GLAST LAT Flight Software Demonstration

SLAC Campus
Building 84, Central Lab Annex

January 28, 2005
Overview Presentation 11:00 AM
Demonstrations 11:15 AM

1 Demonstration Overview

1.1 Agenda for the Demonstrations

The demonstration will take place in the Central Laboratory Annex (Building 84), Room B-101.

Demo Agenda Item	Presenter(s)
1. Overview of the Demonstration	Lawrence Jeung
2. File Management Demonstration	Lawrence Jeung
3. Questions from Attendees	NA

Feel free to jot questions and comments down in the margins of this document or in the space provided on page 25.

1.2 Goals of the Demonstration

The January 28, 2005 demonstration covers:

File Management. This demo uses the Spacecraft Data Interface Simulator (SDIS) to represent the Spacecraft and a test crate to represent the SIU onboard the LAT. The demonstration makes use of special archiving features of the SDIS AstroRT software and FSW utilities that convert AstroRT archive data to MOC format and human-readable records.

The demonstration shows the nearly-complete range of application-level file management operations, including directory and file operations, and file uploads. 12 of 14 file management requirements from Section 5.3.7 of the SRS (Version 4) will meet partial progress with this demonstration.

Note that this demonstration covers application-level (i.e., post-boot) file management functionality. Execution of file upload and file upload cancel commands during the boot stage was previously demonstrated.

1.3 Verification of Requirements

The following table cites the software requirements scheduled to be demonstrated today. Where entries in this table refer to other documents, consult the SRS for the list of citations.

FSW Requirements Demonstrated	Description	Completion Status
5.3.7.1: Directory Create	In order to create a directory, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes directory identifier, device identifier, and unit identifier.	Partial

FSW Requirements Demonstrated	Description	Completion Status
5.3.7.2: Directory Delete	In order to delete a directory, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes directory identifier, device identifier, and unit identifier.	Partial
5.3.7.3: File Delete	In order to delete a file, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes file identifier, directory identifier, device identifier, and unit identifier.	Partial
5.3.7.4: File Copy within a Processor	In order to copy a file within a processor, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes source file identifier, source directory identifier, source device identifier, unit identifier, destination file identifier, destination directory identifier, and destination device identifier.	Partial
5.3.7.5: File Copy from SIU to EPU	In order to copy a file from the SIU to an EPU processor, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes source file identifier, source directory identifier, source device identifier, and destination unit identifier. Note that a separate commit command is sent to the EPU to move the file from EPU RAM into the file system.	Rescheduled to February 4, 2005.
5.3.7.6.1.1: File Dump Command	In order to dump a file, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes file identifier, directory identifier, device identifier, and unit identifier.	Partial
5.3.7.6.1.2: File Dump Data	In response to receiving a File Dump command, the FSW shall transmit the requested file data to the spacecraft.	Partial
5.3.7.6.2: File Dump Cancel	The FSW shall receive as input, from the spacecraft via the CTDB, a command to cancel the dump of a file based on unit identifier.	This requirement is pending removal from next version of the SRS
5.3.7.7.1.1: File Directory Dump Command	In order to dump a file directory, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes directory identifier, device identifier, and unit identifier.	Partial

FSW Requirements Demonstrated	Description	Completion Status
5.3.7.7.1.2: File Directory Dump Data	In response to receiving a File Directory Dump command, the FSW shall transmit the requested data to the spacecraft, including: device identifier, directory identifier, file identifier, setting of read-only flag, archive flag setting, most recent update time, byte size, and number of blocks for each file and subdirectory in the requested directory.	Partial
5.3.7.8.1.1: File System Status Dump Command	The FSW shall receive as input, from the spacecraft via the CTDB, a command to dump the file system status for a specific unit.	Partial
5.3.7.8.1.2: File System Status Dump Data	In response to receiving a File System Status Dump command, the FSW shall transmit the requested data to the spacecraft, including: total block size, blocks used, and blocks free for the requested unit.	Partial
5.3.7.9: File Loads	After entering the boot shell, the FSW processors shall be commandable by the SIU to perform file loads. Further design details are provided in [12].	Partial
5.3.7.10: Upload Cancellation	The FSW shall be commandable to cancel an active file load.	Partial

2 Demonstration Procedure

2.1 Context of the Demonstration

2.1.1 Hardware Context for the File Management Demonstration

The File Management demonstration is carried out using the Spacecraft Data Interface Simulator (SDIS) from Spectrum Astro and a test crate acting as the SIU.

2.1.1.1 The Spacecraft Data Interface Simulator (SDIS)

The SDIS represents significant portions of the interface between the LAT and the Spacecraft. The SDIS comprises one side of the SC C&DH system with a 1553 bus, LVDS Science Interface, and Discrete monitors and controls.

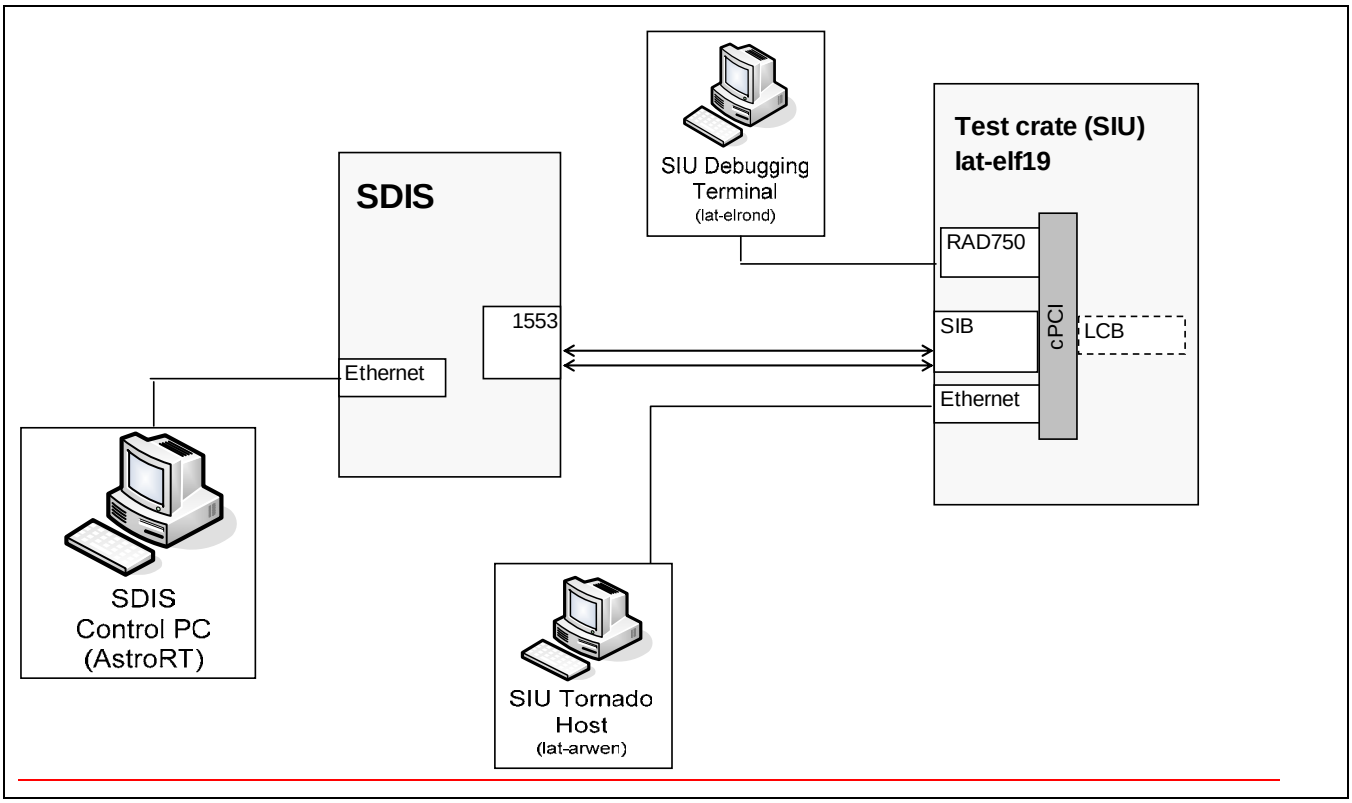
A PC, connected to the SDIS via Ethernet, is used to control the Simulator. The PC runs Spectrum Astro's proprietary Spacecraft command and control package, AstroRT.

2.1.1.2 Test Crate (the "SIU")

The SIU crate contains a RAD750 processor, a 1553-capable cPCI Spacecraft Interface Board (SIB), a cPCI LAT Communications Board, a serial port, and a Ramix Ethernet board.

Today's demo makes use of the 1553 connection between the SIU crate and the SDIS.

Figure 1: Hardware Setup for December 3, 2004 FSW Demonstration



2.1.2 Software Context for the Demonstrations

2.1.2.1 AstroRT

All telecommands are issued from the SDIS Control PC using AstroRT software provided by Spectrum Astro. In past demonstrations, commands and telemetry have been visualized in AstroRT GUI windows. The AstroRT application also exposes an API that can be accessed by Perl code. For today's demonstration, Perl scripts will be used to issue the file management telecommands.

AstroRT also allows users to archive all command and telemetry transactions in .dat files. In today's demonstration, a long series of commands will be issued and the results of these commands will be stored in an AstroRT archive file. Two FSW tools, **siis2moc** and **lfs_moc_dump**, are used to convert archived commands and telemetry to MOC format and display the packet contents, respectively.

2.1.2.2 Flight Software on the Test Crate SIU

The following software libraries are loaded on the SIU during the course of the file management demonstration. Unless otherwise indicated, these packages are production versions of FSW as of January 27, 2005:

/RAD750/prod/librad750_reboot.o, /CMX/prod/libcmx_asBuiltSpy.o, /PBS/prod/libpbs.o, /MSG/prod/libmsg_mt.o, /MSG/prod/libmsg_print.o, /ZLIB/prod/libzlib_inflate.o, /CCSDS/prod/libccsds_pkt.o, /CCSDS/prod/libccsds_dump.o,

/CCSDS/prod/libccsds_swap.o, /SIB/prod/libsib.o, /CTDB/prod/libsumt_rt_sib.o, /ITC/prod/libitc_dump.o, /ITC/prod/libitc.o, /CTS/prod/libctx_lcp_sumt.o, /CTS/prod/libcts_lcp.o, /PBC/prod/libpbc_lcp.o, /LSM/prod/liblsm_dump.o, /LSM/prod/liblsm.o, /MEM/prod/libmem_base.o, /MEM/prod/libmem.o, /MEM/prod/libmem_unit_test.o, /FILE/prod/libfile_hdr.o, /FILE/prod/libfile_path.o, /FILE/prod/libfile_sys.o, /FILE/prod/libfile_upl.o, /FILE/prod/libfile_lcp.o, /FILE/prod/libfile_dump.o, /LCM/prod/liblcm.o, /LFS/prod/liblfs_lcp.o, /LFS/prod/liblfs_dump.o, /LCBD/prod/liblcbd.o, /LEM/prod/liblem.o, /LEM/prod/liblem_lists.o, /LEM/prod/ liblem_cli.o, /PIG/prod/libpig_power.o, /PIG/prod/libpig_flying.o, /PIG/prod/libpig_rooting.o, /LHK/prod/liblhk_cfg.o, /LHK/prod/liblhk_slv.o, /LHK/liblhk_sim.o, /LCP/prod/liblcp.o"

2.2 File Management Demonstration

The file management demonstration proceeds in the following 10 major steps:

- Step 1: Demo Setup: Creating a File to Upload, Enabling the AstroRT Archive Function
- Step 2: Creating Directories and Dumping Directory Contents
- Step 3: Uploading a File
- Step 4: Canceling a File Upload
- Step 5: Copying a File on the SIU
- Step 6: Dumping File Contents
- Step 7: Deleting a File
- Step 8: Requesting File System Status Data
- Step 9: Deleting Directories
- Step 10: Displaying the Contents of the AstroRT Archive File

Before the demonstration begins, the demonstrator has opened terminal windows to control VxWorks sessions, powered up the SDIS, and launched the AstroRT software on the SDIS Control PC.

2.2.1 Step 1: Demo Setup: Creating a File to Upload and Enabling the AstroRT Archive

Step 1 of the file management demo proceeds as follows:

1. The demonstrator powers up the test “SIU” crate.
    a. The SIU enters primary boot. After a few minutes, primary boot is complete and the boot shell is available for commands.
2. On lat-elrond, the SIU debugging terminal, the demonstrator opens a Unix terminal window and launches the FSW code management tool with the **cmx start** command.
3. To create a text file named “echo.txt” to manipulate during the demonstration, the demonstrator dumps the contents of the Unix man page for the Unix echo command by entering **man echo > echo.txt** at the Unix prompt. The contents are shown in Figure 2 on page 17.
4. The demonstrator adds a FSW file header to this text file with a tool from the LFS FSW package: **file_hdr_prefix –k 300 –t 7 –n ECHO echo.txt echo.f**.
5. To check the header in the newly-created “echo.f”, the demonstrator runs **file_hdr_show echo.f**.
6. The demonstrator copies echo.f to a floppy disk or CD, and uses the disk to transfer the file to the SDIS Control PC.
7. At the SDIS Control PC, the demonstrator begins the archiving process in AstroRT, resulting in the creation of an archive file with a name of type “Archive_*.dat”.

8. The demonstrator uses the AstroRT GUI to send the LPBCRTOSEXEC command to the SIU to kick it into secondary boot.
9. At lat-arwen, the SIU Tornado host, the demonstrator loads FSW libraries to the SIU using the **lcp_startup_rad750.vx** script. See Section 2.1.2.2 on page 6 for the list of libraries.

2.2.2 Step 2: Creating Directories and Dumping Directory Contents

In Step 2 of the demo, the demonstrator creates and dumps the contents of directories on the two separate file system devices, ee0 and ee1, present in SIU EEPROM. Step 2 proceeds as follows:

1. At an MS-DOS prompt on the SDIS Control PC, the demonstrator dumps the contents of the root directory, d127, on EPPROM device ee0 by executing a Perl script with the following command: **perl llfsdirdump 0 2 127 0**. This is Transaction 0.
2. The demonstrator then creates a directory, d111, on EEPROM device ee0 with the following command: **perl llfsdircreat.pl 0 2 111 1**. This is Transaction 1.
3. The demonstrator re-dumps the contents of d127 with **perl llfsdirdump.pl 0 2 127 2**. This is Transaction 2.
4. The demonstrator dumps the contents of d111 with **perl llfsdirdump.pl 0 2 111 3**. This is Transaction 3.
5. The demonstrator now dumps the contents of a directory on EPPROM device ee1. Specifically, the contents of /ee1/d127 (the root directory on ee1) are dumped with **perl llfsdirdump.pl 0 3 127 4**. This is Transaction 4.
6. The demonstrator creates a directory, d112 on device ee1, using **perl llfsdircreat.pl 0 3 112 5**. This is Transaction 5.
7. During Transaction 6, the demonstrator dumps directory d127 on device ee1 with **perl llfsdirdump.pl 0 3 127 6**.
8. During Transaction 7, the demonstrator dumps directory d112 on device ee1 with **perl llfsdirdump.pl 0 3 112 7**.

Results of these activities are logged to the AstroRT archive file. Records from this file are displayed for confirmation later in the demonstration.

This part of the demonstration shows that FSW provides functionality to satisfy the following requirements from Version 4 of the LAT Flight Software Specification – Level III:

FSW Requirements Demonstrated	Description	Monitor Verification
5.3.7.1: Directory Create	In order to create a directory, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes directory identifier, device identifier, and unit identifier.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.
5.3.7.7.1.1: File Directory Dump Command	In order to dump a file directory, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes directory identifier, device identifier, and unit identifier.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.
5.3.7.7.1.2: File Directory Dump Data	In response to receiving a File Directory Dump command, the FSW shall transmit the requested data to the spacecraft, including: device identifier, directory identifier, file identifier, setting of read-only flag, archive flag setting, most recent update time, byte size, and number of blocks for each file and subdirectory in the requested directory.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.

2.2.3 Step 3: Uploading a File

In Step 3, the text file with header, echo.f, that was created earlier is uploaded to the SIU. Step 3 of the file management demo proceeds as follows:

1. The demonstrator uploads the file echo.f to File f0000001 in directory d111 on device /ee0 with **perl lfileupload.pl "echo.f" 2 111 0000001**.
2. During Transaction 9, the demonstrator dumps directory d111 on device /ee0 using **perl llfsdirdump.pl 0 2 111 9**.

Results of these activities are logged to the AstroRT archive file. Records from this file are displayed for confirmation later in the demonstration.

This part of the demonstration shows that FSW provides functionality to satisfy the following requirements from Version 4 of the LAT Flight Software Specification – Level III:

FSW Requirements Demonstrated	Description	Monitor Verification
5.3.7.9: File Loads	After entering the boot shell, the FSW processors shall be commandable by the SIU to perform file loads. Further design details are provided in [12].	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.

2.2.4 Step 4: Canceling a File Upload

In this Step, the demonstrator repeats the file upload from Step 3. This time, however, the demonstrator sends the upload file to EEPROM device ee1 and uses the LFILUPLCANCEL telecommand to interrupt the file upload process. Step 4 proceeds as follows

1. The demonstrator uploads file “echo.f” to file f0000001 in directory d112 on device /ee1 using **perl lfileupload.pl “echo.f” 3 112 0000001**.
2. The demonstrator cancels the upload with **perl lfilecancel.pl 0** and resets the file upload system by issuing **perl lfilecancel.pl 0** again.
3. During Transaction 12, to confirm the file was NOT committed to directory d112 and thus that the cancel was successful, the demonstrator dumps directory d112 on device /ee1 with **perl llfsdirdump.pl 0 3 112 12**.

Results of these activities are logged to the AstroRT archive file. Records from this file are displayed for confirmation later in the demonstration.

This part of the demonstration shows that FSW provides functionality to satisfy the following requirements from Version 4 of the LAT Flight Software Specification – Level III:

FSW Requirements Demonstrated	Description	Monitor Verification
5.3.7.10: Upload Cancellation	The FSW shall be commandable to cancel an active file load.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.

2.2.5 Step 5: Copying a File on the SIU

In Step 5, the demonstrator copies echo.f from device ee0 to device ee1 on the SIU. Step 5 proceeds as follows:

1. During Transaction 13, the demonstrator copies file f0000001 in Directory d111 on device ee0 to f0000001 in Directory d112 on device ee1 using **perl llfsfilecopy.pl 0 2 111 0000001 3 112 0000001 13**.
2. During Transaction 14, to confirm the file was copied, the demonstrator dumps directory d112 on device ee1 using **perl llfsdirdump.pl 0 3 112 14**.

Results of these activities are logged to the AstroRT archive file. Records from this file are displayed for confirmation later in the demonstration.

This part of the demonstration shows that FSW provides functionality to satisfy the following requirements from Version 4 of the LAT Flight Software Specification – Level III:

FSW Requirements Demonstrated	Description	Monitor Verification
5.3.7.4: File Copy within a Processor	In order to copy a file within a processor, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes source file identifier, source directory identifier, source device identifier, unit identifier, destination file identifier, destination directory identifier, and destination device identifier.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.

2.2.6 Step 6: Dumping File Contents

At this stage of the demonstration, the test file, echo.f, is stored on both EEPROM devices, ee0 and ee1. In Step 6, the demonstrator dumps the contents of both copies of echo.f to telemetry. Step 6 proceeds as follows

1. During Transaction 15, the demonstrator dumps file f0000001 in directory d111 on device ee0 using **perl llfsfiledump_ctdb.pl 0 2 111 0000001 15**.
2. During Transaction 16, the demonstrator dumps file f0000001 in directory d112 on device ee1 with **perl llfsfiledump_ctdb.pl 0 3 112 0000001 16**.

Results of these activities are logged to the AstroRT archive file. Records from this file are displayed for confirmation later in the demonstration.

This part of the demonstration shows that FSW provides functionality to satisfy the following requirements from Version 4 of the LAT Flight Software Specification – Level III:

FSW Requirements Demonstrated	Description	Monitor Verification
5.3.7.6.1.1: File Dump Command	In order to dump a file, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes file identifier, directory identifier, device identifier, and unit identifier.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.
5.3.7.6.1.2: File Dump Data	In response to receiving a File Dump command, the FSW shall transmit the requested file data to the spacecraft.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.

2.2.7 Step 7: Deleting a File

The demonstrator proceeds to delete both copies of the echo.f test file from EEPROM. Step 7 proceeds as follows

1. During Transaction 17, the demonstrator deletes file f0000001 in directory d111 on device ee0 using **perl llfsfiledel.pl 0 2 111 0000001 17**.
2. During Transaction 18, to confirm the file deletion, the demonstrator dumps directory d111 on device ee0 using **perl llfsdirdump.pl 0 2 111 18**.
3. During Transaction 19, the demonstrator deletes file f0000001 in directory d112 on device ee1 using **perl llfsfiledel.pl 0 3 112 0000001 19**.
4. During Transaction 20, to confirm this second deletion, the demonstrator dumps directory d112 on device ee1 using **perl llfsdirdump.pl 0 3 112 20**.

Results of these activities are logged to the AstroRT archive file. Records from this file are displayed for confirmation later in the demonstration.

This part of the demonstration shows that FSW provides functionality to satisfy the following requirements from Version 4 of the LAT Flight Software Specification – Level III:

FSW Requirements Demonstrated	Description	Monitor Verification
5.3.7.3: File Delete	In order to delete a file, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes file identifier, directory identifier, device identifier, and unit identifier.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.

2.2.8 Step 8: Requesting File System Status Data

In this Step, the demonstrator sends a command requesting a file system status report for each EEPROM device. Step 8 proceeds as follows

1. During Transaction 21, the demonstrator dumps the file system status of device ee0 using **perl llfssysstat.pl 0 2 21**.
2. During Transaction 22, the demonstrator dumps the file system status of device ee1 using **perl llfssysstat.pl 0 3 22**.

Results of these activities are logged to the AstroRT archive file. Records from this file are displayed for confirmation later in the demonstration.

This part of the demonstration shows that FSW provides functionality to satisfy the following requirements from Version 4 of the LAT Flight Software Specification – Level III:

FSW Requirements Demonstrated	Description	Monitor Verification
5.3.7.8.1.1: File System Status Dump Command	The FSW shall receive as input, from the spacecraft via the CTDB, a command to dump the file system status for a specific unit.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.
5.3.7.8.1.2: File System Status Dump Data	In response to receiving a File System Status Dump command, the FSW shall transmit the requested data to the spacecraft, including: total block size, blocks used, and blocks free for the requested unit.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.

2.2.9 Step 9: Deleting Directories

In the final step involving telecommands to the SIU, the demonstrator deletes directory d111 on device ee0 and d112 on device ee1. Step 9 proceeds as follows

1. During Transaction 23, the demonstrator deletes directory d111 on device ee0 with **perl llfsdirdel.pl 0 2 111 23**.
2. During Transaction 24, to confirm the deletion, the demonstrator dumps directory d127 (the root directory) on device ee0 using **perl llfsdirdump.pl 0 2 127 24**.
3. During Transaction 25, the demonstrator deletes directory d112 on device ee1 with **perl llfsdirdel.pl 0 3 112 25**.
4. During Transaction 26, to confirm the deletion, the demonstrator dumps directory d127 (root) on device ee1 with **perl llfsdirdump.pl 0 3 127 26**.

Results of these activities are logged to the AstroRT archive file. Records from this file are displayed for confirmation later in the demonstration.

This part of the demonstration shows that FSW provides functionality to satisfy the following requirements from Version 4 of the LAT Flight Software Specification – Level III:

FSW Requirements Demonstrated	Description	Monitor Verification
5.3.7.2: Directory Delete	In order to delete a directory, the FSW shall receive as input, from the spacecraft via the CTDB, a command that includes directory identifier, device identifier, and unit identifier.	This requirement was successfully demonstrated _____ Monitor Initials If only partially successful, record deviations on page 25.

2.2.10 Step 10: Displaying the Contents of the AstroRT Archive File

At this stage, all commands and telemetry have been exchanged between the SDIS and the test SIU crate. The demonstrator halts logging to the AstroRT archive, uses the **siis2moc** tool to convert the archive file to MOC format, and displays the contents of the archive file, by TransactionID, to a terminal window for verification. Step 10 proceeds as follows

1. The demonstrator uses the AstroRT GUI to halt the archiving process.
2. The demonstrator copies the AstroRT archive (Archive_*.dat) file to a floppy disk or CD, and uses the disk to transfer the file to a PC on the network.
3. On the networked PC, the demonstrator issues the **siis2moc Archive_*.dat** command to convert the AstroRT archive file to a set of telecommand and telemetry files in MOC format.
4. The demonstrator then uses a special-purpose tool from the LFS Flight Software package to display the contents of individual CCSDS packets in the archive file. A series

of commands of the form **lfs_moc_dump –v –x<xid> <MOC format file> > xid<xid>** are issued, where **<xid>** is a particular TransactionID, <**MOC format file**> is one of the files, and **xid<xid>** is a text file containing the piped output of the lfs_moc_dump command.

a. For instance, to dump telemetry from MOC file PKT_*_*_*_00793.0 with TransactionID 0 to a text file named “xid0”, the demonstrator issues the command **lfs_moc_dump –v –x0 PKT_*_*_*_00793.0 > xid0**

b. The <xid> parameters correspond to the TransactionIDs used in Steps 1 through 9 above.

5. To confirm the file management operations were successfully completed, the contents of the text files (e.g., “xid0”) are inspected using a simple Unix file viewer. The contents of these files are shown in Figure 2 through Figure 11 below.

3 Outputs of the Demonstration

This section contains the screen captures, terminal outputs, and other raw results of the demonstrations run today. These figures are provided to assist monitors in verifying that the FSW successfully demonstrated compliance with the software requirements cited above.

3.1 Outputs of the File Management Demonstration

In Steps 1 through 10 of the file management demonstration, a test file and an AstroRT archive are created to capture the results of the telecommands and telemetry used to manipulate files and directories on the test crate SIU. These outputs are presented below in order of the Steps followed during the demo.

3.1.1 Outputs of File Management Demo Step 1

The demonstrator dumps the Unix man page entry for the Unix “echo” command to a text file named echo.txt. See Figure 2 below. This file is uploaded and manipulated during the demo.

Figure 2: Contents of Test File, “echo.txt”

```
User Commands                                                   echo(1)



NAME
     echo - echo arguments

SYNOPSIS
     /usr/bin/echo [string...]

DESCRIPTION
     The echo utility writes its arguments, separated  by  BLANKs
     and  terminated  by  a  NEWLINE,  to the standard output. If
     there are no arguments, only the NEWLINE character  will  be
     written.

     echo is useful for producing diagnostics in  command  files,
     for  sending  known data into a pipe, and for displaying the
     contents of environment variables.

     The C shell, the Korn shell, and the Bourne shell  all  have
     echo  built-in  commands, which, by default, will be invoked
     if  the  user  calls  echo  without  a  full  pathname.  See
     shell_builtins(1).  sh's echo, ksh's echo, and /usr/bin/echo
     understand the back-slashed escape characters,  except  that
     sh's  echo does not understand \a as the alert character. In
     addition, ksh's echo, does not have an -n option. sh's  echo
     and  /usr/bin/echo  only  have  an  -n  option  if the SYSV3
     environment  variable  is  set  (see  ENVIRONMENT  VARIABLES
     below).  If  it  is, none of the backslashed characters men-
     tioned above are available. csh's echo and /usr/ucb/echo, on
     the other hand, have an -n option, but do not understand the
     back-slashed escape characters.
```

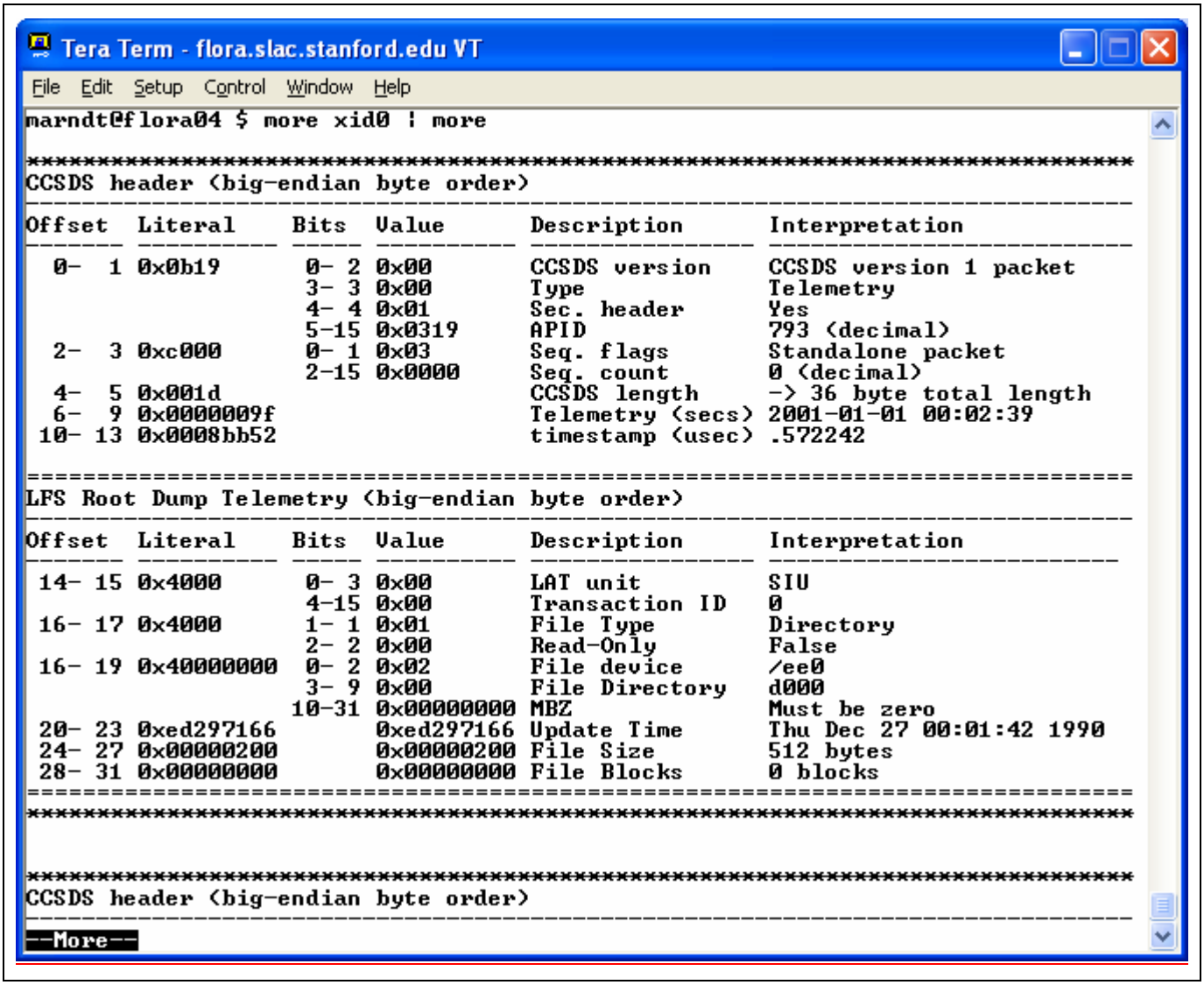
```
      OPERANDS
           The following operand is supported:
      ...
```

3.1.2 Outputs of File Management Demo Step 2

In Step 2, the demonstrator creates a directory, d111, on device ee0 and a directory, d112, on device ee1, and dumps the contents of the root directories (d127) and the contents of the newly created directories to confirm the new directory structure. These operations result in 8 transactions.

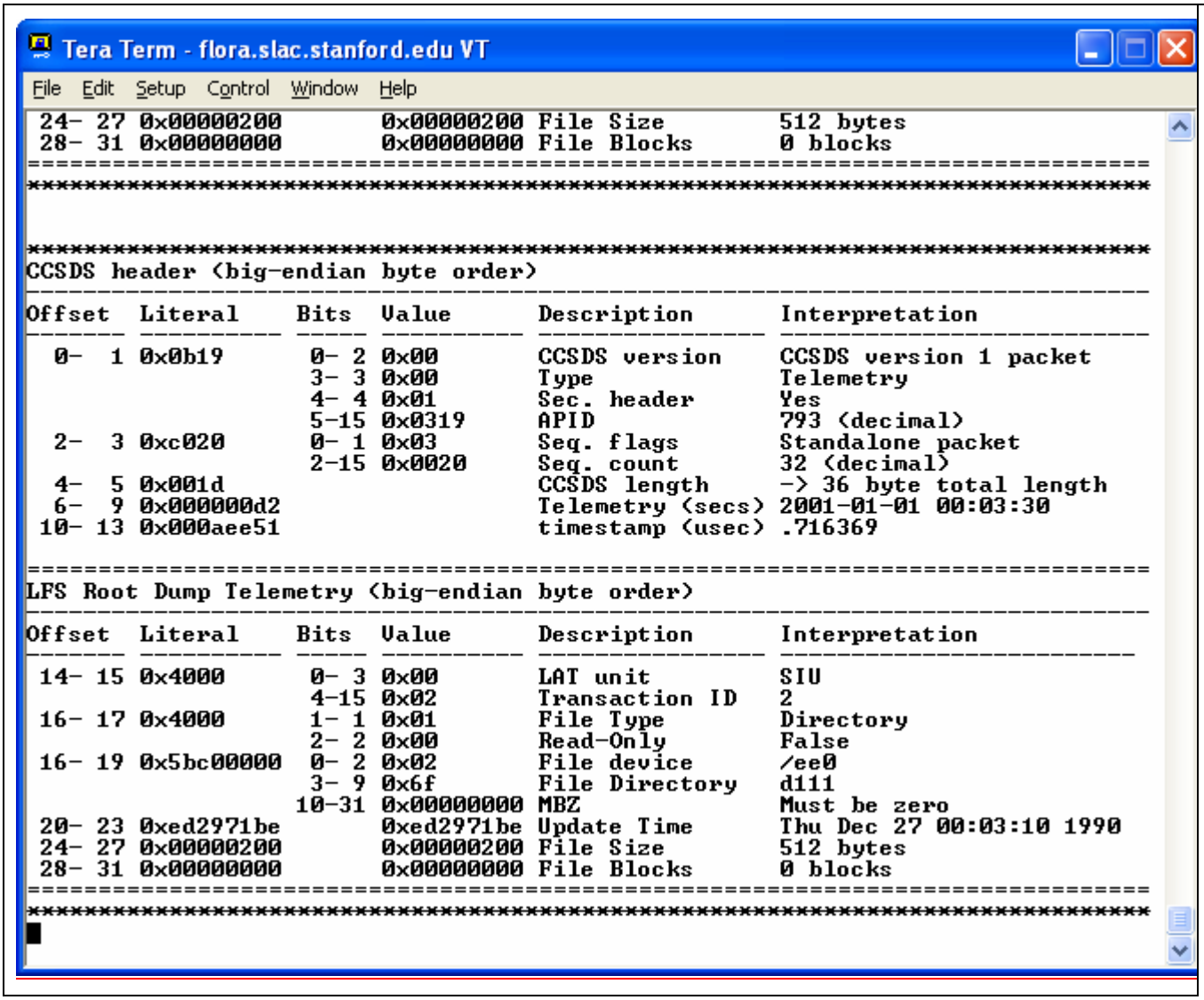
Two of these transactions are shown below. Figure 3 shows the contents of one of the LLFSROOTLIST packets returned by the LLFSDIRDUMP telecommand. By reviewing the *File device* and *File Directory* fields from the whole set of these packets, the demonstrator confirms that no directory d111 appears under the root directory, d127.

Figure 3: Confirming Directory d111 Does Not Exist on Device ee0 (TransactionID 0)



The next figure shows the contents of one of the LLFSROOTLIST packets returned by the LLFSDIRDUMP telecommand. By reviewing the *File device* and *File Directory* fields from the whole set of these packets, the demonstrator confirms that directory d111 now DOES appear under the root directory, d127.

Figure 4: Confirming Directory d111 was Created on Device ee0 (TransactionID 2)



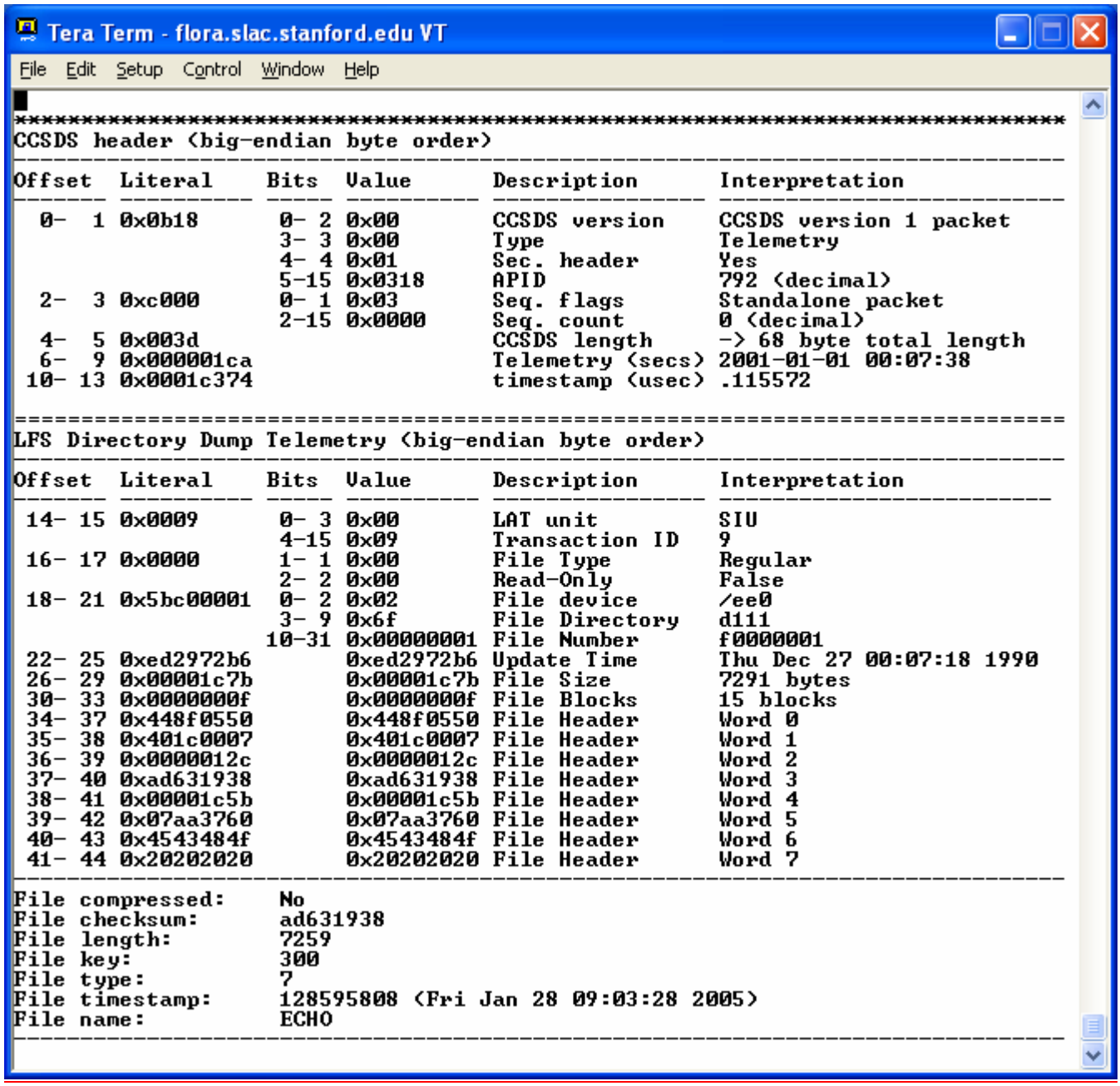
The demonstrator proceeds to the next set of transactions, repeating the process of displaying the contents of the LLFSDIRCREATE command and the LLFSROOTLIST and LLFSDIRLIST telemetry packets. The outputs of these transactions are very similar to those shown above, except that the output shows d112 as the directory identifier.

3.1.3 Outputs of File Management Demo Step 3

In Step 3, the demonstrator uploads echo.f (f0000001) to directory d111 on device ee0.

The next figure shows the LLFSDIRLIST packet returned by the LLFSDIRDUMP telecommand with which the demonstrator dumps the contents of d111 to confirm the test file was uploaded.

Figure 5: Confirming echo.f (File “f0000001”) was Uploaded to Directory d111 (TransactionID 9)



3.1.4 Outputs of File Management Demo Step 4

In Step 4, the demonstrator begins to upload echo.f to directory d112 on device ee1 using the same commands as in Step 3, but uses the LFILUPLCANCEL command to halt the upload.

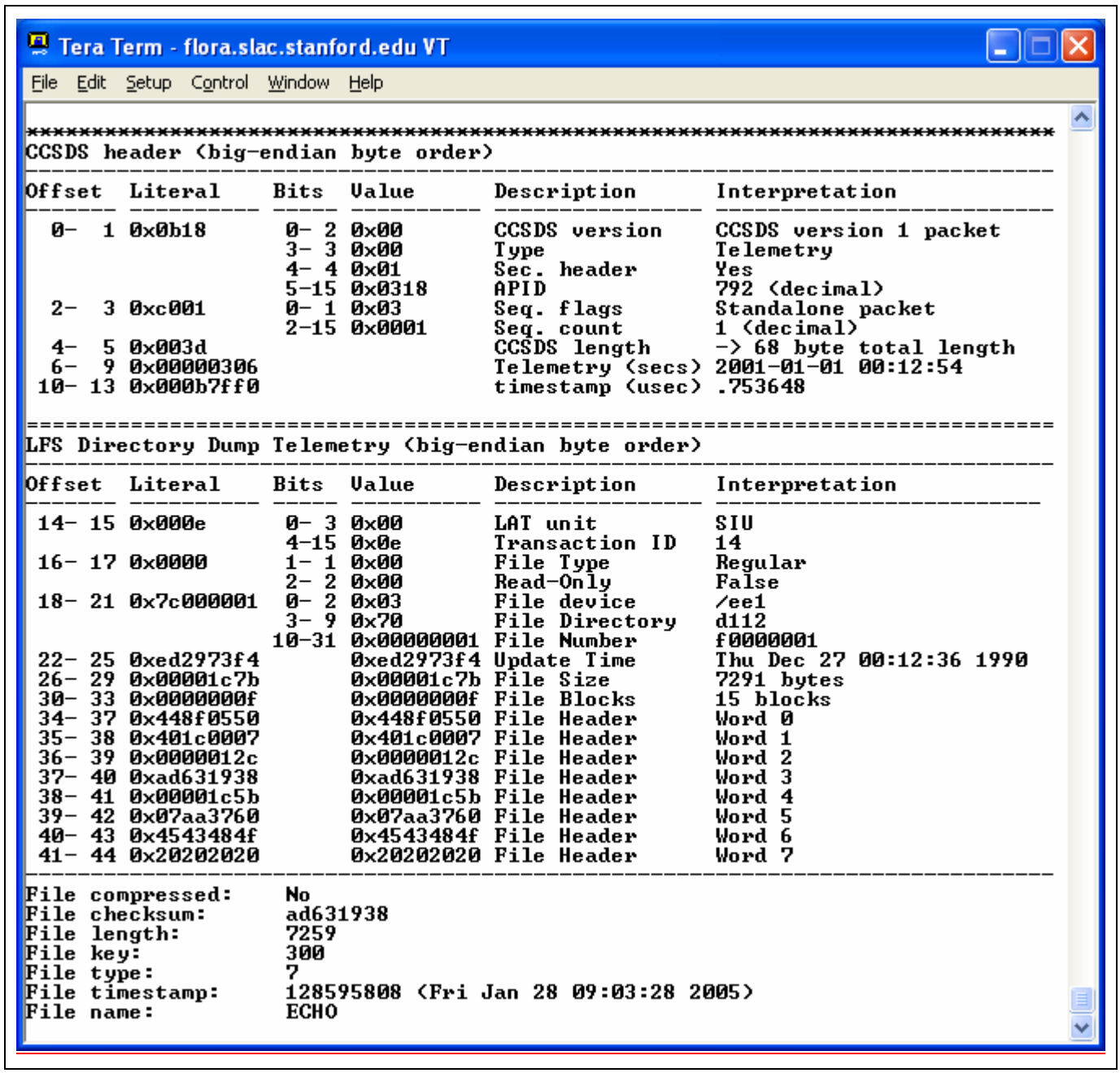
The demonstrator reviews LLFSDIRLIST packets returned by the LLFSDIRDUMP telecommand to check that that echo.f is not present in directory d112 – its upload to this directory was indeed canceled.

3.1.5 Outputs of File Management Demo Step 5

In Step 5, the demonstrator copies the test file echo.f (f0000001)from directory d111 on device ee0 to directory using the LLFSFILECOPY telecommand, then checks the contents of d112 using LLFSDIRDUMP.

The next figure shows the LLFSDIRLIST packet returned by the LLFSDIRDUMP telecommand used to dump the contents of directory d112. This packet shows that echo.f was indeed copied to directory d112

Figure 6: Confirming that the Test echo.f was Copied from Directory d111 to Directory d112 (TransactionID 14)



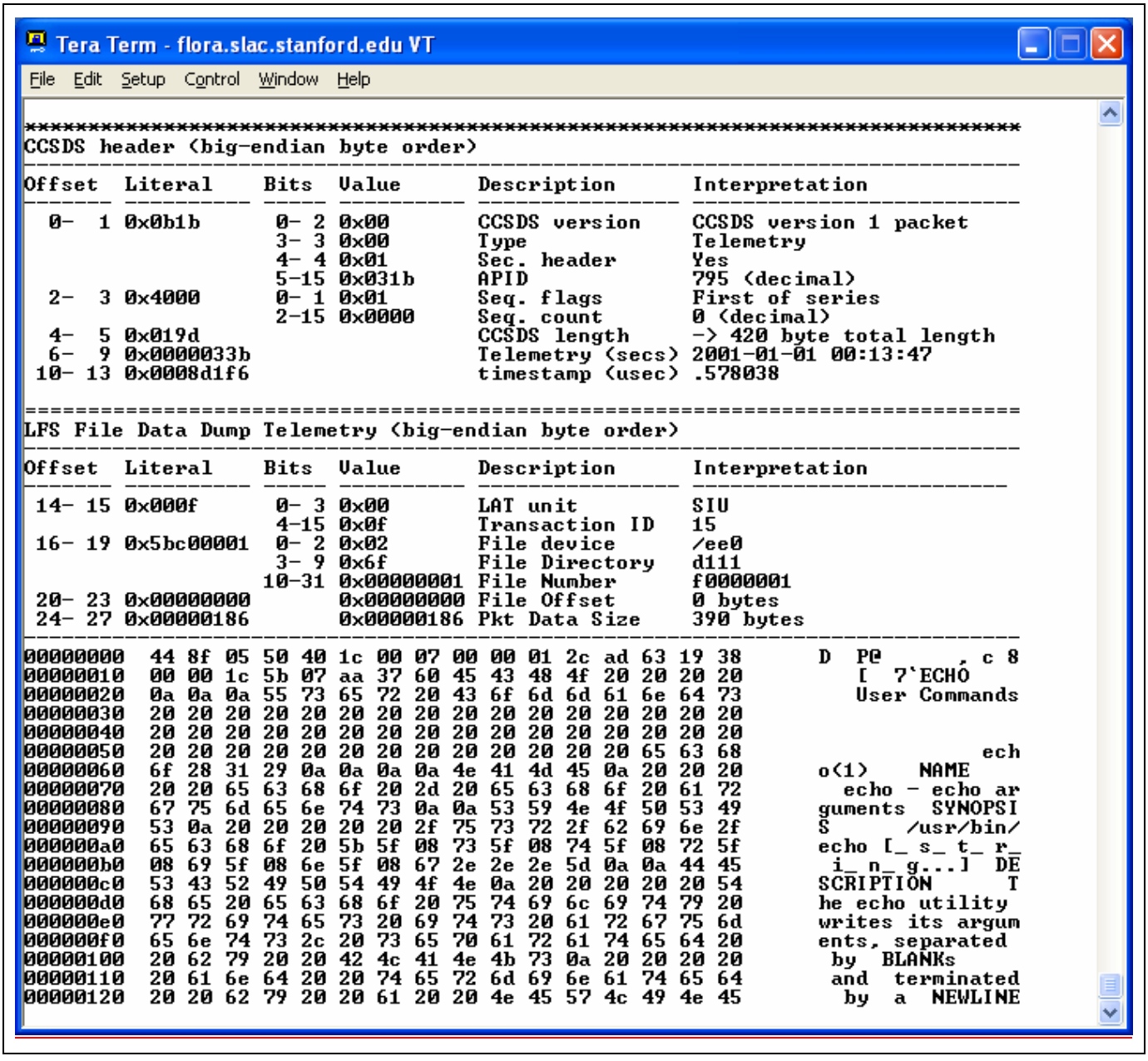
3.1.6 Outputs of File Management Demo Step 6

In Step 6, the demonstrator shows that the contents of files can be dumped to the ground with the LLFSFILEDUMPC telecommand and the data returns in LLFSDUMPCTDB telemetry packets.

During the demonstration, the demonstrator dumps the contents of test echo.f from both directory d111 and d112, but outputs from only one of these dumps are shown below.

In the next figure, the first portion of the contents of one of the LLFSDUMPCTDB telemetry packets are shown. Note in the bottom right portion of the data display, the text of the Unix man page entry for the “echo” command is displayed.

Figure 7: Dumping echo.f to Telemetry (TransactionID 15)



3.1.7 Outputs of File Management Demo Step 7

In Step 7, the demonstrator deletes the test echo.f file from both directory d111 on device ee0 and from directory d112 on device ee1. The demonstrator uses the LLFSFILEDELETE telecommand to accomplish the deletion and uses LLFSDIRDUMP telecommand and the resulting LLFSDIRLIST telemetry packets to confirm the deletion was successful.

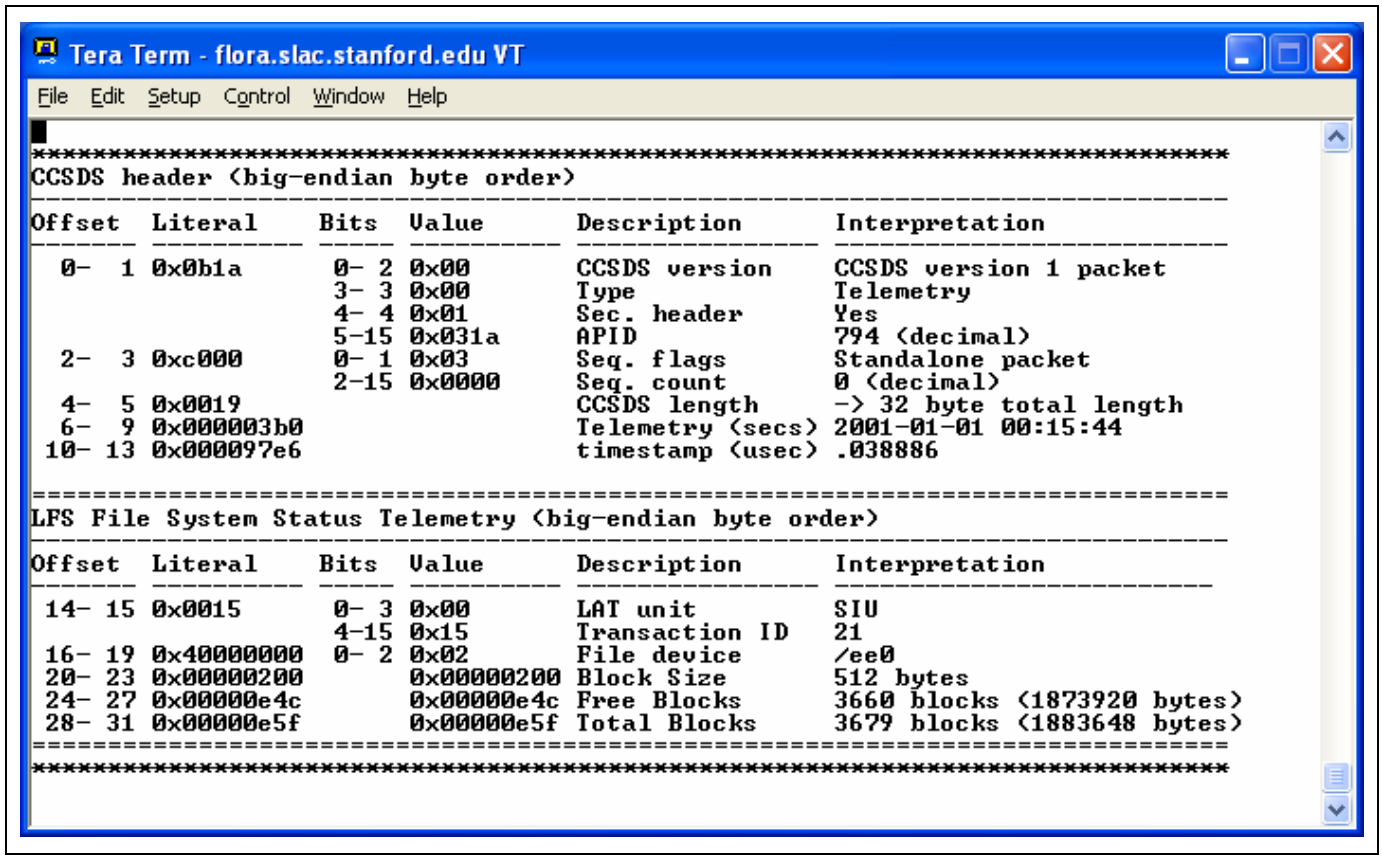
3.1.8 Outputs of File Management Demo Step 8

In Step 8, the demonstrator issues the LLFSSYSSTATUS telecommand to request a report on file system status contained in a series of LLFSSYSLIST telemetry packets.

During the demonstration, the demonstrator requests file system status for both devices, ee0 and ee1. However, the following figures show the outputs for only the ee0 file system status request.

The next figure shows the contents of the LLFSSYSLIST telemetry packet with data on the status of device ee0.

Figure 8: Reviewing the File System Status Report for Device ee0 (TransactionID 21)



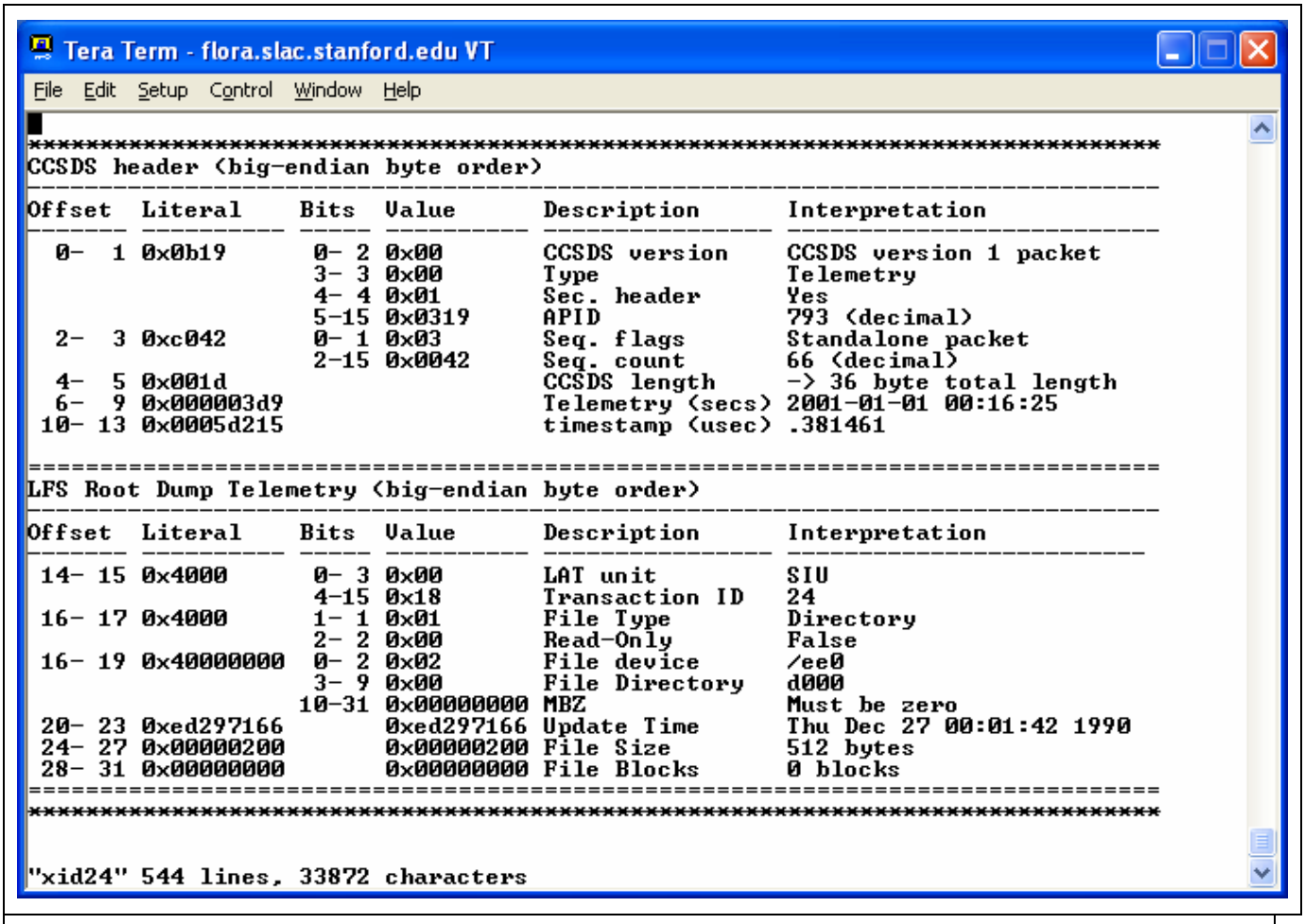
3.1.9 Outputs of File Management Demo Step 9

In Step 9, the demonstrator deletes the 2 directories, d111 and d112, created on the two devices, ee0 and ee1 using the LLFSDIRDELETE telecommand. The demonstrator then uses the LLFSDIRDUMP telecommand on the root directory, d127, on each device to confirm the directories were deleted.

During the demonstration, the demonstrator deletes 2 directories, however, the following figures show the outputs for only the directory d111 deletion sequence.

The demonstrator sends to LLFSDIRDUMP telecommand and reviews the resulting LLFSROOTLIST telemetry packets to confirm that directory d111 is now missing from the root directory, d127. A sample LLFSROOTLIST packet is shown below.

Figure 9: Confirming that the Root Directory, d127, on Device ee0 Does Not Contain d111 (TransactionID 24)



4 Demo Wrapup and Summary

The FSW Team thanks you for attending the demonstration and welcomes any questions about the software and hardware systems showcased. Comments or questions can be written in the following space.

5 Glossary

1553 – MIL-STD-1553B. Serial data bus specification; in particular, the serial data bus and data protocol implemented for the GLAST mission.

APID. CCSDS packet application identifier. A numerical code indicating the general type of data in a CCSDS packet.

Crate. Fond, generic term for development versions of Spacecraft Interface Units (SIUs) or Event-Processor Units (EPUs): custom-built, standalone on-board FSW processors and communications hardware units that control the LAT and communicate with the spacecraft (SIU), and process/filter instrument events (EPU). Crates are used for development purposes and will be replaced in the flight unit with single board computers with the same functionality.

GASU (Global-Trigger/ACD-EM/Signal-Distribution Unit). Portion of the FSW hardware suite that serves as the major hardware interface between data acquisition electronics on the LAT and other hardware and electronics that make up the FSW hardware package. The GASU contains the GEM, EBM, AEM, and CRU.

HKP. Real-time housekeeping telemetry data; telemetry data which relates to the health and safety of the LAT instrument.

LCB – The LAT Communications Board. A cPCI board that allows the internal components of the LAT to communicate with one another.

PID – Programmable Discrete. The RAD750 CPU board contains 32 channels of digital I/O. The primary boot code uses two channels configured as outputs.

PDU (Power Distribution Unit). Portion of the FSW hardware suite that manages power distribution from the spacecraft and monitors the health of other FSW hardware.

RTOS – Real Time Operating System. In particular the VxWorks 5.4 operating system used by the LAT.

SC – The GLAST Spacecraft. As built by Spectrum Astro. Refer to the GLAST LAT Instrument – Spacecraft Interface Control Document for the formal specifications of the SC as seen by the LAT.

SDIS – Spacecraft Data Interface Simulator. The SDIS represents significant portions of the interface between the LAT and the Spacecraft. The SDIS comprises one side of the SC C&DH system with a 1553 bus, LVDS Science Interface, and Discrete monitors and controls.

SDRAM. The RAD750 CPU board 128 MB of synchronous DRAM; the SDRAM serves as the RAD750 main memory.

SIB – Spacecraft Interface Board. The board in the SIU crates that contains the LAT 1553 remote terminal hardware.

SIU – Spacecraft Interface Unit. A type of single board computer (SBC) in the FSW hardware suite that acts as an interface between the spacecraft and the LAT.

SUROM – Startup ROM. 256 KB of EEPROM memory on the RAD750 CPU boards that holds the primary boot code; the SUROM is only programmable on the bench through the PPCI JTAG interface.

T&DF (Trigger and Dataflow System). Large LAT subsystem that provides gamma-ray identification, readout of the detector measurements, assembly of gamma-ray source location and energy measurements, and the streaming of data to the spacecraft. The T&DF subsystem contains the TKR, CAL, and ACD front-end electronics, Tower Electronics Modules (TEMs), ACD Electronics Modules (AEMs), Event Builder Module (EBM), Global Trigger (GLT), Global Trigger Electronics Module (GEM), and CPUs used for instrument configuration and data processing.

VxWorks. Computer operating system used on board the RAD750 processor.