid = [-1]
        end

        "DISPLAY_OPTIONS": begin ;A display option was (un)checked
            display_options[event.value] = event.select ;Record which one.
        end

        "PRINTER": begin        ;A new printer was selected.
            printer = event.index ;Get the list index of the printer.
        end

        "FILENAME": begin       ;A filename was entered.
            WIDGET_CONTROL, filename_text_id, GET_VALUE = tmp

            filename = array2str(tmp)

            if (filename eq '') then begin
                tmp = DIALOG_MESSAGE('No filename provided', /INFORMATION)
                return
            endif
        end
```

```
"SUBSYSTEM": begin
    sub_system = event.index ; get index of subsystem selected
    SNAPSHOT_SUBSYSTEM_COLUMN
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
    accessed_squid = [-1]
end

"SRE_SNAPTYPE": begin
    if event.value then begin ;selected 1 kHz
        sre_snap_type = 1
    endif else begin    ;selected 20 Hz
        sre_snap_type = 20
    endelse
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
    accessed_squid = [-1]
end

"TRE_SNAPTYPE": begin
    tre_snap_type = event.index         ;returns a LONG integer for some reason
    tre_snap_type = fix(tre_snap_type, TYPE = 2) + 1   ;convert to normal integer
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
end

"SQUIDNUMBER": begin
    squid_number = event.index ;returns a LONG integer for some reason
    squid_number = fix(squid_number, TYPE = 2) ;convert to normal integer
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
end

"SQUIDMODE": begin
    if event.value then begin
        squid_mode = 2
    endif else begin
        squid_mode = 1
    endelse
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
    accessed_squid = [-1]
end

"GSS_SNAPTYPE": begin
    if event.value then begin
        gss_snap_type = 2
    endif else begin
        gss_snap_type = 1
    endelse
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
end

else: begin

    IF (uval eq "OKAY") or (uval eq "FIRSTSNAPSHOT") or (uval eq "LASTSNAPSHOT") or (uva
      or (uval eq "SNAP_ID") then begin

        spawn, 'rm ' + GETENV("HOME") + '/temp.*' ; Get rid of my junk files from last r

        WIDGET_CONTROL, snap_id_id, GET_VALUE = snap_id

        if array2str(snap_id) ne '' then USE_SNAP_ID

        use_default_times = 0

        if not use_default_times then begin
            TCAD_INPUT_TIMES__GET, display_state
            if not (*display_state).times_valid then return
        endif

        ;Generate the different command strings for the C scripts.
        ;Need OutputFileName, t1, t2, cycle, snapType, squidNumber, squidNode as options

        SRE_cmd = '/home/science/utilities/database/SQUID_Snapshot '
        TRE_cmd = '/home/science/utilities/database/TRE_Snapshot '
        GSS_cmd = '/home/science/utilities/database/GSS_Snapshot '
```

```
"SUBSYSTEM": begin
    sub_system = event.index ; get index of subsystem selected
    SNAPSHOT_SUBSYSTEM_COLUMN
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
    accessed_squid = [-1]
end

"SRE_SNAPTYPE": begin
    if event.value then begin ;selected 1 kHz
        sre_snap_type = 1
    endif else begin    ;selected 20 Hz
        sre_snap_type = 20
    endelse
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
    accessed_squid = [-1]
end

"TRE_SNAPTYPE": begin
    tre_snap_type = event.index         ;returns a LONG integer for some reason
    tre_snap_type = fix(tre_snap_type, TYPE = 2) + 1   ;convert to normal integer
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
end

"SQUIDNUMBER": begin
    squid_number = event.index ;returns a LONG integer for some reason
    squid_number = fix(squid_number, TYPE = 2) ;convert to normal integer
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
end

"SQUIDMODE": begin
    if event.value then begin
        squid_mode = 2
    endif else begin
        squid_mode = 1
    endelse
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
    accessed_squid = [-1]
end

"GSS_SNAPTYPE": begin
    if event.value then begin
        gss_snap_type = 2
    endif else begin
        gss_snap_type = 1
    endelse
    ;WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ''
    ;WIDGET_CONTROL, last_snapshot_id, SET_VALUE = ''
    WIDGET_CONTROL, snap_id_id, SET_VALUE = ''
end

else: begin

    IF (uval eq "OKAY") or (uval eq "FIRSTSNAPSHOT") or (uval eq "LASTSNAPSHOT")
      or (uval eq "SNAP_ID") then begin

        spawn, 'rm ' + GETENV("HOME") + '/temp.*' ; Get rid of my junk files from

        WIDGET_CONTROL, snap_id_id, GET_VALUE = snap_id

        if array2str(snap_id) ne '' then USE_SNAP_ID

        ;  Clicked on OKAY button or pressed return.  Run the C script and do our

        TCAD_INPUT_TIMES__GET, display_state
        if not (*display_state).times_valid then return

        ;Generate the different command strings for the C scripts.
        ;Need OutputFileName, t1, t2, cycle, snapType, squidNumber, squidNode as

        SRE_cmd = '/home/science/utilities/database/SQUID_Snapshot '
        TRE_cmd = '/home/science/utilities/database/TRE_Snapshot '
        GSS_cmd = '/home/science/utilities/database/GSS_Snapshot '
```

SU GP-B INSP 31

```
;Add OutputFileName (defined above), t1, t2, and cycle to command line                                 ;Add OutputFileName (defined above), t1, t2, and cycle to command line
; The ,2 option for strtrim makes it remove both leading and trailing                                                  ; The ,2 option for strtrim makes it remove both leading
; spaces (all of them)                                                                                                 ; spaces (all of them)

t1 = strtrim((*display_state).start_sct.vtcw/10, 2)                                                     t1 = strtrim((*display_state).start_sct.vtcw/10, 2)
t2 = strtrim((*display_state).stop_sct.vtcw/10, 2)                                                      t2 = strtrim((*display_state).stop_sct.vtcw/10, 2)
cycle = strtrim((*display_state).start_sct.cycle, 2)                                                    cycle = strtrim((*display_state).start_sct.cycle, 2)
SRE_cmd = SRE_cmd + tmpfile + ' ' + t1 + ' ' + t2 + ' ' + cycle                                         SRE_cmd = SRE_cmd + tmpfile + ' ' + t1 + ' ' + t2 + ' ' + cycle
TRE_cmd = TRE_cmd + tmpfile + ' ' + t1 + ' ' + t2 + ' ' + cycle                                         TRE_cmd = TRE_cmd + tmpfile + ' ' + t1 + ' ' + t2 + ' ' + cycle
GSS_cmd = GSS_cmd + tmpfile + ' ' + t1 + ' ' + t2 + ' ' + cycle                                         GSS_cmd = GSS_cmd + tmpfile + ' ' + t1 + ' ' + t2 + ' ' + cycle

;Add sre_snap_type, squid_number, squid_mode to SRE command line                    |  ;                 ;Add sre_snap_type, squid_number, squid_mode to SRE command line
s1 = strtrim(sre_snap_type, 2)                                                                          s1 = strtrim(sre_snap_type, 2)
s2 = strtrim(squid_number, 2)                                                                           s2 = strtrim(squid_number, 2)
s3 = strtrim(squid_mode, 2)                                                                             s3 = strtrim(squid_mode, 2)
SRE_cmd = SRE_cmd + ' ' + s1 + ' ' + s2 + ' ' + s3                                                      SRE_cmd = SRE_cmd + ' ' + s1 + ' ' + s2 + ' ' + s3

;Add tre_snap_type to TRE command line                                              |  ;                 ;Add tre_snap_type to TRE command line
s4 = strtrim(tre_snap_type, 2)                                                                          s4 = strtrim(tre_snap_type, 2)
TRE_cmd = TRE_cmd + ' ' + s4                                                                            TRE_cmd = TRE_cmd + ' ' + s4

;Add gss_snap_type, gss_number to GSS command line                                  |  ;                 ;Add gss_snap_type, gss_number to GSS command line
s5 = strtrim(gss_snap_type, 2)                                                                          s5 = strtrim(gss_snap_type, 2)
GSS_cmd = GSS_cmd + ' ' + s5 + ' ' + s2                                                                 GSS_cmd = GSS_cmd + ' ' + s5 + ' ' + s2

                                                                                    <
; Temporarily get rid of start/stop stuff so I don't have to type it                                   ; Temporarily get rid of start/stop stuff so I don't have to type it
; in each time.                                                                                        ; in each time.

                                                                                    >  ;                SRE_cmd = SRE_cmd + tmpfile + ' ' + '117536400' + ' ' + '117680400' + '
                                                                                    >  ;                TRE_cmd = TRE_cmd + tmpfile + ' ' + '0' + ' ' + '57600' + ' '+ '-2433' +
                                                                                    >  ;                GSS_cmd = GSS_cmd + tmpfile + ' ' + '117536400' + ' ' + '117680400' + '
                                                                                    >  ;cycle = -4500 ; temp
                                                                                    >                  print, SRE_cmd
                                                                                    >                  print, TRE_cmd
                                                                                    >                  print, GSS_cmd

 if use_default_times then begin                                                    <
     SRE_cmd = '/home/science/utilities/database/SQUID_Snapshot ' + tmpfile + '     <
               '117536400' + ' ' + '117680400' + ' '+ '-4500' + ' ' + s1 + ' ' +    <
     TRE_cmd = '/home/science/utilities/database/TRE_Snapshot ' + tmpfile + ' '     <
               '0' + ' ' + '57600' + ' '+ '-2433' + ' ' + s4                        <
     GSS_cmd = '/home/science/utilities/database/GSS_Snapshot ' + tmpfile + ' '     <
               '117536400' + ' ' + '117680400' + ' '+ '-4500' + ' ' + s5 + ' ' +    <
     cycle = -4500                                                                  <
 endif                                                                              <
                                                                                    <
                                                                                    <
;print, SRE_cmd                                                                     <
;print, TRE_cmd                                                                     <
;print, GSS_cmd                                                                     <
                                                                                    <
;Assume we don't want VERBOSE option.                                                                  ;Assume we don't want VERBOSE option.
;Otherwise, would have to add a 1 to cmd.                                                              ;Otherwise, would have to add a 1 to cmd.

;Create an array (accessed_squid) to hold which squid data has already been read                       ;Create an array (accessed_squid) to hold which squid data has already be
;database, so it doesn't access the db every time the user changes which snapsho                       ;database, so it doesn't access the db every time the user changes which

                ; should have an error check to see that the user actually input                                       ; should have an error check to see that the user actuall
WIDGET_CONTROL, first_snapshot_id, GET_VALUE = first_snapshot                                          WIDGET_CONTROL, first_snapshot_id, GET_VALUE = first_snapshot
WIDGET_CONTROL, last_snapshot_id, GET_VALUE = last_snapshot                                            WIDGET_CONTROL, last_snapshot_id, GET_VALUE = last_snapshot

if not bad_id then begin                                                                               if not bad_id then begin

    if not RangeIsGood(first_snapshot, last_snapshot) then return                                          if not RangeIsGood(first_snapshot, last_snapshot) then return

    CASE sub_system OF                                                                                     CASE sub_system OF
        0:  begin ;SRE                                                                                         0:  begin ;SRE
            if not IsInArray(squid_number, accessed_squid) then begin                                              if not IsInArray(squid_number, accessed_squid) then begin
                accessed_squid = [accessed_squid, squid_number]                                                        accessed_squid = [accessed_squid, squid_number]
                ;Execute the script.                                                                                   ;Execute the script.
                SPAWN, SRE_cmd                                                                                         SPAWN, SRE_cmd
                PROCESS_SNAPSHOT_FILE                                                                                  PROCESS_SNAPSHOT_FILE
                if found_no_data then return                                                                           if found_no_data then return
            endif                                                                                                  endif

            if squid_number eq 0 then accessed_squid = [0, 1, 2, 3, 4]                                            if squid_number eq 0 then accessed_squid = [0, 1, 2, 3, 4]
        end                                                                                                    end

        1:  begin ;TRE                                                                                         1:  begin ;TRE
            SPAWN, TRE_cmd                                                                                         SPAWN, TRE_cmd
            PROCESS_SNAPSHOT_FILE                                                                                  PROCESS_SNAPSHOT_FILE
            if found_no_data then return                                                                           if found_no_data then return
        end                                                                                                    end

        2:  begin ;GSS                                                                                         2:  begin ;GSS
```

Printing routine cleanup MCR 3/2

NAH

```
        if not IsInArray(squid_number, accessed_squid) then begin
            accessed_squid = [accessed_squid, squid_number]
            ;print, accessed_squid
            ;Execute the script.
            SPAWN, GSS_cmd
            PROCESS_SNAPSHOT_FILE
            if found_no_data then return
        endif
    end
ENDCASE
```

```
            ; Now a bit of code to make the first/last snapshot behave corre
            ; First, consider case where both fields are empty and only the
            ; Check if there are too many snapshots

        totalNumSnapshots = FIX(TOTAL(nSnapshots))

        if ARRAY_EQUAL(display_options, [0, 0, 1]) and totalNumSnapshots ne 0
                ; The ARRAY_EQUAL function here is equivalent to Matlab's isem

            tmp = strcompress(string('Save all ', totalNumSnapshots, ' snapsho
            tmp = DIALOG_MESSAGE(tmp, /QUESTION, /DEFAULT_NO)

            if (tmp eq 'Yes') then begin ;handle case where squid_number = 0 (
                if (sub_system eq 0 or sub_system eq 2) then begin
                    if squid_number ne 0 then begin
                        first_snapshot = '1'
                        last_snapshot = strtrim(array2str(nSnapshots[squid_num
                        WIDGET_CONTROL, last_snapshot_id, SET_VALUE = last_sna
                        WIDGET_CONTROL, first_snapshot_id, SET_VALUE = first_s
                    endif else begin
                        do_all = 1
                    endelse
                endif else begin  ; (in TRE state)
                    first_snapshot = '1'
                    last_snapshot = strtrim(nSnapshots, 2)
                    WIDGET_CONTROL, last_snapshot_id, SET_VALUE = last_snapsho
                    WIDGET_CONTROL, first_snapshot_id, SET_VALUE = first_snaps
                endelse

                DO_OUTPUT
                return
            endif else return

        endif else begin ;first/last fields still empty, but now one of the ot
            if totalNumSnapshots gt 20 then begin
                tmp = 'Too many (' + strtrim(totalNumSnapshots, 2) + $
                  ') snapshots selected. Display/Print a maximum of 20 at a ti
                tmp = DIALOG_MESSAGE(tmp, /INFORMATION)
                return
            endif else begin
                if totalNumSnapshots ne 0 then begin
                    if sub_system eq 0 then begin
                        last_snapshot = strtrim(array2str(nSnapshots[squid_num
                    endif else begin
                        last_snapshot = strtrim(nSnapshots, 2)
                    endelse
                    first_snapshot = '1'
                    WIDGET_CONTROL, last_snapshot_id, SET_VALUE = last_snapsho
                    WIDGET_CONTROL, first_snapshot_id, SET_VALUE = first_snaps
                    DO_OUTPUT
                    return
                endif
            endelse
        endelse

    ; at least one of the first/last fields is NOT empty
        if not did_output then begin
            ;if squid_number eq 0 and (sub_system eq 0 or sub_system eq 2) and d
            ;    tmp = 'Please select just one SQUID/GSS number for display.'
            ;    tmp = DIALOG_MESSAGE(tmp, /INFORMATION)
            ;    return
            ;endif else begin
                ; if only one empty, set them both to the same value
                if (ARRAY_EQUAL(last_snapshot, '')) then begin
                    WIDGET_CONTROL, last_snapshot_id, SET_VALUE = array2str(firs
                    last_snapshot = array2str(first_snapshot)
                    DO_OUTPUT
                endif else begin
                    if (ARRAY_EQUAL(first_snapshot, '')) then begin
                        WIDGET_CONTROL, first_snapshot_id, SET_VALUE = array2str
                        first_snapshot = array2str(last_snapshot)
```

```
        if not IsInArray(squid_number, accessed_squid) then begin
            accessed_squid = [accessed_squid, squid_number]
            ;print, accessed_squid
            ;Execute the script.
            SPAWN, GSS_cmd
            PROCESS_SNAPSHOT_FILE
            if found_no_data then return
        endif
    end
ENDCASE
```

```
            ; Now a bit of code to make the first/last snapshot behav
            ; First, consider case where both fields are empty and on
            ; Check if there are too many snapshots

        totalNumSnapshots = FIX(TOTAL(nSnapshots))

        if ARRAY_EQUAL(display_options, [0, 0, 1]) and totalNumSnapshot
                ; The ARRAY_EQUAL function here is equivalent to Matlab

            tmp = strcompress(string('Save all ', totalNumSnapshots, '
            tmp = DIALOG_MESSAGE(tmp, /QUESTION, /DEFAULT_NO)

            if (tmp eq 'Yes') then begin ;handle case where squid_numbe
                if (sub_system eq 0 or sub_system eq 2) then begin
                    if squid_number ne 0 then begin
                        first_snapshot = '1'
                        last_snapshot = strtrim(array2str(nSnapshots[sq
                        WIDGET_CONTROL, last_snapshot_id, SET_VALUE = l
                        WIDGET_CONTROL, first_snapshot_id, SET_VALUE =
                    endif else begin
                        do_all = 1
                    endelse
                endif else begin  ; (in TRE state)
                    first_snapshot = '1'
                    last_snapshot = strtrim(nSnapshots, 2)
                    WIDGET_CONTROL, last_snapshot_id, SET_VALUE = last_
                    WIDGET_CONTROL, first_snapshot_id, SET_VALUE = firs
                endelse

                DO_OUTPUT
                return
            endif else return

        endif else begin ;first/last fields still empty, but now one of
            if totalNumSnapshots gt 20 then begin
                tmp = 'Too many (' + strtrim(totalNumSnapshots, 2) + $
                  ') snapshots selected. Display/Print a maximum of 20
                tmp = DIALOG_MESSAGE(tmp, /INFORMATION)
                return
            endif else begin
                if totalNumSnapshots ne 0 then begin
                    if sub_system eq 0 then begin
                        last_snapshot = strtrim(array2str(nSnapshots[sq
                    endif else begin
                        last_snapshot = strtrim(nSnapshots, 2)
                    endelse
                    first_snapshot = '1'
                    WIDGET_CONTROL, last_snapshot_id, SET_VALUE = last_
                    WIDGET_CONTROL, first_snapshot_id, SET_VALUE = firs
                    DO_OUTPUT
                    return
                endif
            endelse
        endelse

    ; at least one of the first/last fields is NOT empty
        if not did_output then begin
            ;if squid_number eq 0 and (sub_system eq 0 or sub_system eq 2
            ;    tmp = 'Please select just one SQUID/GSS number for displ
            ;    tmp = DIALOG_MESSAGE(tmp, /INFORMATION)
            ;    return
            ;endif else begin
                ; if only one empty, set them both to the same value
                if (ARRAY_EQUAL(last_snapshot, '')) then begin
                    WIDGET_CONTROL, last_snapshot_id, SET_VALUE = array2s
                    last_snapshot = array2str(first_snapshot)
                    DO_OUTPUT
                endif else begin
                    if (ARRAY_EQUAL(first_snapshot, '')) then begin
                        WIDGET_CONTROL, first_snapshot_id, SET_VALUE = ar
                        first_snapshot = array2str(last_snapshot)
```

SU GP-B INSP 31

37

```
                                DO_OUTPUT
                            endif else begin
                                totalNumSnapshots = fix(last_snapshot[0]) - fix(first_sn
                                if totalNumSnapshots gt 20 and (display_options[0] ne 0
                                    tmp = 'Too many (' + strtrim(totalNumSnapshots, 2) +
                                      ') snapshots selected. Display/Print a maximum of
                                    tmp = DIALOG_MESSAGE(tmp, /INFORMATION)
                                    return
                                endif else DO_OUTPUT
                            endelse
                        endelse
                    ;endelse
                endif
            ;endelse
        endif           ;   if not bad_id
    ENDIF ELSE BEGIN
        TCAD_AUTOFILL_TIMES, event
    ENDELSE
  end     ; the "else: begin" case, that is, clicked OK or enter
ENDCASE

bad_id = 0
do_all = 0
did_output = 0
;  Note: Need to add check to see that user did not input snapshots less than
;  or equal to zero.


END


;*************************************************************************
;
; SNAPSHOT - Create the Display Data dialog box
;
;*************************************************************************

PRO SNAPSHOT, GROUP_LEADER=group_leader
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mode, sre
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tre_snap_
    GSSData, snap_id_id, do_all, did_output
  COMMON WIDGET_IDS, subsystem_id, squid_number_id, sre_snap_type_id, squid_mode_id, tre_snap_ty
  COMMON FIRST_AND_LAST_SNAPSHOTS, first_snapshot_id, last_snapshot_id

  !EXCEPT = 0   ; suppress "error" messages such as 'floating underflow'

  nSnapshots = lonarr(1, 4) ;array containing number of snapshots for each squid
  SquidData = PTRARR(1, 4) ;pointer array since there are different numbers of snapshots per squ
  GSSData = PTRARR(1, 4) ;pointer array since there are different numbers of snapshots per squid


  ;Default our menu settings.
  time_format = "yes"            ;Default to time conversion.
  printer =0                     ;Default to first printer.

  ;Log into the database, if we aren't already
  if not !dbi_connected then DB_CONNECT, "gpb","gravitydb", server="GPB_DATA"

  ; Initialize the printer information, if needed
  if N_ELEMENTS (printer_list) eq 0 then TCAD_INITIAL_PRINTER

  ; Use this as a flag to see if we should access the database again
  ; or if we've already got the data for the chosen squid
  ; Each time something is changed (like subsystem, or squidmode), this
  ; array gets set back to -1.

  accessed_squid = [-1]   ;arbitrary number greater than 4 (max number of squids)
  bad_id = 0
  do_all = 0
  did_output = 0

  ; Define the structure used to store the panel object's data
  state = {base:                0L, $
           start_time_widget:   0L, $
           stop_time_widget:    0L, $
           timetype_widget:     0L, $
           timetype:             1, $
           changed:              0, $
           time_window:         0L, $
           cycle:                0, $
           start_sct:    NEW_SCT(), $
           stop_sct:     NEW_SCT(), $
           printer_droplist:    0L, $
           times_valid:          0 }
```

```
                                DO_OUTPUT
                            endif else begin
                                totalNumSnapshots = fix(last_snapshot[0]) - fix(f
                                if totalNumSnapshots gt 20 and (display_options[0
                                    tmp = 'Too many (' + strtrim(totalNumSnapshot
                                      ') snapshots selected. Display/Print a maxi
                                    tmp = DIALOG_MESSAGE(tmp, /INFORMATION)
                                    return
                                endif else DO_OUTPUT
                            endelse
                        endelse
                    ;endelse
                endif
            ;endelse
        endif           ;   if not bad_id
    ENDIF ELSE BEGIN
        TCAD_AUTOFILL_TIMES, event
    ENDELSE
  end     ; the "else: begin" case, that is, clicked OK or enter
ENDCASE

bad_id = 0
do_all = 0
did_output = 0
;  Note: Need to add check to see that user did not input snapshots less than
;  or equal to zero.


END


;*************************************************************************
;
; SNAPSHOT - Create the Display Data dialog box
;
;*************************************************************************

PRO SNAPSHOT, GROUP_LEADER=group_leader
  COMMON TCAD_PRINTER_INFO, printer_list
  COMMON SNAPSHOT_INFO, display_options, printer, filename
  COMMON SNAPSHOT_OPTIONS_STUFF, sub_system, base4, base3, base4x, squid_number, squid_mo
    SquidData, filename_text_id, first_snapshot, last_snapshot, cycle, accessed_squid, tr
    GSSData, snap_id_id, do_all, did_output
  COMMON WIDGET_IDS, subsystem_id, squid_number_id, sre_snap_type_id, squid_mode_id, tre_
  COMMON FIRST_AND_LAST_SNAPSHOTS, first_snapshot_id, last_snapshot_id

<
<
<
<
<
<

  ;Default our menu settings.
  time_format = "yes"            ;Default to time conversion.
  printer =0                     ;Default to first printer.

  ;Log into the database, if we aren't already
  if not !dbi_connected then DB_CONNECT, "gpb","gravitydb", server="GPB_DATA"

  ; Initialize the printer information, if needed
  if N_ELEMENTS (printer_list) eq 0 then TCAD_INITIAL_PRINTER

  ; Use this as a flag to see if we should access the database again
  ; or if we've already got the data for the chosen squid
  ; Each time something is changed (like subsystem, or squidmode), this
  ; array gets set back to -1.

  accessed_squid = [-1]   ;arbitrary number greater than 4 (max number of squids)
  bad_id = 0
  do_all = 0
  did_output = 0

  ; Define the structure used to store the panel object's data
  state = {base:                0L, $
           start_time_widget:   0L, $
           stop_time_widget:    0L, $
           timetype_widget:     0L, $
           timetype:             1, $
           changed:              0, $
           time_window:         0L, $
           cycle:                0, $
           start_sct:    NEW_SCT(), $
           stop_sct:     NEW_SCT(), $
           printer_droplist:    0L, $
           times_valid:          0 }
```

MGK 312

38

```
  display_state = PTR_NEW(state,/NO_COPY)                                                                  display_state = PTR_NEW(state,/NO_COPY)
  ;Create and initialize the individual widgets, starting with the top-level                               ;Create and initialize the individual widgets, starting with the top-level
  ;base of the widget tree                                                                                 ;base of the widget tree

  (*display_state).base = WIDGET_BASE ( TITLE="Snapshot", /COLUMN)                                         (*display_state).base = WIDGET_BASE ( TITLE="Snapshot", /COLUMN)

  if KEYWORD_SET (GROUP_LEADER) then begin                                                                 if KEYWORD_SET (GROUP_LEADER) then begin
    WIDGET_CONTROL, (*display_state).base, GROUP_LEADER=group_leader                                         WIDGET_CONTROL, (*display_state).base, GROUP_LEADER=group_leader
  endif                                                                                                    endif

  ; Place the start and stop time input fields on the panel                                                ; Place the start and stop time input fields on the panel
  TCAD_INPUT_TIMES__DRAW, display_state   ;Top Row is Cycle, Start Time, Stop Time                         TCAD_INPUT_TIMES__DRAW, display_state   ;Top Row is Cycle, Start Time, Stop Time

  base1 = WIDGET_BASE ((*display_state).base, /ROW, /ALIGN_CENTER) ; Base containing Cycle & Tim           base1 = WIDGET_BASE ((*display_state).base, /ROW, /ALIGN_CENTER) ; Base containing Cycl

  base2 = WIDGET_BASE(base1, /COLUMN, /ALIGN_LEFT) ; Everything (Subsystem, Snapshot Options, Di           base2 = WIDGET_BASE(base1, /COLUMN, /ALIGN_LEFT) ; Everything (Subsystem, Snapshot Opti

  baseSubSystem = WIDGET_BASE(base2, /ROW, /ALIGN_LEFT)  ; Subsystem droplist                              baseSubSystem = WIDGET_BASE(base2, /ROW, /ALIGN_LEFT)  ; Subsystem droplist
  tmp = WIDGET_LABEL(baseSubSystem, VALUE = 'Subsystem: ')                                                 tmp = WIDGET_LABEL(baseSubSystem, VALUE = 'Subsystem: ')
  subsystem_id = WIDGET_DROPLIST(baseSubSystem, UVALUE='SUBSYSTEM', VALUE = ['SRE', 'TRE', 'GSS'           subsystem_id = WIDGET_DROPLIST(baseSubSystem, UVALUE='SUBSYSTEM', VALUE = ['SRE', 'TRE'

;WIDGET_CONTROL, subsystem_id, SET_DROPLIST_SELECT = 1                                                   ;WIDGET_CONTROL, subsystem_id, SET_DROPLIST_SELECT = 1
  sub_system = 0   ; default value for subsystem (SRE)                                                     sub_system = 0   ; default value for subsystem (SRE)

tmp = WIDGET_LABEL(baseSubSystem, VALUE = '    ')   ;just a spacer                                       tmp = WIDGET_LABEL(baseSubSystem, VALUE = '    ')   ;just a spacer
  baseSnapID = WIDGET_BASE(baseSubSystem, /ROW, /ALIGN_LEFT, FRAME = 2)                                    baseSnapID = WIDGET_BASE(baseSubSystem, /ROW, /ALIGN_LEFT, FRAME = 2)
  tmp = WIDGET_LABEL(baseSnapID, VALUE = 'Snapshot ID (optional) ')                                        tmp = WIDGET_LABEL(baseSnapID, VALUE = 'Snapshot ID (optional) ')
  snap_id_id = WIDGET_TEXT(baseSnapID, /EDITABLE, UVALUE = 'SNAP_ID', VALUE = '', XSIZE = 5)               snap_id_id = WIDGET_TEXT(baseSnapID, /EDITABLE, UVALUE = 'SNAP_ID', VALUE = '', XSIZE =



  base3 = WIDGET_BASE(base2, /ROW, /ALIGN_BOTTOM)  ; Contains Snapshot Options, Display Options            base3 = WIDGET_BASE(base2, /ROW, /ALIGN_BOTTOM)  ; Contains Snapshot Options, Display O

  base4x = WIDGET_BASE(base3, /COLUMN, XSIZE = 300) ;a placeholder for base4, so it remembers it           base4x = WIDGET_BASE(base3, /COLUMN, XSIZE = 300) ;a placeholder for base4, so it remem
  base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2)   ;Snapshot Options Column                  base4 = WIDGET_BASE(base4x, /COLUMN, /ALIGN_LEFT, FRAME = 2)   ;Snapshot Options Column

  base4a = WIDGET_BASE (base4, /ROW)                                                                       base4a = WIDGET_BASE (base4, /ROW)
  tmp = WIDGET_LABEL(base4a, VALUE = 'SQUID Number')                                                       tmp = WIDGET_LABEL(base4a, VALUE = 'SQUID Number')
  squid_number = 0             ;default value for squid_number                                             squid_number = 0             ;default value for squid_number
  squid_number_id = WIDGET_DROPLIST(base4a, UVALUE = 'SQUIDNUMBER', VALUE = ['0 (All)','1','2','           squid_number_id = WIDGET_DROPLIST(base4a, UVALUE = 'SQUIDNUMBER', VALUE = ['0 (All)','1

  base4b = WIDGET_BASE(base4, /ROW)                                                                        base4b = WIDGET_BASE(base4, /ROW)
  tmp = WIDGET_LABEL(base4b, VALUE = "Snapshot Type")                                                      tmp = WIDGET_LABEL(base4b, VALUE = "Snapshot Type")
  sre_snap_type = 20              ;default value for sre_snap_type                                         sre_snap_type = 20              ;default value for sre_snap_type
  sre_snap_type_id = CW_BGROUP(base4b, ['20 Hz', '1 kHz'], /EXCLUSIVE, UVALUE = 'SRE_SNAPTYPE',            sre_snap_type_id = CW_BGROUP(base4b, ['20 Hz', '1 kHz'], /EXCLUSIVE, UVALUE = 'SRE_SNAP

  base4c = WIDGET_BASE(base4, /ROW)                                                                        base4c = WIDGET_BASE(base4, /ROW)
  tmp = WIDGET_LABEL(base4c, VALUE = "SQUID Mode")                                                         tmp = WIDGET_LABEL(base4c, VALUE = "SQUID Mode")
  squid_mode = 1               ;default value for squid_mode                                               squid_mode = 1               ;default value for squid_mode
  squid_mode_id = CW_BGROUP(base4c, ['Default','Locked'], /EXCLUSIVE, UVALUE = 'SQUIDMODE', SET_           squid_mode_id = CW_BGROUP(base4c, ['Default','Locked'], /EXCLUSIVE, UVALUE = 'SQUIDMODE

  base5 = WIDGET_BASE (base3, /COLUMN, /ALIGN_RIGHT, /ALIGN_TOP) ;base5 is all of the Display Op           base5 = WIDGET_BASE (base3, /COLUMN, /ALIGN_RIGHT, /ALIGN_TOP) ;base5 is all of the Dis

                                                                                                     >     base5a = WIDGET_BASE (base5, /ROW, FRAME=2) ;base5a is all of the Display Options and P

                                                                                                     |     base5b = WIDGET_BASE (base5a, /COLUMN, /ALIGN_TOP) ;This is the actual Display Options
                                                                                                     |     display_options = [1,0,0]
                                                                                                     |     tmp = CW_BGROUP (base5b,["Display to Screen","Send all to Printer", $
                                                                                                     <
                                                                                                     <
                                                                                                     <
                                                                                                     <
  ; ORGANIZED FROM TOP DOWN 5, 5a, 5b, 5c, ...                                                       <
  base5a = WIDGET_BASE (base5, /COLUMN, FRAME=2) ;base5a is border around all of RHS except OK/C     <

  base5b = WIDGET_BASE (base5a, /ROW, /ALIGN_TOP) ;contains 5c and 5d                                <

  base5c = WIDGET_BASE (base5b, /COLUMN, /ALIGN_LEFT) ;contains Display/Write-to-File Buttons        <
  display_options = [1,0];                                                                           <
  tmp = CW_BGROUP (base5c,["Display to Screen", $                                                    <
                          "Write Data to File"], COLUMN=1, /RETURN_INDEX, /NONEXCLUSIVE, $                                   "Write Data to File"], COLUMN=1, /RETURN_INDEX, /NONEXCLUSIVE,
                           UVALUE="DISPLAY_OPTIONS", SET_VALUE=display_options)                                               UVALUE="DISPLAY_OPTIONS", SET_VALUE=display_options)
                                                                                                     >     tmp = WIDGET_LABEL(base5b, VALUE = "Show Snapshots:")

  base5d = WIDGET_BASE (base5b, /COLUMN) ;contains blank space and filename box                      |     base5c = WIDGET_BASE(base5a, /COLUMN, /ALIGN_BOTTOM) ;Printer/Filename column
  tmp = WIDGET_LABEL(base5d, VALUE = "  ", YSIZE=30)  ; Blank space for alignment on screen          |     tmp = WIDGET_LABEL(base5c, VALUE = "  ", YSIZE=25)  ; For alignment on screen
  filename_text_id = WIDGET_TEXT(base5d, /EDITABLE, UVALUE="FILENAME", VALUE = '')                   |     tmp = WIDGET_DROPLIST(base5c, UVALUE='PRINTER', XSIZE=30, VALUE=printer_list)
                                                                                                     <
  base5e = WIDGET_BASE(base5a, /ROW)  ; contains printer select row                                  <
  tmp = WIDGET_LABEL(base5e, VALUE = "Select Active Printer: ")                                      <
  tmp = WIDGET_DROPLIST(base5e, UVALUE='PRINTER', XSIZE=30, VALUE=printer_list)                      <
  (*display_state).printer_droplist = tmp                                                                  (*display_state).printer_droplist = tmp
  WIDGET_CONTROL, (*display_state).printer_droplist, SET_DROPLIST_SELECT=printer  ; (NB: changed           WIDGET_CONTROL, (*display_state).printer_droplist, SET_DROPLIST_SELECT=printer  ; (NB:
                                                                                                     >     filename_text_id = WIDGET_TEXT(base5c, /EDITABLE, UVALUE="FILENAME", VALUE = '')
```

cleaned up widgets
MA 3/12

N#4

```
  base5f = WIDGET_BASE(base5a, /ROW)  ; contains text message to user                              |     base5d = WIDGET_BASE(base5c, /ROW, /ALIGN_LEFT)  ;Which snapshots to display
  tmp = WIDGET_LABEL(base5f, VALUE = "Note: For print options, right-click on plot window")        |     first_snapshot_id = CW_FIELD(base5d, /INTEGER, UVALUE = 'FIRSTSNAPSHOT', VALUE = '', XS
                                                                                                   >     ;first_snapshot_id = WIDGET_TEXT(base5d, /EDITABLE, UVALUE = 'FIRSTSNAPSHOT', VALUE = '
                                                                                                   >     tmp = WIDGET_LABEL(base5d, VALUE = " to ")
                                                                                                   >     last_snapshot_id = CW_FIELD(base5d, /INTEGER, UVALUE = 'LASTSNAPSHOT', VALUE = '', XSIZ
                                                                                                   >     ;last_snapshot_id  = WIDGET_TEXT(base5d, /EDITABLE, UVALUE = 'LASTSNAPSHOT', VALUE = ''

  base5g = WIDGET_BASE(base5a, /ROW)  ; contains Show Snapshot row of stuff                        |     base5e = WIDGET_BASE (base5, /ROW, /ALIGN_CENTER)   ;OK/Cancel Buttons
  tmp = WIDGET_LABEL(base5g, VALUE = "Show Snapshots:")                                            |     tmp = WIDGET_LABEL(base5e, VALUE="       ")
  first_snapshot_id = CW_FIELD(base5g, /INTEGER, UVALUE = 'FIRSTSNAPSHOT', VALUE = '', XSIZE = 1   |     tmp = WIDGET_BUTTON (base5e, VALUE="    OK    ", UVALUE="OKAY")
  tmp = WIDGET_LABEL(base5g, VALUE = " to ")                                                       |     tmp = WIDGET_BUTTON (base5e, VALUE="   Cancel   ", UVALUE="CANCEL")
  last_snapshot_id = CW_FIELD(base5g, /INTEGER, UVALUE = 'LASTSNAPSHOT', VALUE = '', XSIZE = 10,   |     tmp = WIDGET_LABEL(base5e, VALUE="       ")

  base5h = WIDGET_BASE (base5, /ROW, /ALIGN_CENTER)   ;OK/Cancel Buttons                           <
  tmp = WIDGET_LABEL(base5h, VALUE="       ")                                                      <
  tmp = WIDGET_BUTTON (base5h, VALUE="    OK    ", UVALUE="OKAY")                                  <
  tmp = WIDGET_BUTTON (base5h, VALUE="   Cancel   ", UVALUE="CANCEL")                              <
  tmp = WIDGET_LABEL(base5h, VALUE="       ")                                                      <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                                                                                                   <
                                                                                                   <
;   base5a = WIDGET_BASE (base5, /ROW, FRAME=2) ;base5a is border around all of the Display Opti   <
                                                                                                   <
;   base5b = WIDGET_BASE (base5a, /COLUMN, /ALIGN_TOP) ;This is the actual Display Options colum   <
;   display_options = [1,0];  ,0]                                                                  <
;   tmp = CW_BGROUP (base5b,["Display to Screen", $     ;"Send Data to Printer", $                 <
;                            "Write Data to File"], COLUMN=1, /RETURN_INDEX, /NONEXCLUSIVE, $      <
;                             UVALUE="DISPLAY_OPTIONS", SET_VALUE=display_options)                 <
;   base5b2 = WIDGET_BASE (base5b, /ROW)    ; Which printer                                        <
;   tmp = WIDGET_LABEL(base5b2, VALUE = "Select Active Printer: ")                                 <
;   tmp = WIDGET_DROPLIST(base5b2, UVALUE='PRINTER', XSIZE=30, VALUE=printer_list)                 <
;   (*display_state).printer_droplist = tmp                                                        <
;   WIDGET_CONTROL, (*display_state).printer_droplist, SET_DROPLIST_SELECT=printer  ; (NB: chang   <
;   tmp = WIDGET_LABEL(base5b, VALUE = "Show Snapshots:")                                          <
                                                                                                   <
;   base5c = WIDGET_BASE(base5a, /COLUMN, /ALIGN_BOTTOM) ;Printer/Filename column                  <
;   tmp = WIDGET_LABEL(base5c, VALUE = "  ", YSIZE=25)  ; For alignment on screen                  <
;   filename_text_id = WIDGET_TEXT(base5c, /EDITABLE, UVALUE="FILENAME", VALUE = '')               <
                                                                                                   <
;   base5d = WIDGET_BASE(base5c, /ROW, /ALIGN_LEFT)  ;Which snapshots to display                   <
;   first_snapshot_id = CW_FIELD(base5d, /INTEGER, UVALUE = 'FIRSTSNAPSHOT', VALUE = '', XSIZE =   <
;   ;first_snapshot_id = WIDGET_TEXT(base5d, /EDITABLE, UVALUE = 'FIRSTSNAPSHOT', VALUE = '', XS   <
;   tmp = WIDGET_LABEL(base5d, VALUE = " to ")                                                     <
;   last_snapshot_id = CW_FIELD(base5d, /INTEGER, UVALUE = 'LASTSNAPSHOT', VALUE = '', XSIZE = 1   <
;   ;last_snapshot_id  = WIDGET_TEXT(base5d, /EDITABLE, UVALUE = 'LASTSNAPSHOT', VALUE = '', XSI   <
                                                                                                   <
;   base5e = WIDGET_BASE (base5, /ROW, /ALIGN_CENTER)   ;OK/Cancel Buttons                         <
;   tmp = WIDGET_LABEL(base5e, VALUE="       ")                                                    <
;   tmp = WIDGET_BUTTON (base5e, VALUE="    OK    ", UVALUE="OKAY")                                <
;   tmp = WIDGET_BUTTON (base5e, VALUE="   Cancel   ", UVALUE="CANCEL")                            <
;   tmp = WIDGET_LABEL(base5e, VALUE="       ")                                                    <
                                                                                                   <
  ; Other default values                                                                               ; Other default values
  tre_snap_type = 1                                                                                    tre_snap_type = 1
  gss_snap_type = 1                                                                                    gss_snap_type = 1

  ; Display the panel                                                                                  ; Display the panel
  WIDGET_CONTROL, (*display_state).base, SET_UVALUE=display_state                                      WIDGET_CONTROL, (*display_state).base, SET_UVALUE=display_state
  WIDGET_CONTROL, (*display_state).base, /REALIZE                                                      WIDGET_CONTROL, (*display_state).base, /REALIZE

  ;Hand off control of the dialog box to the XMANAGER                                                  ;Hand off control of the dialog box to the XMANAGER
  XMANAGER, 'SNAPSHOT', (*display_state).base                                                          XMANAGER, 'SNAPSHOT', (*display_state).base

END                                                                                                  END
```

← Comments

N74

40

MCR 301

```
; /apps/supported/lasp-2.3/tcad/SCCS/s.L1_fmtrpt.pro 2.5
; @(#)RELEASE VERSION 2.5
; FILENAME              : L1_fmtrpt.pro
; SCCS Delta Date       : 03/17/04,02:45:16
;*********************************************************************
; L1_fmtrpt: Main routine for processing TLM outage via L1 data (SF_MSS_ID = 6943)
; Author:
;   PR McGown   - 19 Dec 2003
; OPTIONS:
;       /DETAIL   - does not do anything yet, if ever
;       /ENGONLY  - 1k/2k versus 32k (default)
; Copyright 2003 by the Regents of Leland Stanford Jr University
;*********************************************************************

PRO L1_fmtrpt, gcTDPPATH, min_sct, max_sct, DETAIL=detail, SCIONLY=scionly, ENGONLY=engonly
  stats = {detail: 0, tlmrate: 35, mnem_totana_tm: 0L, mnem_totana_sn: 0L, $
           mss_min_tm: 0, mss_max_tm: 0, mss_min_sn: 0, mss_max_sn: 0, $
           proctime0: 0.D0, proctime1: 0.D0, proctime2: 0.D0 }
      if KEYWORD_SET (DETAIL)    then stats.detail  =  1
      if KEYWORD_SET (ENGONLY)   then stats.tlmrate =  3
      if KEYWORD_SET (SCIONLY)   then stats.tlmrate = 32

  min_cycle = min_sct.cycle
  max_cycle = max_sct.cycle
  actcycle = STRING(min_cycle)
  while ((STRPOS(actcycle,' ')) ge 0) do begin
     nbeg = STRPOS(actcycle,' ')
     nend = STRLEN(actcycle)
     nbeg = nbeg + 1
     nlen = nend - nbeg
     lcFN = STRMID(actcycle,nbeg,nlen)
     actcycle = lcFN
  endwhile


  ; Record the time at which we're processing begins
  stats.proctime0 = SYSTIME (1)
  UNIX2tstamp, stats.proctime0, Rstamp
  vtcw2tstamp, min_sct.vtcw, min_stamp
  vtcw2tstamp, max_sct.vtcw, max_stamp

  ; Open the file to receive the report
  datapath = GETENV('HOME') + "/FMTRPT/"
  makepath = 'mkdir ' + datapath
     out32 = datapath + "l1_fmtrpt_" + actcycle + "_32k.sum"
     out03 = datapath + "l1_fmtrpt_" + actcycle + "_03k.sum"
  CATCH, no_dir_present
  if no_dir_present ne 0 then spawn, makepath

  OPENW, sum32, out32, WIDTH=999,/GET_LUN
  OPENW, sum03, out03, WIDTH=999,/GET_LUN

  ;JAC = '2.5'
  ;print, '2.5'

  VERSION = "(Version-2.2.2)"

  print, "L1 Data Summary ",VERSION,":"
  print, ""
  print, "  BEGIN = ", Rstamp
  print, ""

  printf, sum32, "L1 Data Summary ",VERSION,":"
  printf, sum32, ""
  printf, sum32, "  TLMRATE                          32k"
  printf, sum32, min_sct.cycle, FORMAT='("  Cycle:                    ",I6)'
  printf, sum32, min_sct.vtcw,  FORMAT='("  MinVtcw:    ",I19)'
  printf, sum32, max_sct.vtcw,  FORMAT='("  MaxVtcw:    ",I19)'
  printf, sum32, min_stamp,      FORMAT='("  MinTime:    ",A19)'
  printf, sum32, max_stamp,      FORMAT='("  MaxTime:    ",A19)'
  printf, sum32, ""

  printf, sum03, "L1 Data Summary ",VERSION,":"
  printf, sum03, ""
  printf, sum03, "  TLMRATE                          3k"
  printf, sum03, min_sct.cycle, FORMAT='("  Cycle:                    ",I6)'
  printf, sum03, min_sct.vtcw,  FORMAT='("  MinVtcw:    ",I19)'
  printf, sum03, max_sct.vtcw,  FORMAT='("  MaxVtcw:    ",I19)'
  printf, sum03, min_stamp,      FORMAT='("  MinTime:    ",A19)'
  printf, sum03, max_stamp,      FORMAT='("  MaxTime:    ",A19)'
  printf, sum03, ""

  if (min_cycle ne max_cycle) then begin
      printf, sum32, min_cycle, max_cycle, FORMAT='("\n  EXITING because Min Cycle (", I6, ") !=
      printf, sum03, min_cycle, max_cycle, FORMAT='("\n  EXITING because Min Cycle (", I6, ") !=
      return
```

+ MCR 301

```
| ; /apps/supported/lasp-2.2.1/tcad/SCCS/s.L1_fmtrpt.pro 2.3
| ; @(#)RELEASE VERSION 2.3
  ; FILENAME              : L1_fmtrpt.pro
| ; SCCS Delta Date        : 02/15/04,21:45:55
  ;*********************************************************************
  ; L1_fmtrpt: Main routine for processing TLM outage via L1 data (SF_MSS_ID = 6943)
  ; Author:
  ;   PR McGown   - 19 Dec 2003
  ; OPTIONS:
  ;       /DETAIL   - does not do anything yet, if ever
  ;       /ENGONLY  - 1k/2k versus 32k (default)
  ; Copyright 2003 by the Regents of Leland Stanford Jr University
  ;*********************************************************************

| PRO L1_fmtrpt, gcTDPPATH, min_sct, max_sct, DETAIL=detail, ENGONLY=engonly
|   stats = {detail: 0, tlmrate: 32, mnem_ana_tot: 0L, mss_min: 0, mss_max: 0, $
<
            proctime0: 0.D0, proctime1: 0.D0, proctime2: 0.D0 }
      if KEYWORD_SET (DETAIL)    then stats.detail  = 1
      if KEYWORD_SET (ENGONLY)   then stats.tlmrate = 3
<

    min_cycle = min_sct.cycle
    max_cycle = max_sct.cycle
    actcycle = STRING(min_cycle)
    while ((STRPOS(actcycle,' ')) ge 0) do begin
       nbeg = STRPOS(actcycle,' ')
       nend = STRLEN(actcycle)
       nbeg = nbeg + 1
       nlen = nend - nbeg
       lcFN = STRMID(actcycle,nbeg,nlen)
       actcycle = lcFN
    endwhile

>
    ; Record the time at which we're processing begins
    stats.proctime0 = SYSTIME (1)
    UNIX2tstamp, stats.proctime0, Rstamp
    vtcw2tstamp, min_sct.vtcw, min_stamp
    vtcw2tstamp, max_sct.vtcw, max_stamp

    ; Open the file to receive the report
    datapath = GETENV('HOME') + "/FMTRPT/"
    makepath = 'mkdir ' + datapath
|     ;outfile = datapath + "l1_fmtrpt_" + min_stamp + "_" + max_stamp + ".sum"
|     outfile = datapath + "l1_fmtrpt_" + actcycle + "_" + STRING(stats.tlmrate, FORMAT="(
    CATCH, no_dir_present
    if no_dir_present ne 0 then spawn, makepath

|   OPENW, outsum, outfile, WIDTH=999,/GET_LUN
<

|   print,"L1 Data Summary (Version-2.2.0):"
<
<
<
<
<

    print, ""
    print, "  BEGIN = ", Rstamp
    print, ""

|   printf, outsum,"L1 Data Summary (Version-2.2.0):"
|   printf, outsum, ""
|   printf, outsum, stats.tlmrate, FORMAT="('  TLMRATE                          ',I2,'k')"
|   printf, outsum, min_sct.cycle, FORMAT='("  Cycle:                    ",I6)'
|   printf, outsum, min_sct.vtcw,  FORMAT='("  MinVtcw:    ",I19)'
|   printf, outsum, max_sct.vtcw,  FORMAT='("  MaxVtcw:    ",I19)'
|   printf, outsum, min_stamp,      FORMAT='("  MinTime:    ",A19)'
|   printf, outsum, max_stamp,      FORMAT='("  MaxTime:    ",A19)'
|   printf, outsum, ""
<
<
<
<
<
<
<
<
<
<

    if (min_cycle ne max_cycle) then begin
|       printf, outsum, min_cycle, max_cycle, FORMAT='("\n  EXITING because Min Cycle (",I6
<
        return
```

Rt 5

41

```
  endif

  ; Connect to the database, IF NOT already connected
  IF !dbi_connected THEN keep_connection = 1 ELSE BEGIN
    DB_CONNECT, "gpbops", "gravitydbops", server=!dbi_server
    keep_connection = 0
  ENDELSE

  L1DATA_SUM, gcTDPPATH, sum32, sum03, min_sct, max_sct, stats

  stats.proctime1 = SYSTIME (1)

  stats.proctime2 = SYSTIME (1)
  UNIX2tstamp, stats.proctime2, Rstamp
      tottime   = (stats.proctime2 - stats.proctime0)/60.

  printf, sum32, ""
  printf, sum32, "  SF_MSS_ID"
  printf, sum32, stats.mnem_totana_tm, FORMAT="('  occurs  ',I12)"
  printf, sum32, stats.mss_min_tm, FORMAT="('  min            x', Z4.4)"
  printf, sum32, stats.mss_max_tm, FORMAT="('  max            x', Z4.4)"

  printf, sum03, ""
  printf, sum03, "  SF_MSS_ID"
  printf, sum03, stats.mnem_totana_sn, FORMAT="('  occurs  ',I12)"
  printf, sum03, stats.mss_min_sn, FORMAT="('  min            x', Z4.4)"
  printf, sum03, stats.mss_max_sn, FORMAT="('  max            x', Z4.4)"

  print, ""
  print, "  END   = ", Rstamp
  print, ""

  print, ""
  print, tottime, FORMAT="('      ',F8.2,'  Total Minutes')"

  CLOSE,  sum32
  CLOSE,  sum03

    ; Disconnect from the database (IF we connected within this routine)
    IF NOT keep_connection THEN DB_DISCONNECT
  return
END

PRO L1DATA_SUM, gcTDPPATH, sum32, sum03, min_sct, max_sct, stats

  ; Define the variables we'll use to extract the info.
  num_tmid=1
  tmid   = 6943
  mnem   = 'SF_MSS_ID'
  mnemtxt = mnem
  FOR i=STRLEN(mnemtxt),15 DO mnemtxt=mnemtxt+' '

  InitSCTDN = 0
  num_tmana = 0L

  if (stats.tlmrate ge 32) then begin
      print, mnemtxt, tmid, FORMAT="('  Querying    ',A16,' = ',I5,'  from GPB_L1A..TManalog')"

      DB_SELECT_TMANALOG, min_sct, max_sct, mnem
      IF !db_status gt 0 THEN BEGIN
         print, "  Data retrieval"
         InitSCTDN = InitSCTDN + 1
         if (InitSCTDN eq 1) then begin
             DB_GET_TMANALOG, tm_sct, tm_dat, 20000   ;Get 1st batch and INITIALIZE arrays
             if (!db_status gt 0) then num_tmana = num_tmana + !db_status
         ENDIF
         WHILE !db_status GT 0 DO BEGIN
             DB_GET_TMANALOG, tm_tsct, tm_tdat, 20000   ;Get EACH set of recs.
             if (!db_status gt 0) then begin
                 num_tmana = num_tmana + !db_status
                 tm_dat = [tm_dat,tm_tdat]
                 tm_sct = [tm_sct,tm_tsct]
             endif
         ENDWHILE
      ENDIF
      if (N_ELEMENTS(tm_dat) ge 1) then begin
          tm_sct = tm_sct(0:N_ELEMENTS(tm_dat)-1)
          tm_dat = tm_dat(0:N_ELEMENTS(tm_dat)-1)
```

mck301

mck 301

```
  endif

  ; Connect to the database, IF NOT already connected
  IF !dbi_connected THEN keep_connection = 1 ELSE BEGIN
    DB_CONNECT, "gpbops", "gravitydbops", server=!dbi_server
    keep_connection = 0
  ENDELSE

  L1DATA_SUM, gcTDPPATH, outsum, min_sct, max_sct, stats

  stats.proctime1 = SYSTIME (1)

  stats.proctime2 = SYSTIME (1)
  UNIX2tstamp, stats.proctime2, Rstamp
      tottime   = (stats.proctime2 - stats.proctime0)/60.
<
<
<
<
<
<
<
<
<
<
<
<
  print, ""
  print, "  END   = ", Rstamp
  print, ""
>
>  printf, outsum, ""
>  printf, outsum, "  SF_MSS_ID"
>  printf, outsum, stats.mnem_ana_tot, FORMAT="('  occurs  ',I12)"
>  printf, outsum, stats.mss_min, FORMAT="('  min            x', Z4.4)"
>  printf, outsum, stats.mss_max, FORMAT="('  max            x', Z4.4)"
  print, ""
  print, tottime, FORMAT="('      ',F8.2,'  Total Minutes')"
>  CLOSE,  outsum
<
<
<
    ; Disconnect from the database (IF we connected within this routine)
    IF NOT keep_connection THEN DB_DISCONNECT
  return
END

|  PRO L1DATA_SUM, gcTDPPATH, outsum, min_sct, max_sct, stats

  ; Define the variables we'll use to extract the info.
  num_tmid=1
  tmid   = 6943
  mnem   = 'SF_MSS_ID'
  mnemtxt = mnem
  FOR i=STRLEN(mnemtxt),15 DO mnemtxt=mnemtxt+' '

  InitSCTDN = 0
<
|  num_tmana = 0L
|  if (stats.tlmrate eq 32) then begin
>      print, mnemtxt, tmid, FORMAT="('  Querying    ',A16,' = ',I5)"
       DB_SELECT_TMANALOG, min_sct, max_sct, mnem
       IF !db_status gt 0 THEN BEGIN
          print, "  Data retrieval"
          InitSCTDN = InitSCTDN + 1
          if (InitSCTDN eq 1) then begin
|             DB_GET_TMANALOG, sct, dn, 20000   ;Get 1st batch and INITIALIZE arrays
              if (!db_status gt 0) then num_tmana = num_tmana + !db_status
          ENDIF
          WHILE !db_status GT 0 DO BEGIN
|             DB_GET_TMANALOG, tsct, tdn, 20000    ;Get EACH set of recs.
              if (!db_status gt 0) then begin
                  num_tmana = num_tmana + !db_status
|                 dn = [dn,tdn]
|                 sct= [sct,tsct]
              endif
          ENDWHILE
       ENDIF
|      if (N_ELEMENTS(dn) ge 1) then begin
|          sct = sct(0:N_ELEMENTS(dn)-1)
|          dn  = dn(0:N_ELEMENTS(dn)-1)
```

SU GP-B INSP 31

RHS

12

```
    endif
endif

InitSCTDN = 0
num_snana = 0L

if ((stats.tlmrate mod 32) eq 3) then begin
    print, mnemtxt, tmid, FORMAT="('  Querying     ',A16,' = ',I5,'  from GPB_L1..SNanalog')"
    DB_SELECT_SNANALOG, min_sct, max_sct, mnem
    IF !db_status gt 0 THEN BEGIN
       InitSCTDN = InitSCTDN + 1
       if (InitSCTDN eq 1) then begin
           DB_GET_SNANALOG, sn_sct, sn_dat, 20000   ;Get 1st batch and INITIALIZE arrays
           if (!db_status gt 0) then num_snana = num_snana + !db_status
       ENDIF
       WHILE !db_status GT 0 DO BEGIN
           DB_GET_SNANALOG, sn_tsct, sn_tdat, 20000   ;Get EACH set of recs.
           if (!db_status gt 0) then begin
               num_snana = num_snana + !db_status
               sn_dat = [sn_dat,sn_tdat]
               sn_sct = [sn_sct,sn_tsct]
           endif
       ENDWHILE
    ENDIF
    if (N_ELEMENTS(sn_dat) ge 1) then begin
        sn_sct = sn_sct(0:N_ELEMENTS(sn_dat)-1)
        sn_dat = sn_dat(0:N_ELEMENTS(sn_dat)-1)
    endif
endif

num_totana = num_tmana + num_snana
num_totalana = N_ELEMENTS(tm_dat) + N_ELEMENTS(sn_dat)

IF (num_totalana eq 0) then begin
    PRINTF, sum32, tmid, mnem, FORMAT='("              ", I5, " ", A16, "   NO DATA FOUND")'
    PRINTF, sum03, tmid, mnem, FORMAT='("              ", I5, " ", A16, "   NO DATA FOUND")'
    return
ENDIF

stats.mnem_totana_tm = LONG(num_tmana) + 0L
stats.mnem_totana_sn = LONG(num_snana) + 0L

     if (stats.tlmrate ge 32) then begin
         ANAHDR = "        BEG_TIME              END_TIME         GAP (secs)"
         ANATLR = "  -------------------  -------------------  ------------"
         PRINTF, sum32, ANAHDR
         PRINTF, sum32, ANATLR
         if (stats.mnem_totana_tm gt 3) then begin
             stats.mss_min_tm = MIN(tm_dat)
             stats.mss_max_tm = MAX(tm_dat)
             BegVtcw = tm_sct(0).vtcw
             BegMSS  = tm_dat(0)
             DataGap = 111L
             FOR Ji=0L,(stats.mnem_totana_tm-2L) DO BEGIN
                 gap = tm_sct(Ji+1L).vtcw - tm_sct(Ji).vtcw
                 if (gap gt DataGap) then begin
                     EndVtcw = tm_sct(Ji).vtcw
                     vtcw2tstamp, BegVtcw, BegTime
                     vtcw2tstamp, EndVtcw, EndTime
                     gap_secs = gap / 10.D0
                     PRINTF, sum32, BegTime, EndTime, gap_secs, FORMAT='("  ", A19, "  ", A19, .
                     BegVtcw = tm_sct(Ji+1).vtcw
                 endif
             ENDFOR
             EndVtcw = tm_sct(stats.mnem_totana_tm-1L).vtcw
             vtcw2tstamp, BegVtcw, BegTime
             vtcw2tstamp, EndVtcw, EndTime
             PRINTF, sum32, BegTime, EndTime, FORMAT='("  ", A19, "  ", A19)'
         endif
     endif

     if ((stats.tlmrate mod 32) eq 3) then begin
         ANAHDR = "        BEG_TIME              END_TIME         DUR (secs)"
         ANATLR = "  -------------------  -------------------  ------------"
         PRINTF, sum03, ANAHDR
         PRINTF, sum03, ANATLR
         if (stats.mnem_totana_sn gt 3) then begin
             stats.mss_min_sn = MIN(sn_dat)
             stats.mss_max_sn = MAX(sn_dat)
             BegVtcw = sn_sct(0).vtcw
             BegMSS  = sn_dat(0)
             DataGap = 560L
             FOR Ji=0L,(stats.mnem_totana_sn-2L) DO BEGIN
                 gap = sn_sct(Ji+1L).vtcw - sn_sct(Ji).vtcw
                 if (gap gt DataGap) then begin
                     EndVtcw = sn_sct(Ji).vtcw
```

```
    endif
endif

num_snana = 0L
if (stats.tlmrate ne 32) then begin
    print, mnemtxt, tmid, FORMAT="('  Querying     ',A16,' = ',I5)"

    DB_SELECT_SNANALOG, min_sct, max_sct, mnem
    IF !db_status gt 0 THEN BEGIN
       InitSCTDN = InitSCTDN + 1
       if (InitSCTDN eq 1) then begin
           DB_GET_SNANALOG, sct, dn, 20000   ;Get 1st batch and INITIALIZE arrays
           if (!db_status gt 0) then num_snana = num_snana + !db_status
       ENDIF
       WHILE !db_status GT 0 DO BEGIN
           DB_GET_SNANALOG, tsct, tdn, 20000    ;Get EACH set of recs.
           if (!db_status gt 0) then begin
               num_snana = num_snana + !db_status
               dn = [dn,tdn]
               sct= [sct,tsct]
           endif
       ENDWHILE
    ENDIF
    if (N_ELEMENTS(dn) ge 1) then begin
        sct = sct(0:N_ELEMENTS(dn)-1)
        dn  = dn(0:N_ELEMENTS(dn)-1)
    endif
endif

num_totana = num_tmana + num_snana
num_totalana = N_ELEMENTS(dn)

IF (num_totalana eq 0) then begin
    PRINTF, outsum, tmid, mnem, FORMAT='("              ",I5," ",A16,"   NO DATA FOUND")

    return
ENDIF

     stats.mnem_ana_tot = num_totalana + 0L
     stats.mss_min = MIN(dn)
     stats.mss_max = MAX(dn)
     BegVtcw = sct(0).vtcw
     BegMSS  = dn(0)
     if (stats.tlmrate eq 32) then begin
         ANAHDR="        BEG_TIME              END_TIME         GAP (secs)"
         ANATLR="  -------------------  -------------------  ------------"
         PRINTF,outsum, ANAHDR
         PRINTF,outsum, ANATLR


         DataGap = 111L
         FOR Ji=0L,(num_totalana-2L) DO BEGIN
             gap = sct(Ji+1L).vtcw - sct(Ji).vtcw
             if (gap gt DataGap) then begin
                 EndVtcw = sct(Ji).vtcw
                 vtcw2tstamp, BegVtcw, BegTime
                 vtcw2tstamp, EndVtcw, EndTime
                 gap_secs = gap / 10.D0
                 PRINTF, outsum, BegTime, EndTime, gap_secs, FORMAT='("  ",A19,"  ",A19
                 BegVtcw = sct(Ji+1).vtcw
             endif
         ENDFOR
         EndVtcw = sct(num_totalana-1).vtcw
         vtcw2tstamp, BegVtcw, BegTime
         vtcw2tstamp, EndVtcw, EndTime
                 PRINTF, outsum, BegTime, EndTime, FORMAT='("  ",A19,"  ",A19)'
     endif
     if (stats.tlmrate ne 32) then begin
         ANAHDR="        BEG_TIME              END_TIME         DUR (secs)"
         ANATLR="  -------------------  -------------------  ------------"
         PRINTF,outsum, ANAHDR
         PRINTF,outsum, ANATLR




         DataGap = 560L
         FOR Ji=0L,(num_totalana-2L) DO BEGIN
             gap = sct(Ji+1L).vtcw - sct(Ji).vtcw
             if (gap gt DataGap) then begin
                 EndVtcw = sct(Ji).vtcw
```

mcr 301

Rt 5

SUGP-B MSP 31

```
                vtcw2tstamp, BegVtcw, BegTime
                vtcw2tstamp, EndVtcw, EndTime
                gap_secs = gap / 10.D0
                dur = EndVtcw - BegVtcw
                dur_secs = dur / 10.D0
                PRINTF, sum03, BegTime, EndTime, dur_secs, FORMAT='("  ", A19, "  ", A19,
                BegVtcw = sn_sct(Ji+1).vtcw
            endif
        ENDFOR
        EndVtcw = sn_sct(stats.mnem_totana_sn-1L).vtcw
        vtcw2tstamp, BegVtcw, BegTime
        vtcw2tstamp, EndVtcw, EndTime
        dur = EndVtcw - BegVtcw
        dur_secs = dur / 10.D0
        PRINTF, sum03, BegTime, EndTime, dur_secs, FORMAT='("  ", A19, "  ", A19, "  ", F
    endif
  endif

 return
END
```

MCL 301

```
<
<
<
<
<
                vtcw2tstamp, BegVtcw, BegTime
                vtcw2tstamp, EndVtcw, EndTime
                gap_secs = gap / 10.D0
                dur = EndVtcw - BegVtcw
                dur_secs = dur / 10.D0
                PRINTF, outsum, BegTime, EndTime, dur_secs, FORMAT='("  ",A19,"  ",A19
                BegVtcw = sct(Ji+1).vtcw
            endif
        ENDFOR
        EndVtcw = sct(num_totalana-1).vtcw
        vtcw2tstamp, BegVtcw, BegTime
        vtcw2tstamp, EndVtcw, EndTime
                dur = EndVtcw - BegVtcw
                dur_secs = dur / 10.D0
                PRINTF, outsum, BegTime, EndTime, dur_secs, FORMAT='("  ",A19,"  ",A19
    endif

 return
END
```


44

Att 6

gpb_db.pro

SU GP-B INSP 31

```
1875a1876
>
1876a1878
>
1878a1881
> ; -- End of file
1881,1903d1883
< pro DB_GET_TLMFMTS, tmid, tlmfmts MCR229
< ; Parameters:
< ; tmid    (in)  - The TMID
< ; tlmfmts (out) - arrays of information
< ;
< ; Other Inputs:
< ;
< ; The global variable !dbi_connected is used to determine whether or not
< ; connections to the GP-B databases already exist
< ;
< ; Other Outputs:
< ;
< ; The global variable !db_status is set to indicate whether or not the
< ; call was successful:
< ; < 0 - An error occurred during the database retrieval
< ; = 0 - No data was returned.  This could be either because there are no
< ;       values for the specified analog in the selected time interval
< ;       or because all qualifying values have already been read by the
< ;       previous calls to this routine
< ; > 0 - Specified number of values returned.  This value will be less
< ;       than or equal to nmax.  If !db_status = nmax, then there may
< ;       be more qualifying values to read -- call this routine again
< ;       to get them
1905d1884
<   tlmfmts = { num_fmts: 0B, fmt_id: BYTARR(256), fmt_rate: BYTARR(256), mfbyte: BYTARR(2
56,101) }
1907,1915d1885
<   tmid   = FIX(tmid)
<   fmt_id = 0B
<   fmt_mf = 0B
<   fmt_rate = 0B
<   mfbyte = BYTARR(100)
<   fmt555 = 255B
<   tlmfmts.fmt_rate(fmt555) = 32B
<   fmt_32 = 0B
<   xfmt   = 0B
1917,1919d1886
<   ON_ERROR, 2
<   if not !dbi_connected then $
<     MESSAGE, "You're not connected to the GPB databases: call DB_CONNECT first"
1921,1926d1887
<   ; SEL the data
<   !db_status = CALL_EXTERNAL (!dbi_path, "db_sel_tlmfmts", tmid)
<   if (!db_status le 0) then begin
<       PRINT, "  ERROR selecting records from GPB_L1..TMdecom by TMID"
<       return
<   endif
1928,1965d1888
<   ; GET the data
<   !db_status = CALL_EXTERNAL (!dbi_path, "db_get_tlmfmts", fmt_id, fmt_mf, mfbyte)
<
<   while (!db_status gt 0) do begin
<     tlmfmts.num_fmts = tlmfmts.num_fmts + 1B
<     tlmfmts.fmt_id(fmt_id) = 0B
<        fmt_rate = BYTE(fmt_mf)
```

Att 6

SU GP-B INSP 31

```
<        if (fmt_rate eq 100) then fmt_rate = 32B
<        if (fmt_rate eq  50) then fmt_rate =  2B
<        if (fmt_rate eq  25) then fmt_rate =  1B
<     tlmfmts.fmt_rate(fmt_id) = BYTE(fmt_rate)
<     if (fmt_rate eq 32) then fmt_32 = fmt_32 + 1B
<     for mf_i=0, (fmt_mf-1) do begin
<         beg_byte = mfbyte(mf_i)
<         xfmt = fmt_id
<         if (beg_byte gt 0) then begin
<             if (fmt_rate eq 32) and (beg_byte le 81) then xfmt = fmt555
<             if (xfmt ne fmt555) or (fmt_32 eq 1) then begin
<                 tlmfmts.fmt_id(xfmt) = tlmfmts.fmt_id(xfmt) + 1B
<                 tlmfmts.mfbyte(xfmt,(mf_i+1)) = beg_byte
<                 if (mf_i eq (fmt_mf-1)) then begin
<                     min_begposn = MIN(tlmfmts.mfbyte(xfmt,1:fmt_mf))
<                     max_begposn = MAX(tlmfmts.mfbyte(xfmt,1:fmt_mf))
<                     if (min_begposn gt 0) and (max_begposn gt 0) and (min_begposn eq max
_begposn) then begin
<                         tlmfmts.mfbyte(xfmt,1:100) = 0B
<                         tlmfmts.mfbyte(xfmt,0) = max_begposn
<                     endif
<                 endif
<             endif
<         endif
<     endfor
<     !db_status = CALL_EXTERNAL (!dbi_path, "db_get_tlmfmts", fmt_id, fmt_mf, mfbyte)
<   endwhile
<   return
< end
<
<
< ; -- End of file
```

all HCR 229

tlmfmts to mnemonics

Date: Fri, 19 Mar 2004 22:35:37 +0000 (GMT)
From: Ron Sharbaugh <ron@science.private.net>
To: ron@relgyro.stanford.edu
cc: Ron Sharbaugh <ron@science.private.net>, Samantha Patterson <spatters@scienc
Subject: lasp-2.3 alpha testing bug fixes
MIME-Version: 1.0

CW_Att 7 #1

This documents the putback and subsequent changes on both science and moc

Ron

#7

```
{local@science:6} putback  tcad/tcad_plot.pro tcad/tcad_values.pro
auto_import/defaults.exp auto_import/initialize.exp cron/science_auto.cron
cron/crontab_entry.txt
Parent workspace: /apps/supported/lasp-ParentWS
Child workspace:  /apps/supported/lasp-dev
Comment?
> alpha testing bug fixes - will go into lasp-2.3
>
<

Examined files: 6

Putting back contents changes: 6

update: tcad/tcad_plot.pro
update: tcad/tcad_values.pro
update: auto_import/defaults.exp
update: auto_import/initialize.exp
update: cron/science_auto.cron
update: cron/crontab_entry.txt

Examined files: 6

Contents Summary:
       6   update
```

2 chgs to tcad dir/
Eliminate conditions that created a null pointer.

```
{local@science:7} pwd
/apps/supported/lasp-dev
{local@science:8} diff tcad/tcad_plot.pro
/apps/supported/lasp-2.3/tcad/tcad_plot.pro
1472,1480d1471
     IF NOT PTR_VALID((*display_state).data_list) THEN BEGIN
        tmp = WIDGET_MESSAGE("Select a mnemonic first.")
        return
     ENDIF
     IF N_ELEMENTS((*(*display_state).data_list)) EQ 0 THEN BEGIN
        tmp = WIDGET_MESSAGE("Select a mnemonic first.")
        return
     ENDIF
```

Addition only

```
{local@science:9} diff tcad/tcad_values.pro
/apps/supported/lasp-2.3/tcad/tcad_values.pro
1,2c1,2
< ; /apps/supported/lasp-dev/tcad/SCCS/s.tcad_values.pro 2.15
< ; @(#)RELEASE VERSION 2.15
```

CW_ Att #2

SU GP-B INSP 31

```
---
> ; /apps/supported/lasp-2.3/tcad/SCCS/s.tcad_values.pro 2.14
> ; @(#)RELEASE VERSION 2.14
4c4
< ; SCCS Delta Date        : 03/19/04,19:40:32
---
> ; SCCS Delta Date        : 03/09/04,02:45:34
97c97
                    IF PTR_VALID((*(*display_state).data_list)[rec].snsct) THEN
BEGIN
--
                    IF NOT PTR_VALID((*(*display_state).data_list)[rec].snsct)
THEN BEGIN
{local@science:10}
{local@science:13} diff auto_import/defaults.exp
/apps/supported/lasp-2.3/auto_import/defaults.exp
2,3c2,3
< # /apps/supported/lasp-dev/auto_import/SCCS/s.defaults.exp 2.6
< # @(#)RELEASE VERSION 2.6
---
> # /apps/supported/lasp-2.3/auto_import/SCCS/s.defaults.exp 2.5
> # @(#)RELEASE VERSION 2.5
5c5
< # SCCS Delta Date        : 03/19/04,19:40:32
---
> # SCCS Delta Date        : 03/17/04,02:46:34
76c76
< log_user 1
---
> log_user 0
87c87
< set VARS(tdpproc) "/apps/supported/scripts/tdp"
---
> set VARS(tdpproc) "$VARS(path)/tdp"
{local@science:14} iff auto_import/initialize.exp
/apps/supported/lasp-2.3/auto_import/initialize.exp
2,3c2,3
< # /apps/supported/lasp-dev/auto_import/SCCS/s.initialize.exp 2.4
< # @(#)RELEASE VERSION 2.4
---
> # /apps/supported/lasp-2.3/auto_import/SCCS/s.initialize.exp 2.3
> # @(#)RELEASE VERSION 2.3
5c5
< # SCCS Delta Date        : 03/18/04,19:40:32
---
> # SCCS Delta Date        : 03/17/04,02:46:34
38d37
<                   puts "$tmp"
{local@science:15} diff cron/science_auto.cron
/apps/supported/lasp-2.3/cron/science_auto.cron
2,3c2,3
< # /apps/supported/lasp-dev/cron/SCCS/s.science_auto.cron 1.13
< # @(#)RELEASE VERSION 1.13
---
> # /apps/supported/lasp-2.3/cron/SCCS/s.science_auto.cron 1.12
> # @(#)RELEASE VERSION 1.12
5c5
< # SCCS Delta Date        : 03/19/04,21:23:03
---
> # SCCS Delta Date        : 03/18/04,02:46:35
```