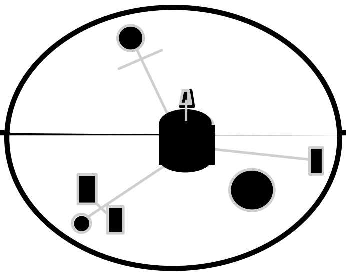
LUNAR PROSPECTOR SYSTEM

GROUND SOFTWARE USER'S GUIDE AND PROGRAMMER'S GUIDE

PREPARED IN ACCORDANCE WITH REQUIREMENTS OF
CONTRACT NAS2-14256, PHASE C/D



LOCKHEED MARTIN

LOCKHEED MARTIN MISSILES & SPACE
1111 LOCKHEED MARTIN WAY
SUNNYVALE, CA 84089-3504

APPROVALS

Prepared By:

William Jacobsen
Software Engineer

Approved By:

Norm Bennett
Software Lead Engineer

Robert W. Goldin
Systems Engineer

Thomas A. Dougherty
Project Manager

REVISION RECORD

REV	AUTHORITY	SIGNATURE	CHANGE PAGES	DATE
--	Author: Sys. Engr.: Proj. Mgr.:		DRAFT (Initial)	27 September 1996
A	Author: Sys. Engr.: Proj. Mgr.:		Update for document review by program (All)	14 November 1996
Ba	Author: Sys. Engr.: Proj. Mgr.:		Update for system testing in B150 (All)	12 February 1997
Bb	Author: Sys. Engr.: Proj. Mgr.:		More updates for system testing in B150 (All)	24 February 1997
Bc	Author: Sys. Engr.: Proj. Mgr.:		More updates for system testing in B150 (All)	26 February 1997
Bd	Author: Sys. Engr.: Proj. Mgr.:		More updates for system testing in B150 (All)	4 March 1997
Be	Author: Sys. Engr.: Proj. Mgr.:		More updates for system testing in B150 (All)	10 March 1997
Bf	Author: Sys. Engr.: Proj. Mgr.:		More updates for system testing in B150 (All)	12 March 1997
Bg	Author: Sys. Engr.: Proj. Mgr.:		More updates for system testing in B150 (All)	19 March 1997
Bh	Author: Sys. Engr.: Proj. Mgr.:		More updates for system testing in B150 (All)	3 April 1997
C	Author: Sys. Engr.: Proj. Mgr.:		Updates for mission operations (All)	1 December 1997

D	Author: Sys. Engr.: Proj. Mgr.:		Updates for Merge Software requirements changes (All)	1 February 1998

TABLE OF CONTENTS

1 SCOPE, FUNCTION, AND SYSTEM DESCRIPTION

1.1 Scope

This document provides User's Guides and Programmer's Guides for the Lunar Prospector ground system software. This document contains instructions which describe the startup of the various software modules, methods of commanding either the spacecraft or simulator, and various ways of storing, visualizing, and postprocessing telemetry information. In addition, this document includes details of the algorithms, timing constraints, data structures, and inputs and outputs, used by the various functions which comprise the ground system software applications. Appendices are provided for Lunar Prospector command mnemonics, formats, and argument types, as well as a sample command procedure.

1.2 Function

The Lunar Prospector ground system software consists of the following independent software modules:

SYBASE	commercial off the shelf software
OASIS-CC	commercial off the shelf software
CmdX	Lockheed Martin custom software (C)
st_vme_up	Lockheed Martin custom software (C)
C&DHS/VODS	Lockheed Martin custom software (C / FORTRAN)
TlmMod	Lockheed Martin custom software (C++)
st_vme_down	Lockheed Martin custom software (C)
FrameX	Lockheed Martin custom software (C)
PostProcTlm	Lockheed Martin custom software (C)

SYBASE is a commercially available database package which is used to store valid spacecraft command bit patterns as well as definitions, calibrations, and limits for telemetry channels and derived monitors.

OASIS-CC is another commercially available software package which was developed at the University of Colorado specifically for the processing and visualization of spacecraft command and telemetry data.

CmdX is mission-specific C code which performs BCH encoding and assembles the uplink command into a serial bit stream.

st_vme_up is a C application program which effects the uplink command transmission via the VME hardware, reads the IRIG-B time registers, and writes the command and time into a log file.

C&DHS/VODS is the module which contains the Lunar Prospector spacecraft command and data handling electronics simulation (C&DHS) as well as the vehicle and orbit dynamics simulation (VODS). The C&DHS routines are written in C while the VODS routines evolved from existing FORTRAN code.

TlmMod is a graphical user interface (GUI) based training and debugging tool written in C++ used to validate the operation of the simulation and also to insert anomalous values into the telemetry downlink data stream during operator training.

st_vme_down is a C application program which reads the downlink telemetry data via the VME hardware and appends the IRIG-B time prior to transmission to FrameX.

FrameX is mission specific C code analogous to CmdX, but used to decode downlink telemetry data received from the spacecraft. The resulting engineering and science data is passed on to OASIS-CC for further reduction and display.

PostProcTlm is an interactive C program for postprocessing telemetry archived by OASIS-CC. The user enters the directory containing the OASIS-CC output files, a subset of the telemetry monitor mnemonics, a time interval, and a new filename, and the program creates a new ASCII output file containing only the desired telemetry data.

1.3 Spacecraft System Description

The Lunar Prospector Spacecraft is of general cylindrical form and is spin stabilized. Propellant tanks and electronic equipment are located internal to the main S/C body. A solar array and 3 deployable booms are located external to the main S/C body. The scientific instruments are mounted on the booms. Medium Gain and Low Gain (omni) Antenna systems provide data downlink and command uplink communication capabilities between the S/C and the network of ground tracking stations. There is no flight software onboard the S/C.

2 APPLICABLE DOCUMENTS

2.1 NASA Documents

a. Spacecraft System Requirements for Lunar Prospector Mission, Spec. No. LITS-000-001, Draft (H), 4 August 1995

2.2 LMMS Documents

a. Interface Control Document - Spacecraft to Ground System, P086866, 2 January 1996

b. Lunar Prospector Mission Operations System Specification, P300S001, 24 August 1995

c. LP Telemetry Channel Assignment and Command Definition Plan, P086867, dated (TBD)

d. Lunar Prospector Command and Data Handling Electronics Specification, P108S002, dated (TBD)

e. Lunar Prospector Mission Command and Telemetry Handbook, P086867, Revision A, 15 March 1996

f. Lunar Prospector Mission Command and Data Handling Electronics Specification, P108S910 25 March 1996

g. EM-AVS-003, Command Generation Software Requirements, 8 May 1996

h. EM-AVS-004, Attitude Determination Software Requirements, 22 May 1996

2.3 CCSDS Documents

a. Telemetry, Summary of Concept and Rationale, CCSDS 100.0-G-1, December 1987

3 STARUP USING SPACECRAFT HARDWARE

This section provides instructions how to start up the Lunar Prospector ground system software, while using the spacecraft hardware during systems test in LMMS Building 150. The ground system software architecture, using the spacecraft hardware, is depicted in Figure 3.1.

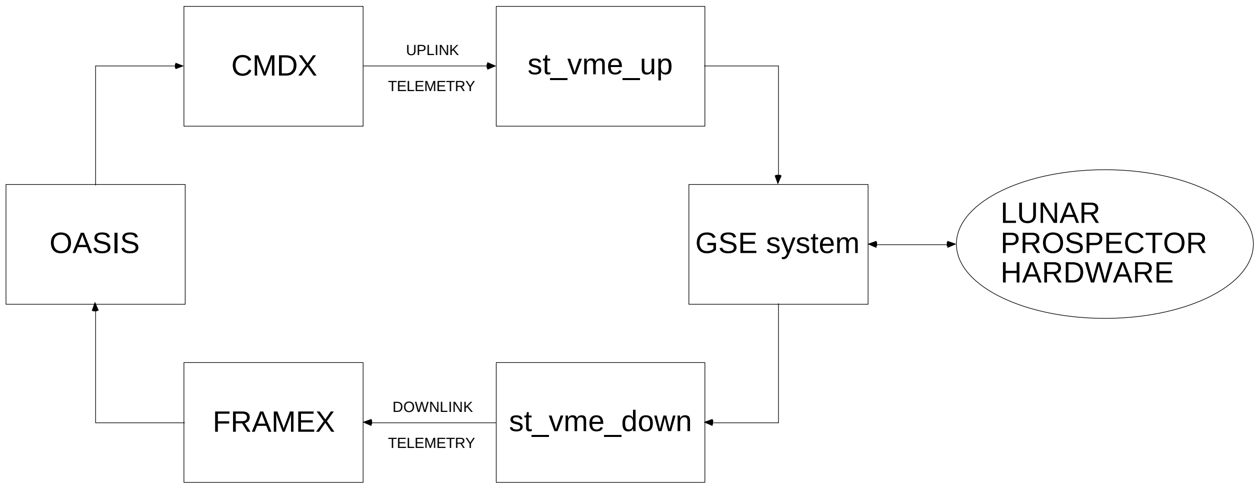


Figure 3-1 Systems Test Software Architecture

The following is a description of the commands required to start the ground software. **Note, that all commands in section 3 of this document are case-sensitive**. Note, the following steps assume the use of the “cheddar” workstation, in the Test Control Center (TCC) in Building 150. The “ricotta” workstation is located on the High Bay floor, next to the HP test stand, in Building 150.

1. Log onto workstation node “cheddar”, using the username “lunargrp” and the corresponding password (obtain from Software Group).

```
cheddar console login: lunargrp
             password: *******
```

2. In a command window, start OASIS, using the following commands:

```
/users/lunargrp% cd test
/users/lunargrp/test% start_oasis
```

This executes a script file (start_oasis.csh), which will start OASIS. The steps executed in this script are:

1) change the working directory to ~lunargrp/oasis_ops_1.0/oasis
2) execute a script file which sets up OASIS-needed environment variable

3) change the working directory to where OASIS executable resides ($RESOURCE_FILES)
4) start OASIS executable

OASIS takes approximately 20-30 seconds to start. When finished, recording of the messages, telemetry stream, and bridge information will be enabled. There will be messages in the “U MESSAGES” window stating this. These are the last messages in the window for start-up.

3. Now, the other ground system software items must be started on workstation node “ricotta”. Open a second command window on node “cheddar”, and log on to node “ricotta” via telnet, using the username “lunargrp” and the corresponding password (obtain from Software Group):

```
/users/lunargrp% telnet ricotta
username: lunargrp
password: *******
```

4. Next, open four command windows on node “ricotta”, positioning each command window as desired. This can be done with the following commands:

```
/users/lunargrp% setenv DISPLAY 129.197.46.2:0.0
/users/lunargrp% x1;x1;x1;x1           This opens 4 x-terminal type windows
```

At this point, the VME rack, containg the National Instruments Sun interface card, Avtec command/telemetry cards, and Datum IRIG-B card, must be powered on. The VME rack is connected to the workstation “ricotta” on the B150 High Bay floor.

5. In one of the windows, start the desired version of CmdX with the appropriate group of commands below:

To run the version of CmdX which produces uplinked commands with the correct Spacecraft ID and BCH codes, use the following commands:

```
/users/lunargrp% cd test
/users/lunargrp/test% source env1
/users/lunargrp/test% CmdX
```

To run the version of CmdX which produces uplinked commands with the incorrect Spacecraft ID and correct BCH codes, use the following commands. The Spacecraft ID is inserted in the command header by the application code.

```
/users/lunargrp% cd test
/users/lunargrp/test% source env1
/users/lunargrp/test% cmdx_bad_hdr
```

To run the version of CmdX which produces uplinked commands with the correct Spacecraft ID and incorrect BCH codes, use the following commands. Note, there are two BCH codes in each uplinked command. When running the following commands, both BCH codes are incorrect.

```
/users/lunargrp% cd test
/users/lunargrp/test% source env1
/users/lunargrp/test% cmdx_bad_bch
```

Note, “sourcing” the env1 file sets the necessary environment variables.

Once one version of CmdX has been started and it is desired to run a different version of CmdX, it is necessary to first shut down the ground system software using the procedure in section 7 of this document. Then, follow the procedure in this section, using the commands for the desired version of CmdX, to start the ground system software.

6. In another window, start st_vme_up, the uplink device driver, with the following commands:

```
/users/lunargrp% cd test
/users/lunargrp/test% source env1
/users/lunargrp/test% st_vme_up log_file
```

where log_file is the name for an output log file which logs st_vme_up messages. This name will be appended with the .log extension. If not provided, the user will be prompted for a name.

7. In another window, start st_vme_down, the downlink device driver, with the following commands:

```
/users/lunargrp% cd test
/users/lunargrp/test% source env1
/users/lunargrp/test% st_vme_down log_file
```

where log_file is the name for an output log file which logs st_vme_down messages. This name will be appended with the .log extension. If not provided, the user will be prompted for a name.

8. In the remaining window, start FrameX with the following commands:

```
/users/lunargrp% cd test
/users/lunargrp/test% source env1
/users/lunargrp/test% FrameX
```

9. In the OASIS stream control window, click the “TLM On” button. . This will start the telemetry processing by OASIS. This last step completes the ground system software startup process. Note, the CMD and CMDX_IN streams are switched on during OASIS start-up.

3.1 Miscellaneous System Test Information

3.1.1 Copying Command Procedures between Machines

The official, "checked-out" command procedures are stored in the following directory on node "cheddar":

```
~lunargrp / flt_oasis / oasis / procedures / testcase
```

The command procedures under development are stored in the following directory on node "cheddar":

```
~lunargrp / flt_oasis / oasis / procedures / dev
```

3.1.2 Printing OASIS Displays

To print an OASIS display with its current display contents on a printer, execute the following commands:

From the CSTOL prompt,

```
CSTOL>  SNAP display_name
```

An example of display_name is "acs1".

This will place the file to be printed in the following directory:

```
~oasis / logs / snaps
```

To print the file, issue the following commands on node "cheddar":

```
% cd ~lunargrp/flt_oasis/oasis/logs/snaps
% lp file_name
```

3.1.3 Turning Off OASIS Limits

To turn off a limit associated with a mnemonic, issue the following command:

```
CSTOL>  UPDATE LATEST_DATA HILO_ENABLED = FALSE WHERE &
         ITEM_NAME = "MNEMONIC"
```

To turn off a set of limits associated with a subsystem, issue the following command:

```
CSTOL>  UPDATE LATEST_DATA HILO_ENABLED = FALSE WHERE &
             EXTERNAL_ELEMENT = “SUBSYSTEM”
```

3.1.4 Sending OASIS Data to Science Instrument PCs

To switch on the data streams from OASIS to the Science Instrument PC s , issue the following commands:

```
CSTOL> switch on magstr
CSTOL> route packet_x to magstr

CSTOL> switch on specstr
CSTOL> route packet_x to specstr
```

where x = 1, 2, 3, or 4

packet_1 = delayed science stream
packet_2 = prior science stream
packet_3 = delayed engineering stream
packet_4 = prior engineering stream

To switch off the data streams from OASIS to the Science Instrument PC s , issue the following commands:

```
CSTOL> switch off magstr
CSTOL> switch off specstr
```

Note: the following connection information is required for the PC’s to sync up with OASIS:

cheddar IP address: 129.197.46.2
mag/er port: 7001
spectrometer port: 7002

4 STARTUP USING VEHICLE AND ENVIRONMENT SIMULATOR

This section provides instructions how to start up the Lunar Prospector ground system software, using the Vehicle and Environment Simulator (VES) in place of the actual hardware. These procedures should be used to teach subsystems engineers the methods of commanding the spacecraft and of acquiring and manipulating telemetry information in preparation for systems test. All of the necessary software to run the VES islocated on workstation "feta", in LMMS Building 251 in Palo Alto, CA. The ground system software architecture, using the VES, is depicted in Figure 4.1.

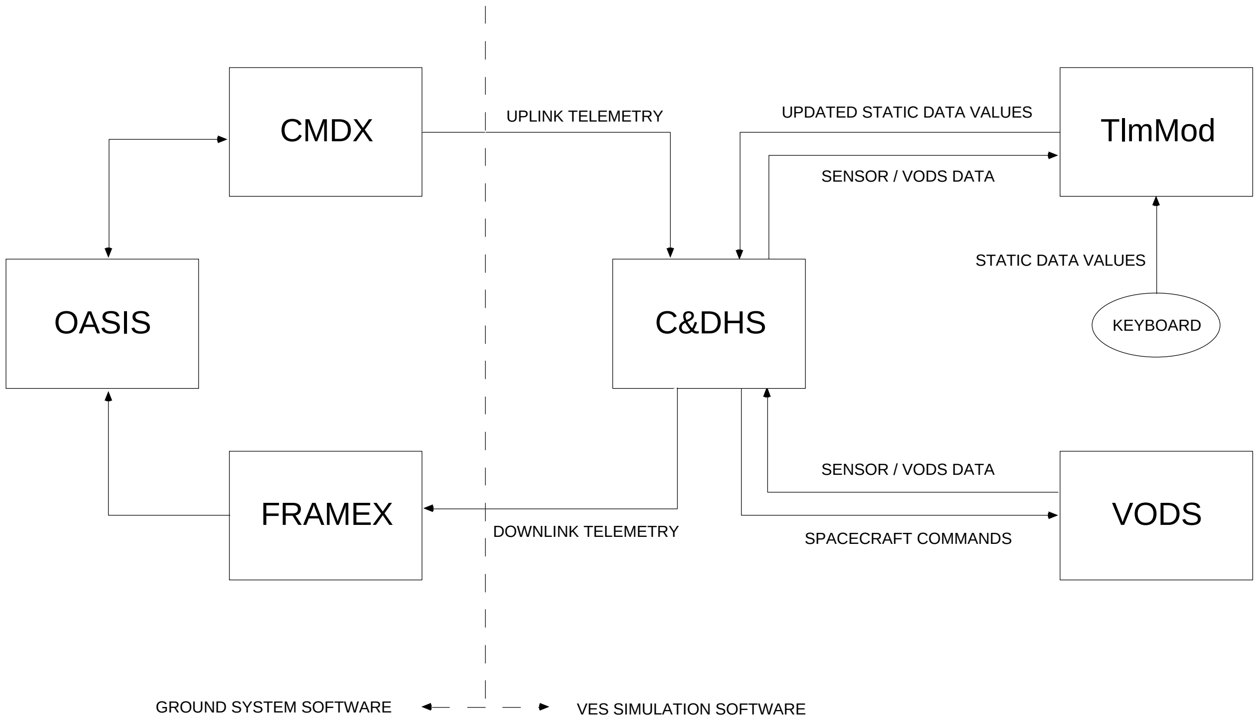


Figure 4-1 VES Software Architecture

Following is a summary of commands required to start the ground system software using the VES.
Note, that all commands in section 4 of this document are case-sensitive.

1. At SUN console login, log onto workstation

```
        console login: lunargrp
             password: *******
```

2. In upper left window, start the necessary processes

```
       /home/lunargrp/test% cd ~lunargrp/ves
```

```
        /lunargrp/ves% source lab_on.sh
```

This script starts all necessary processes. This takes approximately 30-40 seconds. The processes started are OASIS, CmdX, Cdh, and FrameX.

3. In OASIS stream control window, click TLM On button to enable telemetry processing

The following is a more detailed description of the commands required to start the ground system software using the VES:

Using the username "lunargrp" and the corresponding password (obtain from Software Group), log on to workstation.

After logging in, the working directory is /home/lunargrp/test. Enter the following to go to the home directory:

```
        /home/lunargrp/test% cd lunargrp/ves
```

To start the necessary processes, use the provided script. This is done with the following:

```
        /home/lunargrp/ves% source lab_on.sh
```

This script does the following:

1) source a script to set OASIS-needed environment variables
2) source a script to set other various environment variables
3) start OASIS
4) start CmdX
5) start Cdh
6) start FrameX

Once FrameX starts, click the TLM On button on the STR_CTL OASIS panel. This will start the telemetry processing by OASIS. The CMD and CMDX_IN streams are switched on automatically in the OASIS start-up. Telemetry is now ready to be processed by OASIS.

Note: One way to test OASIS telemetry functions in the absence of real, dynamic data is to insert static data in the downlink stream using a program named TlmMod. In order to start up TlmMod, enter the following in the center window:

```
        /home/lunargrp/test% cd
        /home/lunargrp% source flt_tlmmod_on.sh
```

This script sets necessary environment variables and starts the TlmMod executable.

Instructions how to operate TlmMod are given in the next section entitled "Modifying Downlink Telemetry Data Using TlmMod."

4.1 Modifying Downlink Telemetry Data Using TlmMod

This section provides instructions how to view and insert new values in the downlink telemetry stream using the TlmMod program.

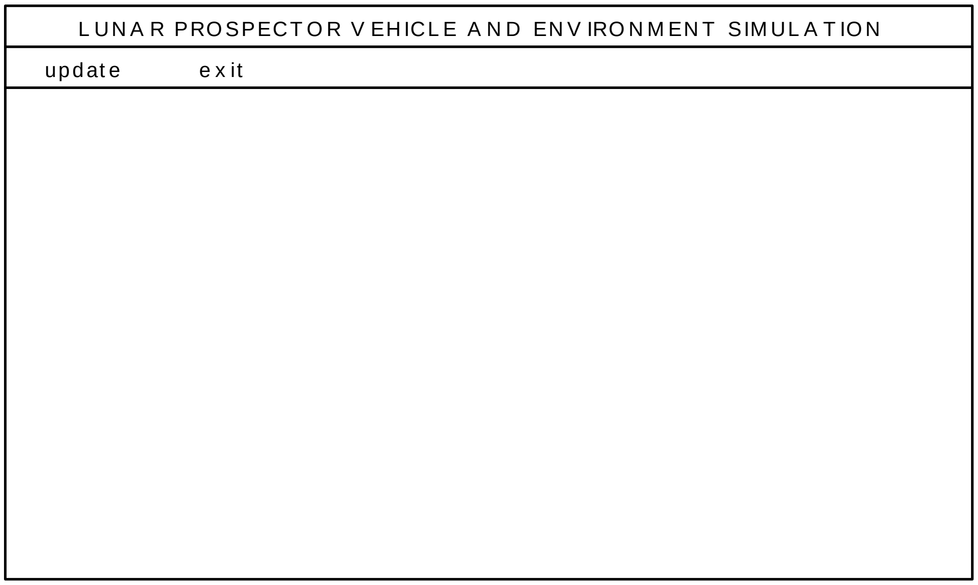


Figure 4.1-1 TlmMod Startup Window

The window in Figure 4.1-1 window appears when the TlmMod program starts up and contains the two menu options "update" and "exit."

To view and update existing telemetry data, click on the "update" menu.

To exit TlmMod, click on the "exit" selection. If the program does not terminate immediately, move the mouse over both this window and the window from which TlmMod was started up.

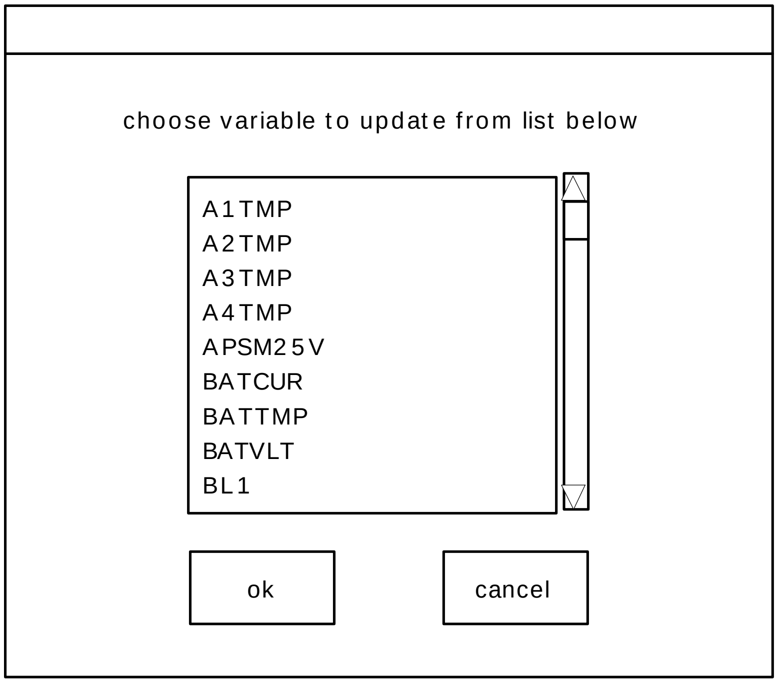


Figure 4.1-2 TlmMod Parameter Selection Window

When the “update” selection is clicked in the main menu, the parameter selection window in Figure 4.1-2 appears which contains a scrollable list box of mnemonics corresponding to each telemetry parameter contained in the downlink stream.

To view and/or change a parameter’s value, use the mouse to highlight the desired parameter and click the “ok” button.

To return to the main menu, click the “cancel” button.

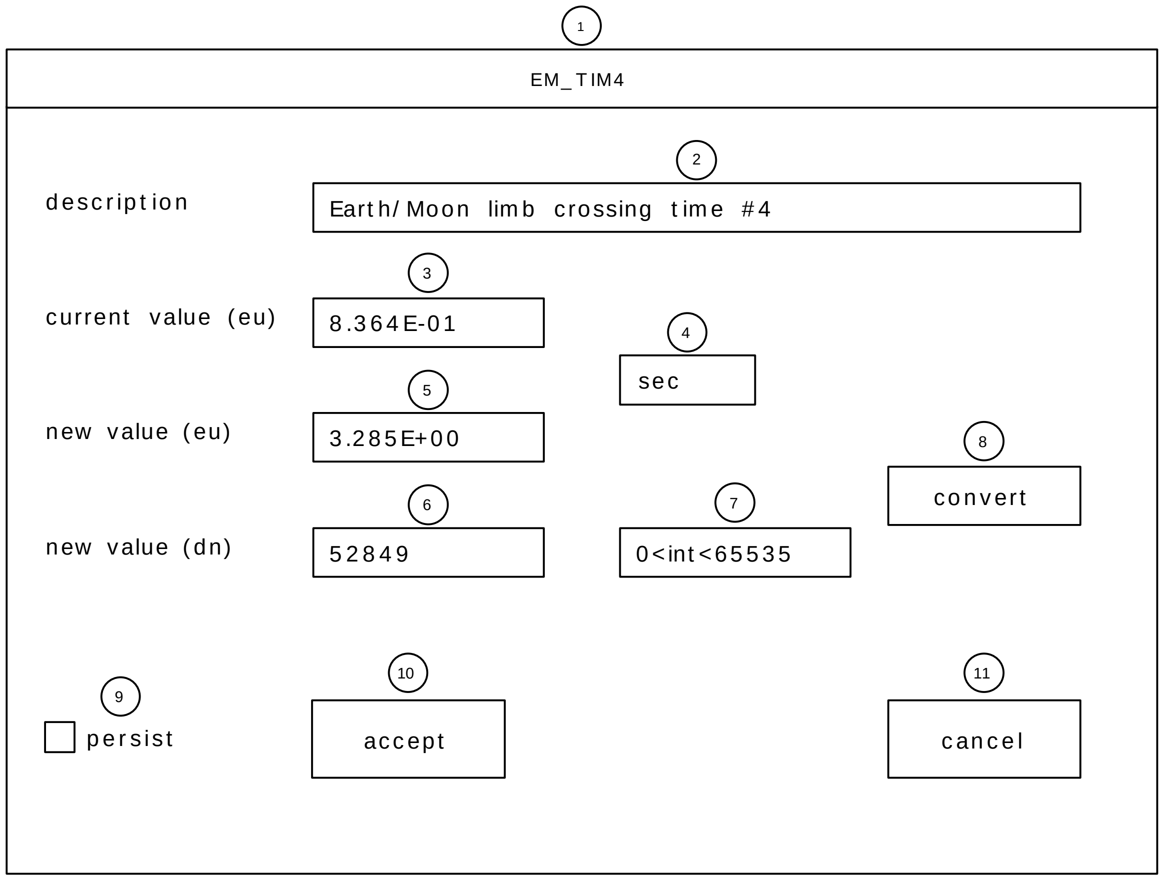


Figure 4.1-3 TlmMod Data Viewing and Modification Window

The window in Figure 4.1-3 is used for viewing and modifying information related to a particular downlink telemetry parameter. For viewing data without modification, only items 1 through 4 are of interest. The text item indicated by bubble 1 is the parameter's mnemonic name. Item 2 contains a more detailed description of the item being viewed. Item 3 is a data field which displays the calibrated equivalent of the raw data number currently being downlinked to FrameX and OASIS. The calibrated value uses the engineering units specified in item 4. The refresh rate of the TlmMod current value display is two seconds, the same as for downlink telemetry.

In order to downlink a modified value of the parameter to FrameX and OASIS, items 5 through 10 must be used. Item 6 is the data entry box where the user may specify a new value for the parameter being viewed. Clicking on item 6 and using the keyboard, the user may enter a raw data number of the type and range specified by item 7. In order to convert the raw data number to the engineering units specified in item 4, click on the "convert" button (item 8). User input of an incorrect type or out of range will result in an error message being printed in item 5. For valid data, the calibrated equivalent of the raw data number entered in item 6 is displayed in item 5. Before accepting the new data, the user may use the “persist” button (item 9) to specify the lifetime of the modified value in the downlink telemetry stream. If the button is clicked and appears pressed “in” or “down,” the new value will be maintained in the downlink stream until the persist button is turned off or the TlmMod program is terminated. If the button appears “out” or “up,” the new value will be written to the downlink stream for a single telemetry frame only before reverting to its original value computed by the VES simulator. In either case, no action is taken until the “accept” button (item 10) is clicked.

Clicking on the “cancel” button (item 11) will exit the window leaving the value of the parameter unchanged.

4.2 C&DHS / VODS Initialization File

The C&DHS/VODS software reads an initialization file named “init.dat” located in the /home/lunargrp directory on the Lunar Prospector Sun workstatio. This ASCII text file defines the startup contents of the downlink telemetry stream in groups of two byte hexadecimal values using the format of Figure 4.2.1 below. To maintain data consistency with other VES software modules, the user should change only the hex values in the right hand column.

```
description                     hex value
First_Parameter                 0x01
Second_Parameter                0x02

etc.                            etc.

Last_Parameter                        0x6a
Eng_Default                     0xff
Sci_Default                     0xbb
```

Figure 4.2-1 VES Initialization File Format

5 COMMANDING

All commands to be sent to the space vehicle which are defined in the SYBASE data base can be sent by using an OASIS macro. Other commands destined for the vehicle but not defined in SYBASE, can be sent by issuing the desired bit pattern. Both of these methods are done from the CSTOL prompt in the CSTOL_PROMPT window. **Unless otherwise noted, all command input is case insensitive.**

OASIS command macros allow the user to send commands without knowing CSTOL commanding syntax. Each command stored in the SYBASE data base is assigned a mnemonic which is used as the name of the macro. For discrete commands, the user simply enters the mnemonic at the CSTOL prompt and the proper bit sequence is sent. If a command has arguments, these follow the mnemonic on the command line. The arguments must be in the correct order and entered as floating point values. A list of all mnemonics is in Appendix A.

If a command does not have a macro assigned to it, the desired bit pattern can be issued through OASIS. Currently this is necessary for commands being sent to command the science instruments. The bit pattern consists of the following:

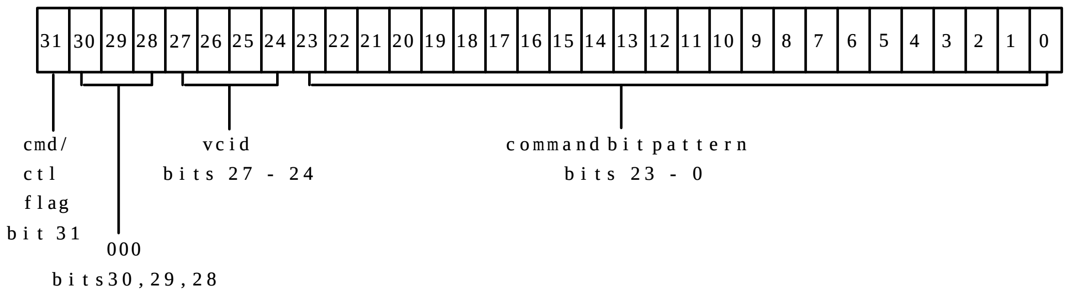


Figure 5.0-1 Command Bit Pattern

The following gives an example of each commanding type:

Discrete Command via Mnemonic:

```
      CSTOL> T3600       ; sends the science format command
```

Command with Arguments via Mnemonic:

```
      CSTOL> HALFREV 10.0 ; sends HALF REVOLUTION parameter cmd with argument of
10.0
```

Command via hexadecimal bit pattern:

```
CSTOL> SEND H#460B0000 TO cmdx   ; sends the science format command
```

It is possible to build a script of commands in order to provide more autonomous commanding. These scripts are known as CSTOL procedures and their file name must have the “.prc” extension. Any available Lunar Prospector command may be used in these procecedures. There are also several other CSTOL directives which are useful in the creation of these procedures. Several directives allow the user to control the flow of the procedure. The following will give a brief overview of the most commonly used directives. See the OASIS-CC CSTOL Reference Manual (Section 10) for a complete discussion of the available directives. This overview is not meant to be a complete discussion of all CSTOL directives.

COMPILE

The COMPILE directive compiles the given procedure. The syntax is:

```
COMPILE aproc
```

The file extension for the procedure name should not be entered.

DECOMPILE

The DECOMPILE directive decompiles the given procedure. If a compiled procedure is changed, the procedure must be decompiled and then recompiled for those changes to take affect. The syntax is:

```
DECOMPILE aproc
DECOMPILE ALL PROCEDURES
```

The second example will decompile all compiled procedures.

GO

The GO directive cannot be used within a procedure. It is used from the CSTOL command line to resume procedure execution (such as from a WAIT). When used with TO, the procedure will resume execution at the given line or label.

PROC & END PROC

The PROC directive marks the beginning of a procedure. It is followed by the procedure name. The END PROC directive marks the end of a procedure. For example:

```
PROC aproc
:
:
END PROC
```

RECORD & RECORD OFF

The RECORD directive enables the logging of event messages, incoming streams, and bridge streams. The RECORD OFF stops this logging. This syntax is:

```
RECORD EVENTS       ; record event log
RECORD tlm          ; records the incoming telemetry stream
RECORD tlm_ascii    ; records the pre-defined bridge named tlm_ascii.
RECORD OFF EVENTS   ; stops logging events
RECORD OFF tlm              ; stops recording of telemetry stream
RECORD OFF tlm_ascii        ; stops recording of bridge
RECORD OFF ALL              ; stops any active logging
```

RETURN
The RETURN directive terminates the current procedure and returns to previous one. RETURN ALL terminates all active procedures.

SNAP
The SNAP directive will save the current contents of the given page. If used with the RESOURCE attribute, the snap is saved graphically. Without the RESOURCE attribute, an approximate image of the page is created. The syntax is:

```
SNAP page_name                    ; the page does not need to be displayed
SNAP page_name RESOURCE   ; the page does need to be displayed
```

START
The START directive initiates the execution of a procedure. If the procedure is not compiled, the START directive will initiate the compilation.

WAIT
The WAIT directive suspends execution of a procedure. It can be a indefinite or conditional wait. The conditions on a wait can be a delta time, clock time, or an expression checking some variable. Examples are:

```
WAIT                ; indefinite wait
WAIT ::1.0          ; delta time - waits 1.0 second
WAIT /-13:00:00     ; clock time - waits until one o’clock today
```

See the CSTOL Reference Manual for syntax of delta times, clock times, and expressions.

WRITE
The WRITE directive inserts a message into the event log and may be used (for example) to indicate the status of the procedure. The form of the WRITE command is as follows:

```
WRITE “This is a message”
WRITE pln tank_1_temp    ; outputs tank 1 temp to the message window
```

For writing a monitor value, use the subsystem (pln) and mnemonic (tank_1_temp).

The following control directives can also be used within a procedure:

```
IF              ELSE IF         ELSE            END IF
LOOP            ESCAPE          END LOOP
```

See the CSTOL Reference Manual for their exact use.

The general form of a procedure is as follows:

```
proc xxxx
       command_1
       command_2
       ...
       command_n
end proc
```

where the procedure is named *xxxx.prc* (all lower case) and must reside in the OASIS directory named /home/lunargrp/oasis/procedures/testcase. The procedure must be entered into an ASCII file independently of the OASIS program using any available text editor. The procedure name may not contain periods and must have at most 16 characters excluding the .prc extension. An example procedure is listed in Appendix C.

In order for the procedure to be used, OASIS must know where to find it and compile it. This is accomplished with the following two commands:

```
CSTOL> new_proc xxxx ./testcase
CSTOL> compile xxxx
```

where *xxxx* is the procedure name (without .prc extension) and ./testcase is the directory name. The following command starts the procedure:

```
CSTOL> start xxxx
```

In order to remove a procedure from OASIS, make sure the procedure has stopped running by typing RETURN at the CSTOL prompt and then entering the following command:

```
CSTOL> del_proc xxxx
```

where *xxxx* is once again the procedure name (without .prc extension).

6 DATA RECORDING AND PROCESSING

To record and process telemetry information, the user must input a variety of commands, all of which should be entered at the CSTOL prompt unless otherwise specified. **Unless otherwise noted, all data recording and processing input is case insensitive.**

OASIS provides the capability to record the messages which appear in the message window. This recording is started and stopped by issuing the following commands:

```
CSTOL> record messages
.
.
.
CSTOL> record off messages
```

The resultant file is placed in the /home/lunargrp/oasis/logs/messages directory. The name of the file is:

```
fyy_mmm_dd_hh_mm_ss.event_messages
```

where yy is the year, mmm is the month, dd is the day, hh is the hour, mm is the minutes, and ss is the seconds at which the file was created.

If desired, a message file can be closed and a new file started without loss of messages. The command for this is the following:

```
CSTOL> switch recording messages
```

The original file will then be closed and a new file, with the above file name format, will then be started.

OASIS provides the ability to write decomposed data to either binary or ASCII files. The Lunar Prospector program currently writes all defined telemetry items (real and derived) to a file in ASCII format.

Data is recorded in the /home/lunargrp/oasis/bridge directory using two files with the names

```
fyy_mmm_dd_hh_mm_ss.tlm_ascii_data
fyy_mmm_dd_hh_mm_ss.tlm_ascii_ident
```

where yy is the year, mmm is the month, dd is the day, hh is the hour, mm is the minutes, and ss is the seconds at which the file was created. The ident file lists the monitors and the ordering which make up the bridge while the data file is the ascii output.

To start and stop recording data, enter the following:

```
        CSTOL> record tlm_ascii
        .
        .
        .
        CSTOL> record off tlm_ascii
```

The current data file can be closed and a new file started without loss of data with the following command:

```
        CSTOL> switch recording of tlm_ascii
```

It may be necessary to record an OASIS session so that it can be replayed in the future. The way to start and stop recording the incoming telemetry stream is to enter the following, once again at the CSTOL prompt:

```
        CSTOL> record tlm
        .
        .
        .
        CSTOL> record off tlm
```

This will create a file in the /home/lunargrp/oasis/logs/telemetry directory with the name

```
        fyy_mmm_dd_hh_mm_ss.tlm
```

where yy is the year, mmm is the month, dd is the day, hh is the hour, mm is the minutes, and ss is the seconds at which the file was created. Again, the following command will stop a current file and start a new file:

```
        CSTOL> switch recording of tlm
```

To retrieve (replay) your session make sure all streams are off. This can be done by clicking the off buttons for TLM, CMDX_IN, and CMD, preferably in that order. Enter the following:

```
        CSTOL> retrieve tlm from file
```

If retrieving prior data:

```
        CSTOL> switch on packet_2   ; prior normal (science) packet
        CSTOL> switch on packet_4   ; prior engineering packet
```

If retrieving delay data:

```
        CSTOL> switch on packet_1   ; delay normal (science) packet
        CSTOL> switch on packet_3   ; delay engineering packet
```

where *`file`* is the name of the recorded file without any extension.

When retrieval is done, enter the following:

```
        CSTOL> retrieve off tlm
        CSTOL> switch off packet_2
        CSTOL> switch off packet_4
or
        CSTOL> switch off packet_1
        CSTOL> switch off packet_3
```

7 SHUTDOWN

When using the spacecraft hardware, terminate the ground system software execution as described below. Essentially, the shutdown procedure is the reverse of the startup procedure. In this manner, the sockets are closed in the reverse order in which they were started, reducing the likelihood of sockets being hung up, resulting in a minimum of down time between completion of the shutdown procedure and commencement of the startup procedure. First, stop the st_vme_up process by pressing the “control” and “c” keys simultaneously in the command window where st_vme_up is executing. Next, stop the CmdX process by pressing the “control” and “c” keys simultaneously in the command window where CmdX is executing. Next, turn off the OASIS streams by clicking the off buttons for TLM, CMDX_IN, and CMD, in the OASIS stream control window, in the specified order. The FrameX process will be stopped by the prior step, specifically by clicking the off button for the TLM OASIS stream. Next, stop the st_vme_down process by pressing the “control” and “c” keys simultaneously in the command window where st_vme_down is executing. Last, quit OASIS by clicking the QUIT button from the main panel.

When using the Vehicle and Environment Simulator (VES), terminate the ground system software execution as described below. First, make sure all OASIS streams are off. This can be done by clicking the off buttons for TLM, CMDX_IN, and CMD, in the OASIS stream control window, preferably in that order. This will also stop the other processes which may be running. If TlmMod is still running, return to the main menu and click exit. Next, quit OASIS by clicking the QUIT button from the main panel. Note, shutting down OASIS will automatically terminate whatever other programs are running,with the exception of TlmMod, which must be shut down as described above.

8 TROUBLESHOOTING

It is possible that the startup sequence may not run properly if OASIS is shut down and then immediately restarted. The source of this trouble lies with the communications links (sockets) between the various processes. If OASIS, CmdX, Cdh, FrameX, etc. are not shut down correctly, the sockets will hang for roughly four minutes until they time out and reset themselves. To check whether any sockets are hung up, enter the following command in any window:

```
/home/lunargrp/test% netstat -a | grep feta.601
```

If any lines contain the text “TIME_WAIT” in the right hand column, repeat the command every minute or so until those lines no longer appear. At that time, the startup sequence should run properly.

9 POSTPROCESSING SOFTWARE

There are two postprocessing software packages used on Lunar Prospector. PostProcTlm is used to analyze engineering telemetry data. The inputs to the program are OASIS generated bridge files. It sorts and plots user specified telemetry monitors over selected time ranges and generates readable sort files and plots.

The Merge software generates a raw, time ordered binary data file containing all science and engineering data sent to the ground stations. To handle corrupted VCDU counts, a preprocessing software program, Preprocessmerge, is run prior to running the Merge software package to take out ‘bad’ VCDU counts. The inputs to the preprocessing program are raw FrameX generated files. The inputs to the Merge software these preprocessed files. The Merge performs a comparison between the prior and delayed data to ensure consistent data quality and in the event of a miscompare, uses the signal-to-noise ratio to select which data (prior or delayed) to put in the output file.

9.1 PostProcTlm Post Processing Telemetry Analysis Software

PostProcTlm is an interactive C program, on node “feta”, for postprocessing telemetry archived by OASIS-CC. The user enters the directory containing the OASIS-CC output files, a subset of the telemetry monitor mnemonics, a time interval, and a new filename, and the program creates a new ASCII output file containing only the desired telemetry data.

Note: Before running the PostProcTlm tool, directories must be created to copy the data files into. These must be located under the “postprocessing” directory. These subdirectories must use the naming convention defined below:

fyy_mon_dd_hh_mm_ss.tlm (eg. f97_mar_04_16_08_39.tlm)

The year (yy), month (mon), day (dd), hour (hh), minute (mm), and second (ss) must be identical to the ‘data’ and ‘ident’ file names that will get copied into the directories.

Copy the ‘ident’ and ‘data’ files into the appropriate subdirectory. This must be performed before running PostProcTlm.

Type the following command at the UNIX shell prompt:

```
% cd ~lunargrp/postprocessing
% postproctlm
```

The program will display instructions and a prompt for an OASIS-CC output directory:

```
Please enter the appropriate information after each prompt.  To complete each
entry, press the carriage return key.


Telemetry directory (Enter ‘&l’ to list the telemetry
directories, or ‘&q” to quit):
```

If you list the directories, the program will display them and return to the prompt. An example would be:

```
Telemetry directory (Enter ‘&l’ to list the telemetry
directories, or ‘&q” to quit): &l

f97_jan_27_09_28_27   f97_mar_11_01_09_57   f96_oct_26_16_08_39

Telemetry directory (Enter ‘&l’ to list the telemetry
directories, or ‘&q” to quit):
```

When you enter a valid directory name, such as f97_jan_27_09_28_27, the program will check the existence of the two ASCII OASIS-CC output files in that directory and display them. The first is the identity file (_ident), which contains such information as the mnemonics, units, and formats of the monitors. The second file is the data file (_data), which contains the data for the monitors.

```
OASIS-CC output directory (Enter ‘&l’ to list the output
directories): f97_jan_27_09_28_27

Directory f97_jan_27_09_28_27 contains the following bridge identity and
data files:

  f97_jan_27_09_28_27.tlm_ascii_ident
  f97_jan_27_09_28_27.tlm_ascii_data

Telemetry subset table filename (Enter ‘&l’ to list old table filenames in the
selected directory, or ‘&q’ to quit):
```

If you list the PostProcTlm output files, the existing output files will be displayed and the prompt returned. These are files that PostProcTlm has previously generated. If no file have been generated nothing gets listed. An example would be:

```
Telemetry subset table filename (Enter ‘&l’ to list old table filenames in the
selected directory, or ‘&q’ to quit): &l

01                                   06
02                                   07
03                                   08
04                                   cdh_temp
05                                   eps

Telemetry subset table filename (Enter ‘&l’ to list old table filenames in the
selected directory, or ‘&q’ to quit):
```

Enter the subset table filename. (This can be any name the user wishes to use.)

If you enter a filename that already exists, the program will display a warning and a Yes / No prompt for overwriting the file with that name. If you enter “N” at the new prompt, the program will return to the prompt for a telemetry subset filename. If you enter “Y”, the program will append the string “.orig” to the original file, create the new file, and continue.

After the filename has been defined, enter the number of monitors to be postprocessed. (**Note**: this number must be between 1 and 7)

```
Number of telemetry monitors to be postprocessed (Enter a number between 1 and
7, or ‘&q’ to quit):2

Monitor mnemomic 1 (Enter ‘&l’ to list the mnemonics in the selected
telemetry, or ‘&q’ to quit):
```

You must now enter the mnemonic name. Otherwise, list the name to choose from.

```
Monitor mnemomic 1 (Enter ‘&l’ to list the mnemonics in the selected
telemetry, or ‘&q’ to quit): &l

APS_M25VDC_VOLT
APS_TEMP
BATT_CHG_STATUS
BATT_CURRENT
BATT_TEMP
BATT_VOLTAGE
BURN_DURATION
CAR_LOCK_STAT
CDHE_TEMP
DEMOD_LOCK_STAT
E_M_SENS_TEMP1
E_M_SENS_TEMP2
E_M_SENSOR_T1
E_M_SENSOR_T2
E_M_SENSOR_T3
E_M_SENSOR_T4
E_M_SENSOR_T5
ELEC_12VDC_VOLT
ELEC_28VDC_CURR
ELEC_5VDC_VOLT
ELEC_TEMP

--More--(Enter ‘q’ to quit)
```

If you press the return or enter key, the program will list twenty-one more mnemonics. You can repeat the process until you reach the bottom of the list, or until you quit. The program will then return to the prompt for a mnemonic:

```
APS_M25VDC_VOLT
APS_TEMP
BATT_CHG_STATUS
```

```
BATT_CURRENT
BATT_TEMP
BATT_VOLTAGE
BURN_DURATION
CAR_LOCK_STAT
CDHE_TEMP
DEMOD_LOCK_STAT
E_M_SENS_TEMP1
E_M_SENS_TEMP2
E_M_SENSOR_T1
E_M_SENSOR_T2
E_M_SENSOR_T3
E_M_SENSOR_T4
E_M_SENSOR_T5
ELEC_12VDC_VOLT
ELEC_28VDC_CURR
ELEC_5VDC_VOLT
ELEC_TEMP

--More--(Enter ‘q’ to quit)
q

Monitor mnemomic 1 (Enter ‘&l’ to list the mnemonics in the selected
telemetry, or ‘&q’ to quit):
```

Enter a mnemonic on the list, MAG_TEMP, for instance, the program will display a prompt for the next mnemonic until it has obtained the selected number of mnemonics:

```
Mnemomic 1 for postprocessing (Enter ‘&l’ to list the mnemonics in the
selected OASIS-CC output): mag_temp

Monitor mnemomic 2 (Enter ‘&l’ to list the mnemonics in the selected
telemetry, or ‘&q’ to quit):
```

The process repeats until you have entered the selected number of mnemonics. The program then asks for the time span to postprocess.

```
Would you like to enter a specific interval of the telemetry’s time span?
(The time span is [069:10:04:23:095, 069:10:14:28:654], and is the default
interval.)  Please enter ‘Y’ or ‘N’:
```

If you enter ‘N’, the program will use the default time span. If you enter ‘Y’, the program will promt for the initial and final times for the time interval to use.

```
Would you like to enter a specific interval of the telemetry’s time span?
(The time span is [069:10:04:23:095, 069:10:14:28:654], and is the default
interval.)  Please enter ‘Y’ or ‘N’: Y

Use the following format
  DDD:HH:MM:SS.SSS
where:
  DDD    = number of days (001-366)
  HH     = number of hours (00-23)
  MM     = number of minutes (00-59)
  SS.SSS = number of seconds (00.000-59.999).

Interval initial time (Enter ‘&q’ to quit): 069:10:06:00.000
Interval initial time (Enter ‘&q’ to quit): 069:10:08:00.000
```

The file has now been created with the user definedfilename, monitors, and time span. The program then prompts if you want the file printed.

```
Would you like to print the table?
Please enter ‘Y’ or ‘N’:
```

The program then asks if the user wants to plot the data from the file just created.

```
Would you like to display a plot of the telemetry subset?
Please enter ‘Y’ or ‘N’:
```

Answering yes invokes MATLAB, which is the tool used to perform the plotting. If there is another user of MATLAB at the time, the program will display the warning:

```
Warning: MATLAB has reached the maximum number of users. Please postprocess
the telemetry again, of rerun this program at a later time.
```

Otherwise, after a few moments, a plot window is diplayed on the terminal and the program asks if the users wants to save and print the plot.

```
Would you like to display a plot of the telemetry subset?
Please enter ‘Y’ or ‘N’: Y

MATLAB performs the plotting. After you create the plot, use the mouse to click
this window to return control to the window.


Would you like to save and print the plot?
Please enter ‘Y’ or ‘N’:
```

If the user answers yes, the displayed plot will disappear and the program asks whether you want to continue to postprocessing more data and repeat the process again.

```
Would you like to postprocess telemetry again?
Please enter ‘Y’ or ‘N’:
```

To quit the program properly and return to the UNIX shell prompt, enter “N” at the Yes/No prompt for creating another output file:

```
Would you like to create another output file?
Please enter ‘Y’ or ‘N’: N
```

All PostProcTlm output file are in the selected output directory. If you list the directory, f97_jan_27_09_28_27, with the UNIX command, ls -rt, you will see the files listed chronologically from top to bottom.

9.2 Merge Software

The Merge software is a two step operation that first filters the raw data then merges this filtered data. The raw FrameX generated data is filtered to take out bad telemetry frames based on the VCDU count and also engineering data packets. The output of the preprocessing software is the input to the Merge software.

9.2.1 Merge Preprocessing Software

The C application program 'preprocessmerge' was developed to filter out bad data frames and also engineering data packets. The input binary file for the program is generated by FrameX. Before running the 'preprocessmerge' software program, copy the raw files to be processed into the directory "/home/lunargrp/mergesoftware".

The 'preprocessmerge' software will prompt the user to enter the name of the file to process. The program will only accept 1 file at a time and will allow the user to quit the program if necessary .

If the user enters a file name that doesn't exist in the working directory, "/lunargrp/mergesoftware", then the program will stop executing and notify the user that the file could not be found.

The following files are generated by the 'preprocessmerge' software:

yyyy_mmm_dd_hh_nn.filter - contains good data frames which will be used by the 'mergesw' program

yyyy_mmm_dd_hh_nn.pprocess_summary - a summary report of the following:
Total Number of Frames
Number of Good Data Frames
Number of Bad Data Frames
Percent of Good Data Frames

yyyy_mmm_dd_hh_nn.vcdu_list - contains a listing of the following:
Transfer Frame ID, VCDU, VCDU Status (good or bad vcdu count)

The FrameX raw file names to be preprocessed must have the following naming convention:

```
yyyy_mmm_dd_hh_nn.raw
```

Where *yyyy* is the year, *mmm* is the month, *dd* is the day, *hh* is the hour and *nn* is the minute.

Note: when entering the filename at the program prompt, do not include the .raw extension.

To execute the program "preprocessmerge", go to the directory "/lunargrp/mergesoftware" and type the following at the command prompt:

```
% preprocessmerge
```

The program will prompt the user for input:

```
Lunar Prospector Pre-Processor Merge Software

Enter the file name to pre-process and hit the Return key.
(i.e. 1997_Sep_30_09_15)

1997_Oct_01_14_15

Please wait while raw data is being pre-processd.
```

The program will create the following files: *1997_Oct_01_14_15.filter*
1997_Oct_01_14_15.pprocess_summary
1997_Oct_01_14_15.vcdu_list

9.2.2 Merge Software

The C application program 'mergesw' merges the delay and prior science data packets, which are generated by FrameX and filtered by the preprocessing software described above, into a time ordered format and stored into a binary file. Before running the merge software program, verify that the filtered raw files to be merged are in the directory "~lunargrp/mergesoftware".

The merge software will prompt the user to enter the number of files to be merged, 1 or 2 files. The program will only accept 1 or 2 files as input and will allow the user to quit the program if necessary . If one file is chosen the merge software will prompt the user to enter the filename. This starts the data comparison from the beginning of the file to the end-of-file (EOF) - 1600 counts (VCDU). If two files are chosen the merge software will prompt the user to enter the first file name then the second file name. The merge software generates a temporary file that will append the second file to the end of the first file and write the combined data to a file called "bufferfile.filldata", this process is transparent to the user. The program will start the data comparison from the prior data at EOF1 - 1600 (EOF1 - end of file 1) to the prior data at EOF2 - 1600 (EOF2 -

end of file 2). The output file will take on the name of the first file with the extension ".merge" appended to it.

If the user enters a file name that doesn't exist in the working directory, "~lunargrp / mergesoftware", then the program will stop executing and notify the user that the file could not be found.

Note: The two files must be entered in chronological order or the program will not run since it is expecting a higher VCDU count in the second file than the first.

Note: The merge software only works for the raw science data packets.

Note: The merge software can not be run on files that are less than 1600 counts long. If files are less than 1600 counts long, two files must be appended together to make it long enough. The following Unix command can be used to create a larger file by concatenating two files together. Be sure that the output filename has the same format as defined above.

Note: The merge software will overwrite any existing file with the same name.

```
% cat filename1 filename2 > output_filename
```

The filtered raw file names to be merged must have the following naming convention:

```
yyyy_mmm_dd_hh_nn.filter
```

Where *yyyy* is the year, *mmm* is the month, *dd* is the day, *hh* is the hour and *nn* is the minute.

The output file name takes on the name of the first file compared and appends a .merge extension.

Note: When entering the filename, do not include the .filter extension. Also, the merge software will overwrite any existing file with the same name.

To execute the program "mergesw", go to the directory "~lunargrp / mergesoftware" and type the following at the command prompt:

```
% mergesw
```

The program will prompt the user for input: i.e.

```
Lunar Prospector Merge Software

Please enter the number of data files to merge ( 1 or 2):
```

```
and hit the Return key.  Press 'Ctrl-C' to quit.

2

You have requested 2 data files to be merged.
Enter the first file name and hit the Return key.
(i.e. 1997_Sep_30_09_15)

1997_Oct_01_14_15

Enter the second file name and hit the Return key.
(i.e. 1997_Sep_30_09_15)

1997_Oct_01_14_16

Creating merged science data file.
```

The program will create the output file: *1997_Oct_01_14_15.merge.*
1997_Oct_01_14_15.summary

where the .merge file contains the merged data and the .summary contains the following information: total number of frames, VCDU count range, number of prior frames chosen, number of delay frames chosen, number of frames that mismatched, and number of time when only one frame available.

When the merge is complete, any ".raw" files stored in the mergesoftware directory ("/lunar/mergesoftware") should be deleted since these files take up a lot of disk space.

Note: Delete any files with the extensions ".filldata" and ".tempmerge". These files are generated by the merge software and are removed at the end of the program. If the program is exited before its completion, these files will remain in the working directory.

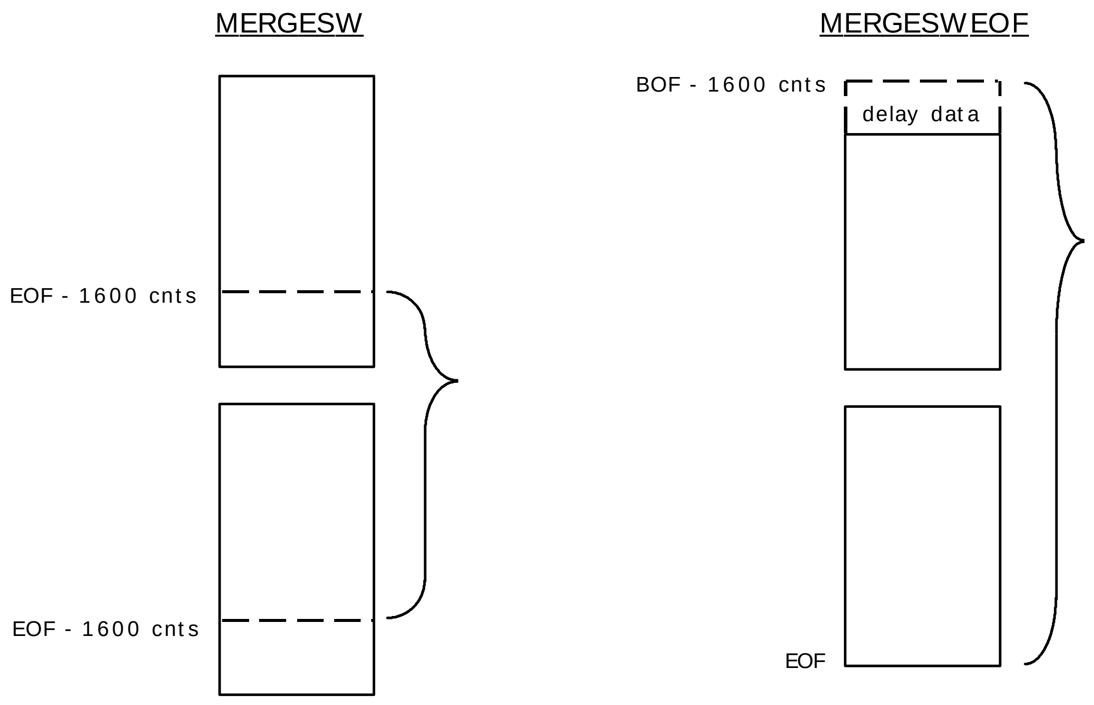
There is also a special merge software packedge called 'mergesweof' that was developed for cases when there are large data outages so that the maximum amount of data can be merged. This program prompts the user for the same information as the mergesw program. Like the 'mergesw' program, raw data files must first be run through the preprocessing program. To start this program, the following command is sent on the command line:

```
% mergesweof
```

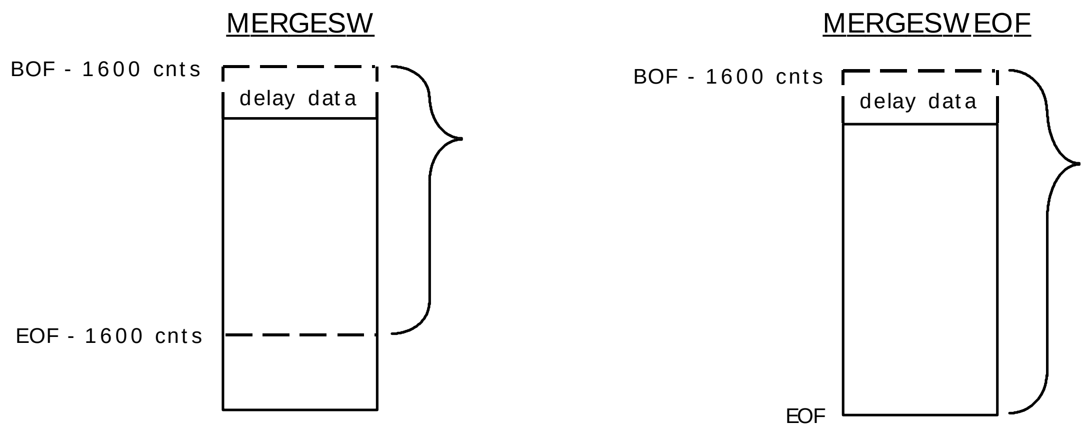
Note: Refer to the mergesw discussion above for instructions to the prompts.

Figure 9.2.2-1 shows the differences between the mergesw and mergesweof. It shows what data is being compared and output to a file.

2 file comparisons:



1 file comparisons:



BOF = beginning of file

EOF = end of file

cnts = VCDU counts

} = telemetry frames compared and written to .merge file

Figure 9.2.2-1 Merge Software Comparison Definitions

10 ATTITUDE DETERMINATION SOFTWARE (ADS)

The Lunar Prospector Attitude Determination Software (ADS) estimates the inertial orientation of the Lunar Prospector spin axis using attitude sensor telemetry and orbit ephemeris data provided by the NASA Flight Dynamics Division (FDF).

ADS does all of its calculations in the Mean-Ecliptic-of-J2000 reference frame. All inputs and outputs are expressed in this reference frame. The only exception is the NASA FDF Ephemeris Files, which are provided by FDF in the Mean-Earth-Equator-of-J2000 reference frame. ADS is aware of this, and internally transforms the ephemeris data to the Mean-Ecliptic-of-J2000 frame.

ADS expresses its attitude solutions as the right ascension and declination of the spacecraft spin axis relative to the Mean-Ecliptic-of-J2000 reference frame. Right ascension and declination are defined in Figure 10-1.

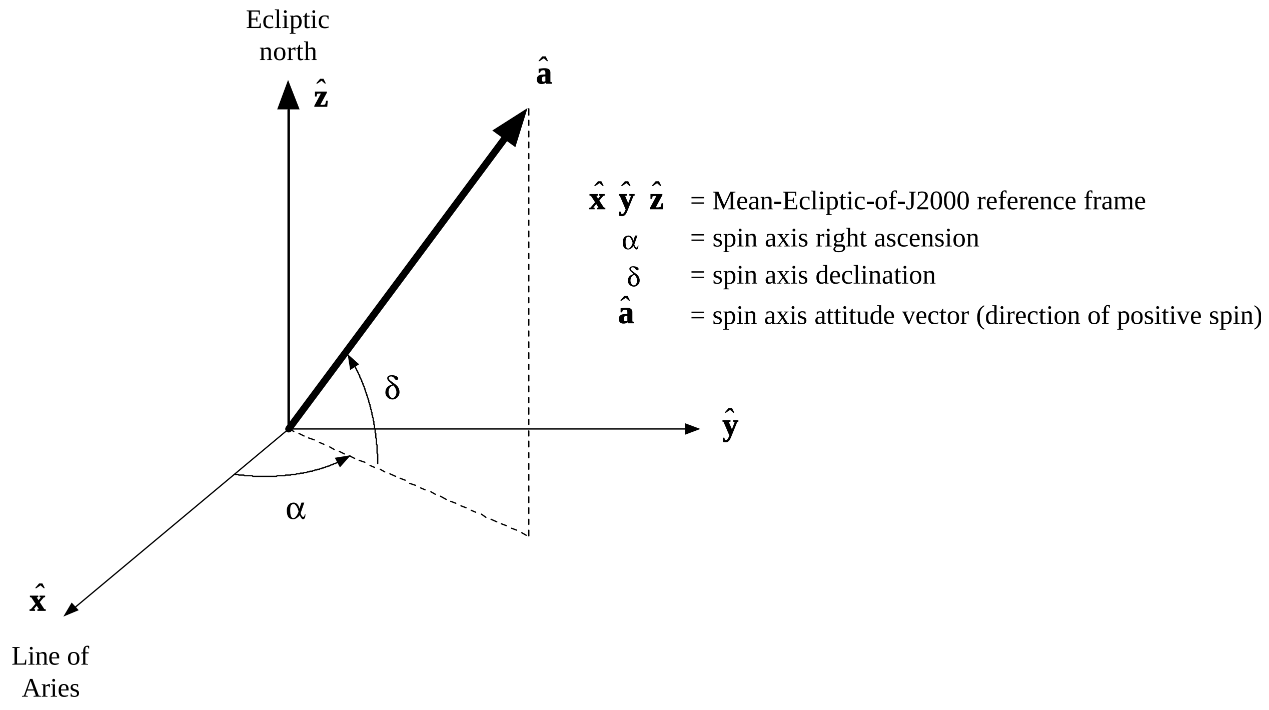

Ecliptic north
$\hat{\mathbf{z}}$
$\hat{\mathbf{a}}$
$\hat{\mathbf{x}}\ \hat{\mathbf{y}}\ \hat{\mathbf{z}}$ = Mean-Ecliptic-of-J2000 reference frame
α = spin axis right ascension
δ = spin axis declination
$\hat{\mathbf{a}}$ = spin axis attitude vector (direction of positive spin)
δ
$\hat{\mathbf{y}}$
α
$\hat{\mathbf{x}}$
Line of Aries

Figure 10-1 ADS definition of right ascension and declination. Angles are shown in the positive direction

Directory paths

The ADS executable, its source code, and its input files are all on each of the Lunar Prospector Mission Control Center (MCC) Sun workstations (feta, gouda, and gouda2) in the directory **~lunargrp/att/ads/**. The executable file is named **ads**.

ADS requires calibration of the Earth / Moon Sensor data. The calibration curves are calculated in a two-step process using the ORBGEOM and EMSSIM programs. The ORBGEOM executable, its source code, and its input files are all in the directory **~lunargrp/att/orbgeom/**. The executable file is named **orbgeom**.

EMSSIM is a Matlab script, and requires Matlab in order to execute. Matlab is currently loaded on feta and gouda. The EMSSIM source code and input files are in the directory **~lunargrp/att/ems/**. The EMSSIM script is contained in the file **emssim.m**.

EMSSIM should be used with real spacecraft telemetry only. The spacecraft Vehicle & Environments Simulation (VES) contains a simplified model of the EMS. The calibration curves for this model differ somewhat from the calibration curves for the real EMS. When calibrating EMS data generated by the VES, use the program VESCAL instead. VESCAL is also located in the directory **~/lunargrp/att/ems/**. VESCAL does not require Matlab. The executable version of VESCAL is named **vescal**.

Execution sequence

Generating an attitude solution using ADS is a multi-step process. The sequence of steps is outlined below. Details of each step are provided in later sections. Those sections are numbered to exactly match the step numbers listed in the outline.

The "Command" column of the outline makes a number of assumptions, which you will need to adjust as necessary:

- You are logged in as **lunargrp**
- You are logged in on either **feta** or **gouda** (**matlab** is not available on **gouda2**)
- OASIS is running on **gouda2**
- The attitude sensor bridge file names are **f98_jan_06_02_25_00.ss_bridge_data** and **f98_jan_06_02_25_04.em_bridge_data**
- The FDF ephemeris file is in the **~lunargrp/att** directory and is named **980106.FDFephem**

Step	Action	UNIX Command(s)
1.	**Prepare the calibration curves for the EMS data**	
1.1	Connect to the ORBGEOM directory	`cd ~/att/orbgeom`

1.2	Verify that the correct ephemeris file is in use	`diff LP_ephem.dat ~/att/980106.FDFephem`
		`cp ~/att/980106.FDFephem LP_ephem.dat`

1.3	Update the attitude file to the current best estimate of the attitude	`textedit LP_attitude.dat`
1.4	Generate the EMS scan geometry	`orbgeom`

1.5	Generate the EMS edge calibration curves	`cp LP_scans.dat ~/att/ems`
		`cd ~/att/ems`

1.5.1 or	Calibration curves for flight data or	`matlab` `emssim`
1.5.1	Calibration curves for VES data	`vescal`
1.6	Copy the EMS calibration curves to ADS	`cp LP_emscal.dat ~/att/ads`
2.	**Fetch the raw attitude telemetry**	
2.1	Connect to the ADS directory	`cd ~/att/ads`

2.2	ftp the bridge files from the OASIS bridge file directory	`ftp gouda2`
		`cd ~/oasis_ops1.0/oasis/bridge`
		`ls *bridge_data`
		`get f98_jan_06_02_25_00.ss_bridge_data`
		`get f98_jan_06_02_25_04.em_bridge_data`
		`quit`

3.	**Generate an attitude solution (ADS)**	

3.1	Verify that the correct ephemeris file is in use	`diff LP_ephem.dat ~/att/980106.FDFephem`
		`cp ~/att/980106.FDFephem LP_ephem.dat`

3.2	Update sensor cant angles for current stowed/deployed configuration	`textedit LP_sensor.dat`
3.3	Verify that the correct attitude bridge files are available	`ls *bridge_data`
3.4	Run ADS	`ads`

3.5	View the attitude unit sphere plots	`cd ~/att/ads`
		`matlab`
		`adsplot`

3.6	If estimated attitude differs significantly from attitude used in step 1.3, start again at step 1.3	

Step 1. Prepare the calibration curves for the EMS data

This section describes in more detail the sequence of steps required to generate a calibration curve for the EMS raw data. The EMS produces a time tag when its boresight crosses the horizon from space to a body (the “leading edge”) and when its boresight crosses the horizon from a body to space (the “trailing edge”). Because the sensor has a finite field of view and internal processing delays, the time tag does not correspond exactly to the instant of the edge crossing. The calibration curves are analytic predictions of the difference between the sensor time tag and the actual edge crossing time. ADS adjusts the EMS time tags using these calibration curves.

Preparation of the calibration curves is a two-step process. First, the program ORBGEOM calculates the EMS ideal scan geometry for a user-specified series of points in the spacecraft orbit. The output of ORBGEOM is the ideal chord width and the location of each scan on the body (Earth or Moon) relative to the Sun. This Sun-based geometry is important for Moon scans, since the EMS response is determined by lunar surface temperature and the lunar surface temperature is strongly dependent on the lighting conditions.

Second, the program EMSSIM simulates the EMS performance for each scan path generated by ORBGEOM. By comparing the EMS response to the ideal scan generated by ORBGEOM, EMSSIM can determine the EMS leading and trailing edge biases.

EMSSIM is intended for real spacecraft telemetry only. The spacecraft Vehicle & Environments Simulation (VES) contains a simplified model of the EMS. The calibration curves for this model differ somewhat from the calibration curves for the real EMS. When calibrating EMS data generated by the VES, the program VESCAL should be used instead.

ORBGEOM requires the following input files.

File name	Description / usage
`LP_attitude.dat`	Spacecraft attitude file. Contains current best estimate of spacecraft spin rate and spin axis attitude (right ascension and declination relative to the ecliptic mean-of-J2000 reference frame). This file must be updated manually each time a new definitive attitude solution is calculated.
`LP_ephem.dat`	Spacecraft ephemeris file. Provides vectors from the Earth or the Moon to the spacecraft as a function of time. This file is updated regularly by FDF, and should be replaced each time a new update is available. Updates are downloaded from the FDF website at http://fdd.gsfc.nasa.gov/lp
`mn2000.dat`	NASA SLP file. Provides vectors to Earth, Moon, Sun, and other planets. This file should never be changed, unless instructed to do so by the Flight Dynamics Facility (FDF) at GSFC. It is write-protected in the ORBGEOM directory.

A sample **LP_attitude.dat** file is shown below. Note the "last update" field is for information only – it is not used by ORBGEOM. The spin axis right ascension and declination should be updated manually when a new attitude solution is found. The spin rate should be updated each time the spacecraft spin rate changes.

```
LUNAR PROSPECTOR ATTITUDE DATA FILE

last update                           yymmdd.hhmmss  =   971014.163705
current spin axis right ascension     deg            =      165.8
current spin axis declination         deg            =       65.2
current spin rate                     rpm            =       12.0
```

The first 100 lines from a sample **LP_ephem.dat** file are shown below. Most of these lines are header information. Significant information contained in the header includes the Ephemeris start and end time, which define the time span of the file, and the central body indicator (1=Earth, 2=Moon). ORBGEOM scans are generated for the central body indicated in this file.

The remainder of the file is a series of ephemeris records, each of which defines the spacecraft position and velocity relative to the central body specified in the header information. The time of each record is relative to the ephemeris start time specified in the header. The position and velocity vectors are expressed in the Mean-Earth-Equator-of-J2000 reference frame.

```
EPHEMERIS FILE INFORMATION:
---------------------------

SATELLITE ID NUMBER =   9753321
RUN TITLE =     ### LUNARP TRANSFER ORBIT EPHEM 1/6 NOM ###
TAPE IDENTIFIER = GTDS
EPHEMERIS START TIME (YYMMDD.)    =    980106.0000000000000000
EPHEMERIS START TIME (SEC OF DAY) =      8940.0000000000000000
EPHEMERIS END TIME (YYMMDD.)      =    980111.0000000000000000
EPHEMERIS END TIME (SEC OF DAY)   =         0.0000000000000000
EPHEMERIS DELTA-T (SECONDS)       =        60.0000000000000000
ORBIT THEORY (COWELL OR BROUWER)  = COWELL
INTEGRATION STEP (1=FIXED)        =    1
COORDINATE SYSTEM           = 2000
COORD. SYSTEM INDICATOR     =    4
YYMMDD OF FILE CREATION     = 971008.0
HHMMSS.SSS OF FILE CREATION = 132739.000


INITIAL CONDITION INFORMATION:
------------------------------

ELEMENT EPOCH (MM/DD/YY  HH:MM:SS.SSS) =   1/ 6/98   2:28:42.291
INITIAL CARTESIAN ELEMENTS :    X (KM) =      -463.0825262583412
                                Y (KM) =      -6197.904842591940
                                Z (KM) =      -2110.754093584153
                           VX (KM/SEC) =       9.951585550850749
                           VY (KM/SEC) =      -2.122371374669885
                           VZ (KM/SEC) =       3.988412510298945
INITIAL KEPLERIAN ELEMENTS :    A (KM) =       198523.7094335387
                                    E  =      0.9669369442056979
                               I (DEG) =       29.13246735468717
                            RAAN (DEG) =       303.2696001558697
                              AP (DEG) =       318.4516507696209
                              MA (DEG) =      0.8865339687475564E-03
INITIAL BROUWER MEAN ELEMENTS : A (KM) =       194468.9397066015
                                    E  =      0.9662335459424093
                               I (DEG) =       29.11307479084779
                            RAAN (DEG) =       303.2518380092671
                              AP (DEG) =       318.4684316771654
                              MA (DEG) =      0.1060559178629347E-02
CENTRAL BODY INDICATOR (1=EARTH)          =     1.0
DRAG PERTURBATION (0=YES,1=NO)            =     1.0
SOLAR RADIATION PERTURBATION (0=YES,1=NO) =     0.0
SUN POINT MASS PERTURBATION (0=YES,1=NO)  =     0.0
MOON POINT MASS PERTURBATION (0=YES,1=NO) =     0.0
SPACECRAFT AREA (M**2)          =       2.130000000000000
SPACECRAFT MASS (KG)            =       297.0000000000000
DRAG COEFFICIENT                =       2.000000000000000
SOLAR REFLECTIVITY COEFFICIENT  =       1.400000000000000
ATMOSPHERIC DENSITY MODEL       =
RHO1  =      0.0000000000000000E+00
RHO2  =      0.0000000000000000E+00
RHO3  =      0.0000000000000000E+00
RHO4  =      0.5235987755983001


OTHER INFORMATION:
------------------

ECCENTRIC ANOMALY AT EPOCH (DEG)     =      0.2681340248465708E-01
TRUE ANOMALY AT EPOCH (DEG)          =      0.2068117394754810
ANOMALISTIC PERIOD (MINUTES)         =       14671.62784867214
```

```
 GREENWICH HOUR ANGLE AT EPOCH (RAD)  =       1.839110096929809
 INITIAL GREENWICH HOUR ANGLE (RAD)   =       2.491025222803789
 FINAL GREENWICH HOUR ANGLE (RAD)     =       1.925124534793656
 GEOCENTRIC SUN POSITION AT EPOCH (KM):  X  =       39544090.76390587
                                         Y  =      -129993211.3346548
                                         Z  =      -56360194.68166067
 YYMMDD OF EARLIEST MEASUREMENT       =       0.0
 HHMMSS.SSS OF EARLIEST MEASUREMENT   =       0.000
 YYMMDD OF LATEST MEASUREMENT         =       0.0
 HHMMSS.SSS OF LATEST MEASUREMENT     =       0.000
 LEAP SECOND DURING EPHEM (2=YES)     =     1


 EPHEMERIS:
 ----------

TIME FROM EPOCH (SEC)        X (KM)                Y (KM)               Z (KM)              XDOT (KM/S)           YDOT (KM/S)           ZDOT (KM/S)
--------------------- --------------------- --------------------- --------------------- --------------------- --------------------- ---------------------
 0.00000000000000E+00 -0.28676044294472E+03 -0.62341165370081E+04 -0.20396602674799E+04  0.99609523824127E+01 -0.19671342816779E+01  0.40404020120186E+01
 0.60000000000000E+02  0.31111918746346E+03 -0.63362791118643E+04 -0.17922662417349E+04  0.99599865355601E+01 -0.14379089649970E+01  0.42022617937429E+01
 0.12000000000000E+03  0.90745144529139E+03 -0.64067242449952E+04 -0.15358681481695E+04  0.99097716394690E+01 -0.91159403338423E+00  0.43401989267519E+01
 0.18000000000000E+03  0.14993744931452E+04 -0.64459252718346E+04 -0.12719495697883E+04  0.98137518541732E+01 -0.39797741528094E+00  0.44528180832016E+01
 0.24000000000000E+03  0.20842891828004E+04 -0.64549057870559E+04 -0.10020401129043E+04  0.96771472687344E+01  0.94488196677957E-01  0.45399678731955E+01
 0.30000000000000E+03  0.26599478051279E+04 -0.64351438817087E+04 -0.72764439050619E+03  0.95063407040818E+01  0.55916247433890E+00  0.46025980319717E+01
 0.36000000000000E+03  0.32245039176859E+04 -0.63884560346184E+04 -0.45018299477298E+03  0.93082236250613E+01  0.99141449837932E+00  0.46425142768240E+01
 0.42000000000000E+03  0.37765253369424E+04 -0.63168774339661E+04 -0.17094967377679E+03  0.90896249674936E+01  0.13885590733826E+01  0.46620904395443E+01
 0.48000000000000E+03  0.43149765440886E+04 -0.62225516258316E+04  0.10891434394552E+03  0.88568815402677E+01  0.17496156141164E+01  0.46639928248554E+01
 0.54000000000000E+03  0.48391803511315E+04 -0.61076382879208E+04  0.38843063520664E+03  0.86155695980961E+01  0.20749688290606E+01  0.46509497501182E+01
 0.60000000000000E+03  0.53487679624781E+04 -0.59742420841656E+04  0.66678198682122E+03  0.83703809417398E+01  0.23660083426079E+01  0.46255830022660E+01
 0.66000000000000E+03  0.58436249470398E+04 -0.58243625976870E+04  0.94330243327427E+03  0.81251082011260E+01  0.26247952040889E+01  0.45902988990147E+01
 0.72000000000000E+03  0.63238388121784E+04 -0.56598627700573E+04  0.12174623618567E+04  0.78827039767812E+01  0.28537821024314E+01  0.45472302602690E+01
 0.78000000000000E+03  0.67896514517603E+04 -0.54824521993749E+04  0.14888513581376E+04  0.76453803084546E+01  0.30555967053031E+01  0.44982166718519E+01
 0.84000000000000E+03  0.72414181377436E+04 -0.52936816368572E+04  0.17571606886763E+04  0.74147241060175E+01  0.32328846476534E+01  0.44448105071316E+01
 0.90000000000000E+03  0.76795735036484E+04 -0.50949453174031E+04  0.20221666598822E+04  0.71918124449772E+01  0.33882035565067E+01  0.43882985012273E+01
 0.96000000000000E+03  0.81046041877948E+04 -0.48874883843569E+04  0.22837155382147E+04  0.69773182649010E+01  0.35239573495143E+01  0.43297311607365E+01
 0.10200000000000E+04  0.85170274115501E+04 -0.46724173126741E+04  0.25417103331486E+04  0.67716020870908E+01  0.36423604167320E+01  0.42699547667397E+01
```

Step 1.1 Connect to the orbgeom directory

ORBGEOM is contained in the directory **~lunargrp/att/orbgeom/**. Move to this directory by typing the UNIX command

```
cd ~/att/orbgeom
```

Step 1.2 Verify that the correct ephemeris file is in use

A copy of the file **LP_ephem.dat** should be in the ORBGEOM directory. This file should define the orbit ephemeris for the time interval during which you wish to calculate an attitude solution. Verify that it matches the FDF ephemeris file by typing the UNIX command

```
diff LP_ephem.dat ~/att/980106.FDFephem
```

(assuming that the FDF ephemeris file is named 980106.FDFephem). If UNIX tells you the files are different, update the ORBGEOM copy by typing the UNIX command

```
cp ~/att/980106.FDFephem LP_ephem.dat
```

Step 1.3 Update the attitude file to the current best estimate of the attitude

To calculate the EMS scan geometry, ORBGEOM needs an estimate of the spacecraft attitude. Using your favorite UNIX text editor, update the right ascension and declination in the file **LP_attitude.dat** to match the best estimate of the attitude prior to running ADS. If you don’t have a favorite editor, this UNIX command will work:

```
textedit LP_attitude.dat
```

If no best estimate is available, use the nominal attitude for this point in the mission. If after running ADS you find that the attitude you entered into **LP_attitude.dat** is way off, then come back to this point, enter the new estimate into **LP_attitude.dat**, and redo the attitude determination process.

Step 1.4 Generate the EMS scan geometry

Run ORBGEOM by typing the command

orbgeom

The interaction between you and ORBGEOM will appear as shown below, with your inputs highlighted in bold.

The start and end times you enter should completely encompass the time range of the attitude telemetry you will be using in ADS (see Step 3.4 section). ADS will ignore any EMS telemetry that falls outside the range you specify here.

The time step is somewhat arbitrary – ADS will linearly interpolate between time steps on the EMS calibration curves. When near the Earth or the Moon, the time step should be no more than a few minutes, since the EMS scan geometry changes relatively rapidly. During transfer between the Earth and the Moon the time step can be much longer, since the scan geometry changes very slowly.

The EMS cant angle is the angle between the EMS boresight and the major principal axis of the spacecraft. Because the principal axis is a function of the spacecraft mass properties, this angle changes during boom deployments.

Spacecraft configuration	**EMS cant angle (deg)**
All booms stowed	95.058
MAG boomlet deployed, all others stowed	90.904
All booms deployed	90.035

```
feta% orbgeom


                      LUNAR PROSPECTOR
               Orbit Geometry Utility Software
                        Version 1.1

         Copyright 1997 Lockheed Martin Corporation
                   All rights reserved


          -----------  LP_attitude.dat  ----------
          Spin axis right ascension =   165.80 deg
          Spin axis declination     =    65.20 deg
          ----------------------------------------


1. Create orbit geometry plots
2. Generate input vectors for EMS calibration
3. Quit

Enter action: 2
Start time (yymmdd.hhmmss UTC) : 980106.052000
End   time (yymmdd.hhmmss UTC) : 980106.063000
Time step (sec)                : 180
EMS cant angle (deg)           : 90.035


 EMS calibration body = 1  (1=Earth, 2=Moon)
 EMS cant angle                =    90.035000 deg
 S/C attitude right ascension  =   165.800000 deg
 S/C attitude declination      =    65.200000 deg
 S/C attitude vector           =    -0.406636    0.102895    0.907777
 Unit vector from S/C to Earth =    -0.640049   -0.740942   -0.203328
 Unit vector from S/C to Moon  =     0.933881    0.343957   -0.097776
 Unit vector from S/C to Sun   =     0.270595   -0.962693   -0.000077
 Distance from S/C to Earth =        49887.806 km
 Distance from S/C to Moon  =       328984.052 km
 Distance from S/C to Sun   =    147127162.472 km
 Subsatellite point                     Subsolar reference frame           Subsolar reference frame
   Sun         Sun       EMS    S/C position vector wrt central body       S/C attitude vector              Dihedral
Angle
Longitude  Latitude   Chord         x           y           z           x          y          z          LE
midscan     TE
  (deg)      (deg)    (deg)       (km)        (km)        (km)                                          (deg)
(deg)     (deg)
--------- ---------  -------   ----------  ----------  ----------   ---------  ---------  ---------  --------  ------
--  --------
  123.463    11.731   14.783   -26933.464   40749.155   10143.375   -0.209198  -0.363565   0.907776   296.072
303.463   310.855
  123.739    11.621   14.601   -27476.271   41138.634   10173.799   -0.209211  -0.363557   0.907776   296.438
303.739   311.039
  124.008    11.513   14.425   -28016.294   41524.029   10203.204   -0.209224  -0.363549   0.907776   296.795
304.008   311.220
  124.270    11.408   14.254   -28553.580   41905.450   10231.621   -0.209238  -0.363542   0.907776   297.143
304.270   311.397
  124.526    11.304   14.087   -29088.175   42283.001   10259.084   -0.209251  -0.363534   0.907776   297.482
304.526   311.569
  124.775    11.203   13.926   -29620.121   42656.782   10285.622   -0.209265  -0.363526   0.907776   297.812
304.775   311.738
  125.019    11.104   13.769   -30149.461   43026.890   10311.265   -0.209278  -0.363518   0.907776   298.135
305.019   311.904
  125.258    11.007   13.617   -30676.237   43393.417   10336.040   -0.209292  -0.363511   0.907776   298.449
305.258   312.066
  125.491    10.911   13.469   -31200.490   43756.450   10359.973   -0.209305  -0.363503   0.907776   298.756
305.491   312.225
  125.719    10.818   13.324   -31722.257   44116.076   10383.090   -0.209319  -0.363495   0.907776   299.056
305.719   312.381
  125.941    10.726   13.184   -32241.578   44472.375   10405.414   -0.209332  -0.363487   0.907776   299.349
305.941   312.533
  126.159    10.637   13.047   -32758.490   44825.426   10426.970   -0.209346  -0.363480   0.907776   299.636
306.159   312.683
  126.373    10.548   12.914   -33273.029   45175.303   10447.778   -0.209359  -0.363472   0.907776   299.916
306.373   312.830
  126.582    10.462   12.784   -33785.231   45522.081   10467.860   -0.209373  -0.363464   0.907776   300.190
306.582   312.974
  126.786    10.377   12.658   -34295.129   45865.827   10487.236   -0.209386  -0.363456   0.907776   300.458
306.786   313.115
  126.987    10.294   12.534   -34802.758   46206.610   10505.925   -0.209400  -0.363449   0.907776   300.720
306.987   313.254
  127.184    10.212   12.414   -35308.150   46544.495   10523.947   -0.209413  -0.363441   0.907776   300.977
307.184   313.390
  127.376    10.131   12.296   -35811.337   46879.543   10541.319   -0.209427  -0.363433   0.907776   301.228
307.376   313.524
```

```
  127.565    10.052   12.181   -36312.350   47211.816   10558.057   -0.209440  -0.363425   0.907776   301.475
307.565   313.656
  127.751     9.974   12.069   -36811.220   47541.371   10574.180   -0.209453  -0.363417   0.907776   301.716
307.751   313.785
  127.932     9.898   11.959   -37307.976   47868.265   10589.702   -0.209467  -0.363410   0.907776   301.953
307.932   313.912
  128.111     9.823   11.852   -37802.648   48192.553   10604.640   -0.209480  -0.363402   0.907776   302.185
308.111   314.037
  128.286     9.749   11.748   -38295.263   48514.285   10619.006   -0.209494  -0.363394   0.907776   302.412
308.286   314.160
  128.458     9.676   11.645   -38785.849   48833.514   10632.817   -0.209507  -0.363386   0.907776   302.636
308.458   314.281
```

ORBGEOM outputs a file called **LP_scans.dat**. The **LP_scans.dat** for the above sample run of ORBGEOM is shown below. The contents of this file are essentially identical to the tabular output ORBGEOM displays to the user on the screen.

The first line of this file is a header record. The first column on this line defines the central body being scanned by the EMS (Earth=1, Moon=2). The second column is the EMS cant angle. All remaining columns on this line are always zero.

Each line after the first describes a single scan of the EMS across the central body. The contents of each column are defined in the following table. The "subsolar reference frame" is a reference frame with its origin at the center of the central body, its +x axis parallel to the line from the central body to the sun, its +z axis parallel to the ecliptic normal, and its +y axis completing the right-hand coordinate triad.

Column	Contents
1	Scan time (Julian day)
2 - 3	Sun longitude and sun latitude of subsatellite point (Both are 0.0 when sun is directly overhead) (deg)
4	Ideal EMS chord width (deg)
5 - 7	Spacecraft position vector (x,y,z), expressed in the subsolar reference frame (km)
8 - 10	Spacecraft attitude vector (from LP_attitude.dat), expressed in the subsolar reference frame
11 - 13	Dihedral angle from the sun crossing to the ideal EMS leading edge, midscan point, and trailing edge (deg)

It would be worthwhile at this point to jot down the range of EMS chord widths and midscan dihedral angles (columns 4 and 12), as these will be helpful when running ADS (see Step 3.4 section).

```
1.000000000000     90.035     0.000    0.000         0.000        0.000        0.000    0.000000   0.000000   0.000000     0.000     0.000     0.000
2450819.722222223878    123.463    11.731   14.783   -26933.464   40749.155   10143.375   -0.209198  -0.363565   0.907776   296.072   303.463   310.855
2450819.724305557087    123.739    11.621   14.601   -27476.271   41138.634   10173.799   -0.209211  -0.363557   0.907776   296.438   303.739   311.039
2450819.726388890762    124.008    11.513   14.425   -28016.294   41524.029   10203.204   -0.209224  -0.363549   0.907776   296.795   304.008   311.220
2450819.728472223971    124.270    11.408   14.254   -28553.580   41905.450   10231.621   -0.209238  -0.363542   0.907776   297.143   304.270   311.397
2450819.730555557180    124.526    11.304   14.087   -29088.175   42283.001   10259.084   -0.209251  -0.363534   0.907776   297.482   304.526   311.569
```

```
2450819.732638890389   124.775   11.203   13.926   -29620.121   42656.782   10285.622   -0.209265  -0.363526   0.907776   297.812   304.775   311.738
2450819.734722224064   125.019   11.104   13.769   -30149.461   43026.890   10311.265   -0.209278  -0.363518   0.907776   298.135   305.019   311.904
2450819.736805557273   125.258   11.007   13.617   -30676.237   43393.417   10336.040   -0.209292  -0.363511   0.907776   298.449   305.258   312.066
2450819.738888890482   125.491   10.911   13.469   -31200.490   43756.450   10359.973   -0.209305  -0.363503   0.907776   298.756   305.491   312.225
2450819.740972223692   125.719   10.818   13.324   -31722.257   44116.076   10383.090   -0.209319  -0.363495   0.907776   299.056   305.719   312.381
2450819.743055557366   125.941   10.726   13.184   -32241.578   44472.375   10405.414   -0.209332  -0.363487   0.907776   299.349   305.941   312.533
2450819.745138890576   126.159   10.637   13.047   -32758.490   44825.426   10426.970   -0.209346  -0.363480   0.907776   299.636   306.159   312.683
2450819.747222223785   126.373   10.548   12.914   -33273.029   45175.303   10447.778   -0.209359  -0.363472   0.907776   299.916   306.373   312.830
2450819.749305556994   126.582   10.462   12.784   -33785.231   45522.081   10467.860   -0.209373  -0.363464   0.907776   300.190   306.582   312.974
2450819.751388890669   126.786   10.377   12.658   -34295.129   45865.827   10487.236   -0.209386  -0.363456   0.907776   300.458   306.786   313.115
2450819.753472223878   126.987   10.294   12.534   -34802.758   46206.610   10505.925   -0.209400  -0.363449   0.907776   300.720   306.987   313.254
2450819.755555557087   127.184   10.212   12.414   -35308.150   46544.495   10523.947   -0.209413  -0.363441   0.907776   300.977   307.184   313.390
2450819.757638890762   127.376   10.131   12.296   -35811.337   46879.543   10541.319   -0.209427  -0.363433   0.907776   301.228   307.376   313.524
2450819.759722223971   127.565   10.052   12.181   -36312.350   47211.816   10558.057   -0.209440  -0.363425   0.907776   301.475   307.565   313.656
2450819.761805557180   127.751    9.974   12.069   -36811.220   47541.371   10574.180   -0.209453  -0.363417   0.907776   301.716   307.751   313.785
2450819.763888890389   127.932    9.898   11.959   -37307.976   47868.265   10589.702   -0.209467  -0.363410   0.907776   301.953   307.932   313.912
2450819.765972224064   128.111    9.823   11.852   -37802.648   48192.553   10604.640   -0.209480  -0.363402   0.907776   302.185   308.111   314.037
2450819.768055557273   128.286    9.749   11.748   -38295.263   48514.285   10619.006   -0.209494  -0.363394   0.907776   302.412   308.286   314.160
2450819.770138890482   128.458    9.676   11.645   -38785.849   48833.514   10632.817   -0.209507  -0.363386   0.907776   302.636   308.458   314.281
```

Step 1.5 Generate the EMS edge calibration curves

The EMS calibration curve generation program, EMSSIM, requires the following input files.

File name	Description / usage
`LP_scans.dat`	EMS boresight scan file. Contains data describing the path followed by the EMS boresight as it scans across either the Earth or the Moon. This file is output by ORBGEOM, and is an input to EMSSIM (or VESCAL).

EMSSIM is located in the directory **~lunargrp/att/ems**. Copy **LP_scans.dat** to this directory by typing the UNIX command

```
cp LP_scans.dat ~/att/ems
```

Then switch to this directory by typing the UNIX command

```
cd ~/att/ems
```

EMSSIM is a Matlab script. Start Matlab by typing the command

```
matlab
```

When you receive the Matlab prompt (>>), type the command

```
emssim
```

EMSSIM will prompt you for the spacecraft spin rate (in rpm) and the EMS voltage threshold (in volts). The spin rate is the value of the spacecraft telemetry mnemonic EMSPIN during the time interval for which you will be running ADS (see Step 3.4 section). The voltage threshold is the value of the telemetry mnemonic EMSTHR during this same interval.

When EMSSIM is finished, you will receive the Matlab prompt (>>) again. Type the command

```
quit
```

This sequence of commands and the computer response is shown below. User input is highlighted in bold.

```
feta% cp LP_scans.dat ~/att/ems
feta% cd ~/att/ems
feta% matlab

                            < M A T L A B (R) >
 (c) Copyright 1984-94 The MathWorks, Inc.
                            All Rights Reserved
                               Version 4.2c
                                Dec 31 1994

Commands to get started: intro, demo, help help
Commands for more information: help, whatsnew, info, subscribe

>> emssim

 Lunar Prospector Earth-Moon Sensor performance simulation
                   Version 2.1

          Daniel Swanson    408-756-4738

    Copyright 1997 Lockheed Martin Corporation
               All rights reserved


 Spin rate (rpm): 12

 C&DH EMS voltage threshold (V): 0.2

 Sun longitude, latitude =  123.463,   11.731  deg
 Sun longitude, latitude =  123.739,   11.621  deg
 Sun longitude, latitude =  124.008,   11.513  deg
 Sun longitude, latitude =  124.270,   11.408  deg
 Sun longitude, latitude =  124.526,   11.304  deg
 Sun longitude, latitude =  124.775,   11.203  deg
 Sun longitude, latitude =  125.019,   11.104  deg
 Sun longitude, latitude =  125.258,   11.007  deg
 Sun longitude, latitude =  125.491,   10.911  deg
 Sun longitude, latitude =  125.719,   10.818  deg
 Sun longitude, latitude =  125.941,   10.726  deg
 Sun longitude, latitude =  126.159,   10.637  deg
 Sun longitude, latitude =  126.373,   10.548  deg
 Sun longitude, latitude =  126.582,   10.462  deg
 Sun longitude, latitude =  126.786,   10.377  deg
 Sun longitude, latitude =  126.987,   10.294  deg
 Sun longitude, latitude =  127.184,   10.212  deg
 Sun longitude, latitude =  127.376,   10.131  deg
 Sun longitude, latitude =  127.565,   10.052  deg
 Sun longitude, latitude =  127.751,    9.974  deg
 Sun longitude, latitude =  127.932,    9.898  deg
 Sun longitude, latitude =  128.111,    9.823  deg
 Sun longitude, latitude =  128.286,    9.749  deg
 Sun longitude, latitude =  128.458,    9.676  deg

>> quit

 15137000 flops.

feta%
```

EMSSIM outputs a file called **LP_emscal.dat**. This file contains the EMS edge calibration curves. This file for the above run of EMSSIM is shown below. **LP_emscal.dat** defines the calibration that ADS will apply to the EMS telemetry. Each line of this file describes a single scan of the EMS across the central body. The contents of each column of this file are defined in the following table. ADS interpolates between entries in this file when the actual time of a scan does not match a time from column 1.

Column	Contents
1	Scan time (Julian day)
2	EMS leading edge bias (sec). Negative if EMS detects the leading edge before the boresight crosses the true leading edge; positive if EMS detects the leading edge after the true leading edge.
3	EMS trailing edge bias (sec). Negative if EMS detects the trailing edge before the boresight crosses the true trailing edge; positive if EMS detects the trailing edge after the true trailing edge.
4	Ideal EMS chord width (deg)
5	Spin period (sec) for the spin rate specified in LP_attitude.dat

```
2.4508197222222239e+06   1.2844262722227939e-02   1.2187348251597330e-01   1.4760000000000005e+01   5.0000000000000000e+00
2.4508197243055571e+06   1.3126473696110308e-02   1.2158040471995091e-01   1.4616000000000005e+01   5.0000000000000000e+00
2.4508197263888908e+06   1.3373257831395646e-02   1.2132978780840509e-01   1.4472000000000005e+01   5.0000000000000000e+00
2.4508197284722240e+06   1.2581103170775876e-02   1.2211657244104335e-01   1.4184000000000005e+01   5.0000000000000000e+00
2.4508197305555572e+06   1.2750838414318344e-02   1.2193660929381767e-01   1.4040000000000004e+01   5.0000000000000000e+00
2.4508197326388904e+06   1.2888845576920049e-02   1.2178901473960912e-01   1.3896000000000004e+01   5.0000000000000000e+00
2.4508197347222241e+06   1.2995931627658441e-02   1.2167291649605394e-01   1.3752000000000004e+01   5.0000000000000000e+00
2.4508197368055573e+06   1.3072999783312111e-02   1.2158736792489033e-01   1.3608000000000004e+01   5.0000000000000000e+00
2.4508197388888905e+06   1.3121031844089970e-02   1.2153136761387107e-01   1.3464000000000004e+01   5.0000000000000000e+00
2.4508197409722237e+06   1.3141075262826596e-02   1.2150387061223089e-01   1.3320000000000004e+01   5.0000000000000000e+00
2.4508197430555574e+06   1.3134225601274752e-02   1.2150380334832145e-01   1.3176000000000004e+01   5.0000000000000000e+00
2.4508197451388906e+06   1.3101618864596709e-02   1.2153006834915436e-01   1.3032000000000004e+01   5.0000000000000000e+00
2.4508197472222238e+06   1.3044419514321337e-02   1.2158155308921414e-01   1.2888000000000003e+01   5.0000000000000000e+00
2.4508197493055570e+06   1.2963825964922848e-02   1.2165712084651759e-01   1.2744000000000003e+01   5.0000000000000000e+00
2.4508197513888907e+06   1.2861050417358966e-02   1.2175562600965373e-01   1.2600000000000003e+01   5.0000000000000000e+00
2.4508197534722239e+06   1.3737328720623054e-02   1.2087590635038126e-01   1.2600000000000003e+01   5.0000000000000000e+00
2.4508197555555571e+06   1.3593717039154685e-02   1.2101697736057093e-01   1.2456000000000003e+01   5.0000000000000000e+00
2.4508197576388908e+06   1.3427898602996979e-02   1.2117802219523766e-01   1.2312000000000003e+01   5.0000000000000000e+00
2.4508197597222240e+06   1.3239371125501220e-02   1.2135773221133195e-01   1.2168000000000003e+01   5.0000000000000000e+00
2.4508197618055572e+06   1.3033827476271131e-02   1.2155691480632436e-01   1.2024000000000003e+01   5.0000000000000000e+00
2.4508197638888904e+06   1.3812574945114897e-02   1.2077814034893830e-01   1.2024000000000003e+01   5.0000000000000000e+00
2.4508197659722241e+06   1.3576330777506085e-02   1.2101554362685651e-01   1.1880000000000003e+01   5.0000000000000000e+00
2.4508197680555573e+06   1.3317987189331748e-02   1.2126827175729593e-01   1.1736000000000002e+01   5.0000000000000000e+00
2.4508197701388905e+06   1.3043167348518514e-02   1.2153612251396206e-01   1.1592000000000002e+01   5.0000000000000000e+00
```

If the EMS attitude telemetry was generated by the Vehicle and Environments Simulator (VES) instead of by the EMS hardware, then VESCAL must be used instead of EMSSIM. The sequence to run VESCAL is the same as is used to run EMSSIM, except that VESCAL is run instead of entering Matlab and running EMSSIM.

```
feta% cp LP_scans.dat ../   ~/att/ems
feta% cd ~/att/ems
feta% vescal
Enter current spin rate (rpm) :                   12
Enter current voltage threshold (EMSTHR) value : 0.2
feta%
```

VESCAL uses produces an output file **LP_emscal.dat** with a format identical to that produced by EMSSIM.

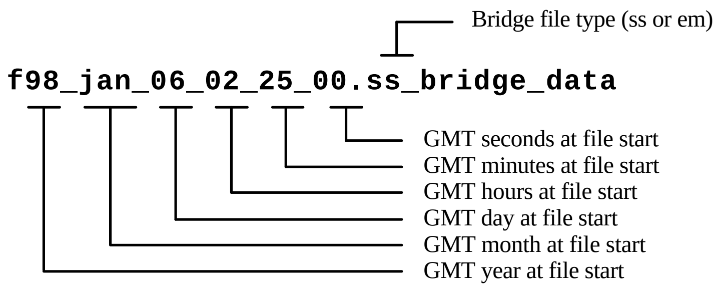
Step 1.6 Copy the EMS calibration curves to ADS

Copy the file **LP_emscal.dat** to the ADS directory by typing the UNIX command

cp LP_emscal.dat ~/att/ads

Step 2. Fetch the raw attitude telemetry

OASIS stores the attitude sensor telemetry in two files called bridge files. The bridge file directory is on the MCC Sun Workstation running OASIS. The Flight Controllers will know which one this is. For the purposes of this document, we will assume that the bridge files are on **gouda2**. The bridge files are in the directory **~lunargrp/oasis_ops1.0/oasis/bridge**. The bridge files will have a name of the form


Bridge file type (ss or em)
f98_jan_06_02_25_00.ss_bridge_data
GMT seconds at file start
GMT minutes at file start
GMT hours at file start
GMT day at file start
GMT month at file start
GMT year at file start

The contents of each column of the sun sensor bridge file are defined in the following table.

Column	Contents
1	Data quality indicator
2	Ground receipt day-of-year (GMT)
3	Ground receipt time-of-day, expressed in milliseconds (GMT)
4	Telemetry monitor SSTIM (sec)
5	Telemetry monitor SSSINV (volts)
6	Telemetry monitor SSCOSV (volts)
7	Telemetry monitor SSBIASV (volts)
8	Telemetry monitor SSANG (hex gray code)
9	Telemetry derived monitor SEA (deg)
10	Telemetry derived monitor ASPIN (sec)
11	Telemetry monitor TANKPRESS (psia)

The contents of each column of the Earth / Moon sensor bridge file are defined in the following table.

Column	Contents
1	Data quality indicator
2	Ground receipt day-of-year (GMT)
3	Ground receipt time-of-day, expressed in milliseconds (GMT)
4	Telemetry monitor EMTIM1 (sec)
5	Telemetry monitor EMTIM2 (sec)
6	Telemetry monitor EMTIM3 (sec)

7	Telemetry monitor EMTIM4 (sec)
8	Telemetry monitor EMTIM5 (sec)
9	Telemetry monitor EMTIM6 (sec)
10	Telemetry monitor EMTHRESHLEV (6 2-bit fields, expressed as a single hex number)

A sample sun sensor bridge file is shown below.

```
0   6  19740000   0.356789E+02   0.000000E+00   0.000000E+00   0.000000E+00    0    0.000000E+00  0.500000E+01   0.450000E+03
0   6  19742000   0.356789E+02   0.000000E+00   0.000000E+00   0.000000E+00    0    0.379400E+00  0.500000E+01   0.450000E+03
0   6  19744000   0.161556E+01   0.175781E+00   0.279297E+01   0.000000E+00    0    0.379400E+00  0.500000E+01   0.450000E+03
0   6  19746000   0.161556E+01   0.175781E+00   0.279297E+01   0.250000E+01   2A  -0.120807E+02  0.500000E+01   0.450000E+03
0   6  19748000   0.661556E+01   0.175781E+00   0.273438E+01   0.250000E+01   2A  -0.120807E+02  0.500000E+01   0.450000E+03
0   6  19750000   0.661556E+01   0.175781E+00   0.273438E+01   0.250000E+01   2A  -0.120886E+02  0.500000E+01   0.450000E+03
0   6  19752000   0.116156E+02   0.175781E+00   0.267578E+01   0.250000E+01   2A  -0.120886E+02  0.500000E+01   0.450000E+03
0   6  19754000   0.116156E+02   0.175781E+00   0.267578E+01   0.250000E+01   2A  -0.120966E+02  0.500000E+01   0.450000E+03
0   6  19756000   0.356789E+02   0.000000E+00   0.000000E+00   0.250000E+01   2A  -0.120966E+02  0.500000E+01   0.450000E+03
0   6  19758000   0.356789E+02   0.000000E+00   0.000000E+00   0.000000E+00    0  -0.117172E+02  0.500000E+01   0.450000E+03
0   6  19760000   0.166156E+02   0.175781E+00   0.277344E+01   0.000000E+00    0  -0.117172E+02  0.500000E+01   0.450000E+03
0   6  19762000   0.166156E+02   0.175781E+00   0.277344E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19764000   0.216156E+02   0.175781E+00   0.275391E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19766000   0.216156E+02   0.175781E+00   0.275391E+01   0.250000E+01   2A  -0.120860E+02  0.500000E+01   0.450000E+03
0   6  19768000   0.266156E+02   0.175781E+00   0.267578E+01   0.250000E+01   2A  -0.120860E+02  0.500000E+01   0.450000E+03
0   6  19770000   0.266156E+02   0.175781E+00   0.267578E+01   0.250000E+01   2A  -0.120966E+02  0.500000E+01   0.450000E+03
0   6  19772000   0.316156E+02   0.175781E+00   0.273438E+01   0.250000E+01   2A  -0.120966E+02  0.500000E+01   0.450000E+03
0   6  19774000   0.316156E+02   0.175781E+00   0.273438E+01   0.250000E+01   2A  -0.120886E+02  0.500000E+01   0.450000E+03
0   6  19776000   0.356789E+02   0.000000E+00   0.000000E+00   0.250000E+01   2A  -0.120886E+02  0.500000E+01   0.450000E+03
0   6  19778000   0.356789E+02   0.000000E+00   0.000000E+00   0.000000E+00    0  -0.117092E+02  0.500000E+01   0.450000E+03
0   6  19780000   0.461556E+01   0.175781E+00   0.277344E+01   0.000000E+00    0  -0.117092E+02  0.500000E+01   0.450000E+03
0   6  19782000   0.461556E+01   0.175781E+00   0.277344E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19784000   0.961556E+01   0.175781E+00   0.269531E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19786000   0.961556E+01   0.175781E+00   0.269531E+01   0.250000E+01   2A  -0.120939E+02  0.500000E+01   0.450000E+03
0   6  19788000   0.146156E+02   0.175781E+00   0.271484E+01   0.250000E+01   2A  -0.120939E+02  0.500000E+01   0.450000E+03
0   6  19790000   0.146156E+02   0.175781E+00   0.271484E+01   0.250000E+01   2A  -0.120913E+02  0.500000E+01   0.450000E+03
0   6  19792000   0.196156E+02   0.175781E+00   0.277344E+01   0.250000E+01   2A  -0.120913E+02  0.500000E+01   0.450000E+03
0   6  19794000   0.196156E+02   0.175781E+00   0.277344E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19796000   0.356789E+02   0.000000E+00   0.000000E+00   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19798000   0.356789E+02   0.000000E+00   0.000000E+00   0.000000E+00    0  -0.117039E+02  0.500000E+01   0.450000E+03
0   6  19800000   0.246156E+02   0.175781E+00   0.271484E+01   0.000000E+00    0  -0.117039E+02  0.500000E+01   0.450000E+03
0   6  19802000   0.246156E+02   0.175781E+00   0.271484E+01   0.250000E+01   2A  -0.120913E+02  0.500000E+01   0.450000E+03
0   6  19804000   0.296156E+02   0.175781E+00   0.269531E+01   0.250000E+01   2A  -0.120913E+02  0.500000E+01   0.450000E+03
0   6  19806000   0.296156E+02   0.175781E+00   0.269531E+01   0.250000E+01   2A  -0.120939E+02  0.500000E+01   0.450000E+03
0   6  19808000   0.261556E+01   0.175781E+00   0.277344E+01   0.250000E+01   2A  -0.120939E+02  0.500000E+01   0.450000E+03
0   6  19810000   0.261556E+01   0.175781E+00   0.277344E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19812000   0.761556E+01   0.175781E+00   0.275391E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19814000   0.761556E+01   0.175781E+00   0.275391E+01   0.250000E+01   2A  -0.120860E+02  0.500000E+01   0.450000E+03
0   6  19816000   0.356789E+02   0.000000E+00   0.000000E+00   0.250000E+01   2A  -0.120860E+02  0.500000E+01   0.450000E+03
0   6  19818000   0.356789E+02   0.000000E+00   0.000000E+00   0.000000E+00    0  -0.117066E+02  0.500000E+01   0.450000E+03
0   6  19820000   0.126156E+02   0.175781E+00   0.269531E+01   0.000000E+00    0  -0.117066E+02  0.500000E+01   0.450000E+03
0   6  19822000   0.126156E+02   0.175781E+00   0.269531E+01   0.250000E+01   2A  -0.120939E+02  0.500000E+01   0.450000E+03
0   6  19824000   0.176156E+02   0.175781E+00   0.275391E+01   0.250000E+01   2A  -0.120939E+02  0.500000E+01   0.450000E+03
0   6  19826000   0.176156E+02   0.175781E+00   0.275391E+01   0.250000E+01   2A  -0.120860E+02  0.500000E+01   0.450000E+03
0   6  19828000   0.226156E+02   0.175781E+00   0.277344E+01   0.250000E+01   2A  -0.120860E+02  0.500000E+01   0.450000E+03
0   6  19830000   0.226156E+02   0.175781E+00   0.277344E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19832000   0.276156E+02   0.175781E+00   0.269531E+01   0.250000E+01   2A  -0.120833E+02  0.500000E+01   0.450000E+03
0   6  19834000   0.276156E+02   0.175781E+00   0.269531E+01   0.250000E+01   2A  -0.120939E+02  0.500000E+01   0.450000E+03
```

```
0   6  19836000   0.356789E+02    0.000000E+00    0.000000E+00    0.250000E+01   2A  -0.120939E+02   0.500000E+01    0.450000E+03
```

A sample Earth / Moon sensor bridge file is shown below.

```
0   6  19740000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19742000   0.675556E+00   0.731667E+00   0.992222E+00   0.104778E+01   0.356789E+02   0.356789E+02   B90
0   6  19744000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19746000   0.567556E+01   0.573167E+01   0.599222E+01   0.604778E+01   0.356789E+02   0.356789E+02   B90
0   6  19748000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19750000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19752000   0.106756E+02   0.107317E+02   0.109922E+02   0.110478E+02   0.356789E+02   0.356789E+02   B90
0   6  19754000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19756000   0.156756E+02   0.157317E+02   0.159922E+02   0.160478E+02   0.356789E+02   0.356789E+02   B90
0   6  19758000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19760000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19762000   0.206756E+02   0.207317E+02   0.209922E+02   0.210494E+02   0.356789E+02   0.356789E+02   B90
0   6  19764000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19766000   0.256756E+02   0.257317E+02   0.259922E+02   0.260494E+02   0.356789E+02   0.356789E+02   B90
0   6  19768000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19770000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19772000   0.306756E+02   0.307317E+02   0.309922E+02   0.310494E+02   0.356789E+02   0.356789E+02   B90
0   6  19774000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19776000   0.367556E+01   0.373222E+01   0.399222E+01   0.404944E+01   0.356789E+02   0.356789E+02   B90
0   6  19778000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19780000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19782000   0.867556E+01   0.873278E+01   0.899222E+01   0.904944E+01   0.356789E+02   0.356789E+02   B90
0   6  19784000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19786000   0.136756E+02   0.137328E+02   0.139922E+02   0.140494E+02   0.356789E+02   0.356789E+02   B90
0   6  19788000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19790000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19792000   0.186756E+02   0.187328E+02   0.189922E+02   0.190494E+02   0.356789E+02   0.356789E+02   B90
0   6  19794000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19796000   0.236756E+02   0.237328E+02   0.239922E+02   0.240494E+02   0.356789E+02   0.356789E+02   B90
0   6  19798000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19800000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19802000   0.286756E+02   0.287333E+02   0.289922E+02   0.290494E+02   0.356789E+02   0.356789E+02   B90
0   6  19804000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19806000   0.167556E+01   0.173333E+01   0.199222E+01   0.204944E+01   0.356789E+02   0.356789E+02   B90
0   6  19808000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19810000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19812000   0.667556E+01   0.673389E+01   0.699278E+01   0.704944E+01   0.356789E+02   0.356789E+02   B90
0   6  19814000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19816000   0.116756E+02   0.117339E+02   0.119928E+02   0.120494E+02   0.356789E+02   0.356789E+02   B90
0   6  19818000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19820000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19822000   0.166756E+02   0.167344E+02   0.169928E+02   0.170494E+02   0.356789E+02   0.356789E+02   B90
0   6  19824000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19826000   0.216756E+02   0.217350E+02   0.219928E+02   0.220494E+02   0.356789E+02   0.356789E+02   B90
0   6  19828000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19830000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
0   6  19832000   0.266756E+02   0.267350E+02   0.269928E+02   0.270494E+02   0.356789E+02   0.356789E+02   B90
0   6  19834000   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02   0.356789E+02     0
```

```
0   6  19836000   0.316761E+02    0.317361E+02    0.319928E+02    0.494444E-01    0.356789E+02    0.356789E+02    B90
```

Step 2.1 Connect to the ADS directory

ADS is located in the directory **~lunargrp/att/ads**. Connect to the ADS directory by typing the UNIX command

cd ~/att/ads

Step 2.2 ftp the bridge files from the OASIS bridge file directory

We will assume here that OASIS is running on the MCC Sun workstation **gouda2**. ftp to this by typing the UNIX command

ftp gouda2

Log in to **gouda2** using the user name **lunargrp** and the password **lunar123**.

Connect to the bridge file directory on **gouda2** by typing the ftp command

cd ~/oasis_ops1.0/oasis/bridge

List all bridge files contained in the bridge file directory by typing the ftp command

ls *bridge_data

Many bridge files may be shown. Identify the sun sensor bridge file and Earth/Moon sensor bridge file you want to use. For example, assume that the bridge files you want are **f98_jan_06_02_25_00.ss_bridge_data** and **f98_jan_06_02_25_04_em_bridge_data**. Copy these files to the ADS directory by typing the ftp commands

get f98_jan_06_02_25_00.ss_bridge_data
get f98_jan_06_02_25_04.em_bridge_data

Finally, quit out of ftp by typing the ftp command

quit

A complete interactive ftp session is shown below. User inputs are highlighted in bold.

```
feta% ftp gouda2
Connected to gouda2.
220 gouda2 FTP server (UNIX(r) System V Release 4.0) ready.
Name (gouda2:dswanson): lunargrp
331 Password required for lunargrp.
Password: lunar123
```

```
230 User lunargrp logged in.
ftp> cd ~/oasis_ops1.0/oasis/bridge
250 CWD command successful.
ftp> ls *bridge_data
200 PORT command successful.
150 ASCII data connection for /bin/ls (137.228.119.12,35838).
f98_jan_06_02_25_00.ss_bridge_data
f98_jan_06_02_25_04.em_bridge_data
226 ASCII Transfer complete.
remote: *data
72 bytes received in 0.017 seconds (4.2 Kbytes/s)
ftp> get f98_jan_06_02_25_00.ss_bridge_data
200 PORT command successful.
150 ASCII data connection for f98_jan_06_02_25_00.ss_bridge_data
(137.228.119.12,35839).
226 ASCII Transfer complete.
local: f98_jan_06_02_25_00.ss_bridge_data remote: f98_jan_06_02_25_00.ss_bridge_data
1025100 bytes received in 1.1 seconds (9.2e+02 Kbytes/s)
ftp> get f98_jan_06_02_25_04.em_bridge_data
200 PORT command successful.
150 ASCII data connection for f98_jan_06_02_25_04.em_bridge_data
(137.228.119.12,35839).
226 ASCII Transfer complete.
local: f98_jan_06_02_25_04.em_bridge_data remote: f98_jan_06_02_25_04.em_bridge_data
1025100 bytes received in 1.1 seconds (9.2e+02 Kbytes/s)
ftp> quit
221 Goodbye.
feta%
```

Step 3. Generate an attitude solution (ADS)

This section describes in more detail the sequence of steps required to generate an attitude solution using ADS. ADS performs a number of functions to produce an attitude solution.

1. ADS reads in the raw attitude telemetry from the attitude bridge files.
2. ADS allows the user to select a specific time interval within the telemetry data.
3. If the telemetry was generated when the spin rate was above 15 rpm, ADS compensates for the attitude data buffering that occurs in the C&DH.
4. ADS filters the telemetry to remove spurious pulses (i.e., the sun pulse, or the Earth pulse when in orbit around the Moon, or the Moon pulse during trans-lunar operations)
5. ADS sorts the filtered attitude data into groups. Each group consists of one sun pulse and one subsequent EMS pulse.
6. ADS adjusts the data for known sensor alignment errors.
7. ADS corrects the EMS data using the EMS calibration curves generated by EMSSIM.
8. ADS calculates the attitude using one of five different, user selectable, solution techniques.
9. ADS generates a celestial sphere plot showing the attitude solution.

The attitude solution techniques are described in detail in EM AVS-004. They will not be described in detail here.

ADS requires the following input files.

File name	**Description / usage**
`LP_ephem.dat`	Spacecraft ephemeris file. Provides vectors from the Earth or the Moon to the spacecraft as a function of time. This file is updated regularly by FDF, and should be replaced each time a new update is available. Updates are downloaded from the FDF website at http://fdd.gsfc.nasa.gov/lp
`mn2000.dat`	NASA SLP file. Provides vectors to Earth, Moon, Sun, and other planets. This file should never be changed, unless instructed to do so by the Flight Dynamics Facility (FDF) at GSFC. It is write-protected in the ORBGEOM directory.
`LP_emscal.dat`	EMS calibration data. Provides bias adjustments for space-to-horizon (leading edge) and horizon-to-space (trailing edge) transitions. This file is generated using the ORBGEOM and EMSSIM (or VESCAL) utility programs. It should be recreated each time a new set of attitude data is to be processed by ADS.
`LP_sensor.dat`	Attitude sensor description file. Contains current best estimate of a number attitude sensor parameters. This file is created prior to launch, and may be updated manually if flight data indicates parameters differ from pre-flight estimates. It must also be updated following boom deployment to accommodate the shift in the principal axis orientation, which changes the angles between the spin axis and the attitude sensors.
OASIS attitude data bridge files	Sun Sensor and EMS bridge files. Generated by OASIS. These files contain the spacecraft attitude telemetry. The bridge file names are generated based on the GMT

	clock time when the file is started. Example file names: `f98_jan_06_02_25_00.ss_bridge_data` `f98_jan_06_02_25_04.em_bridge_data`

The files **LP_ephem.dat** and **mn2000.dat** are described in the Step 1 section. The file **LP_emscal.dat** is described in the Step 1.5 section. The attitude data bridge files are described in the Step 2 section.

A sample **LP_sensor.dat** file is shown below. Note that lines that must be changed following boom deployment are commented. To change the file, remove the asterisk from the start of the line describing the current state of the spacecraft, and insert an asterisk at the start of the line describing the prior state of the spacecraft.

The values shown for the Sun Sensor and Earth / Moon sensor angles are derived from I&T measurements, vendor boresight measurements, and principal axis estimates based on final mass properties measurements. These values are used to calibrate the sensor data, so it is important that they be as accurate as possible.

All uncertainty parameters are "best-guess" estimates. They are used by ADS to estimate the uncertainty of the attitude solution. Exact knowledge of these parameters is not required to generate an attitude solution.

```
*********************************************************************
*                                                                   *
*              The ADS attitude sensor parameter file               *
*                                                                   *
*********************************************************************
*
*
***********************************************
* ADS version this input file is intended for *
***********************************************
*
>   2.00   < ADS version
*
*
***************************
*  Sun sensor parameters  *
***************************
*
* SS cant angle (Angle from spin axis to sun sensor boresight)
>   94.8676   < deg (all booms stowed)
*>  90.8386   < deg (only MAG boomlet deployed)
*>  89.9934   < deg (all booms deployed)
*
* SS azimuth angle (Angle from +x-axis to sun sensor boresight)
> 1.2361         < deg
*
* Sun angle measurement uncertainty, 1-sigma  (estimated)
>    1.0      < deg (all booms stowed)
*>   0.2      < deg (only MAG boomlet deployed)
*>   0.02     < deg (all booms deployed)
*
```

```
* CEP time measurement uncertainty, 1-sigma  (estimated)
> 0.0004      < sec
*
*
**********************************
*  Earth/Moon sensor parameters  *
**********************************
*
* EMS cant angle (Angle from spin axis to EMS boresight)
>   95.0579   < deg (all booms stowed)
*>  90.9044   < deg (only MAG boomlet deployed)
*>  90.0351   < deg (all booms deployed)
*
* EMS azimuth angle (Angle from +x-axis to EMS boresight)
> -0.103         < deg
*
* EMS cant angle uncertainty, 1-sigma (estimated)
>   1.0       < deg (all booms stowed)
*>  0.2       < deg (only MAG boomlet deployed)
*>  0.02      < deg (all booms deployed)
*
* EMS leading edge uncertainty, 1-sigma (estimated)
> 0.5        < deg
*
* EMS trailing edge uncertainty, 1-sigma (estimated)
> 0.5        < deg
*
*
**************************************
*  Ephemeris uncertainty parameters  *
**************************************
*
* Ephemeris file circular position uncertainty, 1-sigma (from FDF)
> 10.0        < km
```

ADS generates the following output files.

File name	**Description / usage**
`echoxxx.ads`	File containing a playback of the complete ADS session. This file provides a permanent record of the processing of the attitude data. The “xxx” field in the file name is a 3-digit field, which is increment each time ADS is executed.
`cones.out,` `grid.out,` `points.out,` `ref.out,` `sem.out,` `method.out`	Files containing the plot data for the attitude solution celestial sphere.
`DEBUG_SStags,` `DEBUG_EMtags`	History of the filtering applied to the attitude sensor data. These are of interest only to the curious.
`DEBUG_groups`	ADS creates groups of attitude data. Each group consists of one sun sensor time, one EMS leading edge time, and one EMS trailing edge time. This file lists all the attitude sensor groups after filtering of the raw telemetry is complete. These groups are the starting point for calculating attitude solutions.

The only file the user should ever have to look at is **echoxxx.ads**. The format of this file should be self-evident, since it is identical to what is displayed on the screen during an interactive session with ADS.

Step 3.1 Verify that the correct ephemeris file is in use

A copy of the file **LP_ephem.dat** should be in the ADS directory. This file should define the orbit ephemeris for the time interval during which you wish to calculate an attitude solution. Verify that it matches the FDF ephemeris file by typing the UNIX command

```
diff LP_ephem.dat ~/att/980106.FDFephem
```

(assuming that the FDF ephemeris file is named 980106.FDFephem). If UNIX tells you the files are different, update the ADS copy by typing the UNIX command

```
cp ~/att/980106.FDFephem LP_ephem.dat
```

Step 3.2 Update sensor cant angles for current stowed/deployed configuration

To calibrate the attitude sensor data, ADS needs an estimate of the angle between the spacecraft principal axis and the boresight of each of the attitude sensors. Using your favorite UNIX text editor, update the SS cant angle, the EMS cant angle, the sun angle measurement uncertainty, and the EMS cant angle uncertainty in the file **LP_sensor.dat**. If you don’t have a favorite editor, this UNIX command will work:

```
textedit LP_attitude.dat
```

See Step 3 section for more details on modifying this file.

Step 3.3 Verify that the correct attitude bridge files are available

List all bridge files contained in the ADS directory by typing the UNIX command

```
ls *bridge_data
```

Make sure that the bridge files you transferred from the OASIS bridge file directory in step 2.2 are shown. If not, repeat Steps 2.1 and 2.2.

Step 3.4 Run ADS

Run ADS by typing the command

```
ads
```

The interaction between you and ADS will appear as shown below, with your inputs highlighted in bold. Stepping through the session, note the following items of interest:

- ADS gives you the name of the echo file at the beginning of the run. In this example the echo file name is **echo018.ads**.

- ADS displays the complete contents of the attitude sensor data file **LP_sensor.dat**. Review this to verify that it is correct for the current spacecraft configuration.

- ADS tells you the time span covered by the data in the attitude sensor bridge files. It then lets you specify the start and end time of the range of data you wish to use. This is useful if you have a bridge file that has multiple events in it, such as boom deployments or thruster maneuvers.

  EXTREMELY IMPORTANT: ADS expects that the attitude data in the range you specify here is a contiguous block of like data. This means that the range should all be at essentially the same spin rate, that no boom deployments or thruster maneuvers should occur in the middle of the range, and that the EMS voltage threshold (EMSTHR) is the same throughout the range. ADS also expects that the EMS calibration curves you generated in Step 1 are for the spin rate of the range.

- ADS next lets you choose whether you want to generate only a sun-only solution. If you answer "yes" to this question, ADS will do no further processing of the EMS telemetry. It skips the next four questions about EMS pulse widths and dihedral angles, and will only offer you the sun-only solution technique later on. This is useful if you have a stretch of telemetry where nothing is in the field-of-view of the EMS. In this case ADS would otherwise abort because it can find no EMS pulses to group with the sun pulses.

- ADS next lets you specify the minimum and maximum acceptable EMS pulse width. The nominal EMS pulse width is the EMS chord width calculated by ORBGEOM in Step 1.5. Near the Earth or Moon this chord width will typically be quite large. The sun may also pass through the EMS field-of-view, but it will have a small pulse width, on the order of 10°. By setting the minimum pulse width higher than the sun width, ADS can filter out the sun pulses. Similarly, the Earth can be filter out when near the Moon, and the Moon can be filter out near the Earth. Note that this technique cannot be used in trans-lunar space, since the

Earth, Moon, and Sun pulse widths are all about the same size. The dihedral angle calculations (discussed below) will also filter out unwanted bodies.

The minimum pulse width has a second use in Lunar Mapping orbit. The EMS calibration curves become highly nonlinear just before the EMS loses the Moon from its field-of-view, and just after it reacquires the Moon back into its field-of-view. By setting the minimum pulse width to about 120°, the telemetry from the nonlinear regions of the calibration curve will be ignored.

The ORBGEOM output gives the nominal EMS chord width. This chord is calculated assuming ideal sensor performance and perfect knowledge of the attitude. When using ADS it is recommended that the user allow margin for uncertainties. In the sample ADS run shown below, ORBGEOM claims that the ideal chord width decreases from 14.8° at the start of the range to 11.7° at the end of the range (see Step 1.5 section). The EMS tends to bias the chord widths high. The EMS pulse width range used in this example is 10° to 30°.

- ADS next lets you specify the nominal dihedral angle, and the acceptable tolerance (plus or minus) on this angle. The dihedral angle is the spacecraft rotation angle from the instant the +x-axis crosses the sun line to the instant the +x-axis crosses the midpoint of the Earth or Moon. ADS discards any EMS pulses that do not fall within the specified dihedral angle range. The ORBGEOM output gives the nominal dihedral angle, which in this example increases from 303.5° at the start to 314.3° at the end (see Step 1.5 section). Again it is recommended that the user allow margin for uncertainties. The dihedral angle range used in this example is 300° ± 30°.

- ADS next shows statistics from the telemetry filtering and grouping process. Sun sensor time tags are filtered first, then Earth / Moon sensor time tags. For the sun sensor filtering:

  – “Too early” and “too late”: time tags that fall outside the user-specified time range.
  – “Main frame merge”: SSTIM is telemetered every 2 seconds, but updated only every 4 seconds. Every other time tag is redundant
  – “<no data>”: If no sun pulse occurs in the last 4 second window, an SSTIM value of hex FADE is telemetered. These are filtered out.
  – “invalid ASPIN”: sun pulses linked with obviously incorrect spin periods (<0.75 sec or >15 sec) are filtered out.
  – “Duplicate”: If two consecutive time tags have the same value, one is filtered out.

- "Failing spin period test": The time between a sun pulse and the prior sun pulse should be an integer multiple of the spin period. If it's not, the sun pulse is filtered out.
- "before first EMS pulse": ADS will use only one EMS pulse before the first EMS pulse. All others are filtered out.

For the Earth / Moon sensor filtering:

- "Too early" and "too late": time tags that fall outside the user-specified time range.
- "<no data>": If no Earth / Moon sensor pulse occurs in the last 2 second window, an EMTIM value of hex FADE is telemetered. These are filtered out.
- "Before first sun pulse": ADS generates attitude data into groups consisting of a sun pulse and a subsequent EMS pulse. EMS pulses occurring before the first sun pulse can never be put into such a group, so they are filtered out.
- "pulses before first sun pulse": should always be zero, since EMS pulses before first sun pulse were filtered out earlier. This is an artifact left over from an earlier version of ADS.
- "pulses after last sun pulse": the last sun pulse can be grouped with only one EMS pulse. All other EMS pulses after the last sun pulse are filtered out.
- "Narrow pulses" and "Fat pulses": EMS pulses shorter or longer than the user-specified EMS minimum and maximum chord are filtered out
- "pulses eliminated during grouping": Number of sun sensor and EMS pulses that survived the previous filters but didn't get grouped into a valid SS / EMS pair. Reasons for this include failing the user-specified dihedral angle conditions, multiple sun pulses between consecutive EMS pulses, or multiple EMS pulses between consecutive sun pulses.

• ADS next calibrates the sun sensor and EMS data using the azimuth bias and cant angle data from the **LP_sensor.dat** file, and the EMS calibration curves from the **LP_emscal.dat** file. This is transparent to the user, except for a statement about the number of data groups occurring before or after the **LP_emscal.dat** time range. These data groups are discarded. If any groups are discarded, a larger time range should have been specified when ORBGEOM was run (Step 1.5).

• Finally, ADS allows the user to pick an attitude determination technique. For each technique ADS will calculate two attitude estimates. These are output in the form of a right ascension and declination. One of the solutions is the true solution, the other is a "ghost" or "phantom" solution. It is up to the user to determine which is which. This is usually quite simple, since one solution will be near the expected solution and the other will be far from it. If it is not easy to determine, try different techniques to see one of the solutions from each technique falls near the same point. This is the correct solution.

ADS also calculates an analytic uncertainty (variance) for the right ascension and declination based on the uncertainties specified in the file **LP_sensor.dat**. The quality of an attitude solution is determined by the geometry for that solution, and at a given point in the orbit the geometry for one solution technique may be better than for another. The analytic variance provides a way to gauge the relative quality of the solution produced by two different techniques.

ADS also calculates a numeric variance. ADS generates an attitude solution for each of the attitude data groups that survived the filtering process. The numeric variance is the spread in these solutions. It provides another technique for assessing the quality of the solution.

It is recommended that the user try each of the five attitude solution techniques, then select the solution with the smallest analytic variance. If multiple solutions have similar variances, then it may be worthwhile to select the average of the solutions. This is at the discretion of the user.

- After producing an attitude solution, ADS prompts for a “view vector right ascension and declination”. This is the direction from which the attitude solution unit sphere (described in Step 3.5 section) is viewed. For best viewing, pick a right ascension and declination near the attitude solution.

  It is recommended that after each of the five attitude solution techniques the user plot the attitude solution unit sphere for that technique, as described in the Step 3.5 section. This provides a convenient graphical technique for assessing the quality of each of the solutions.

- When the user quits from ADS, ADS will output an error message:

```
Note: IEEE floating-point exception flags raised:
    Inexact;  Invalid Operation;
 See the Numerical Computation Guide, ieee_flags(3M)
```

  Ignore this message. ADS deliberately executes an operation that produces a “Not-a-Number” (NaN) when generating plot data for the unit sphere. This operation produces the above error message.

```
feta% ls *bridge_data
f98_jan_06_02_29_00.em_bridge_data  f98_jan_06_05_29_00.em_bridge_data
f98_jan_06_02_29_00.ss_bridge_data  f98_jan_06_05_29_00.ss_bridge_data
feta% ads


                     LUNAR PROSPECTOR
              Attitude Determination Software
                       Version 2.0

         Copyright 1997 Lockheed Martin Corporation
                    All rights reserved


A complete record of this session is saved in echo018.ads





-----------------  Attitude Sensor data file contents  -----------------

   Sun sensor cant    angle                           :    89.9934 deg
   Sun sensor azimuth angle                           :     1.2361 deg

   EMS cant    angle                                  :    90.0351 deg
   EMS azimuth angle                                  :    -0.1030 deg

   Sun angle measurement uncertainty (1-sigma)        :     0.0200 deg
   CEP measurement time uncertainty  (1-sigma)        :   0.000400 sec
   EMS cant angle uncertainty (1-sigma)               :     0.0200 deg
   EMS leading edge uncertainty (1-sigma)             :     0.5000 deg
   EMS trailing edge uncertainty (1-sigma)            :     0.5000 deg

   Ephemeris circular position uncertainty (1-sigma) :    10.0000 km

------------------------------------------------------------------------


 Enter sun sensor bridge file name: f98_jan_06_05_29_00.ss_bridge_data
 Enter EM  sensor bridge file name: f98_jan_06_05_29_00.em_bridge_data

 Attitude telemetry start time (yymmdd.hhmmss UTC) = 980106.052900
 Attitude telemetry end   time (yymmdd.hhmmss UTC) = 980106.061400

 Enter start time for attitude solution processing (yymmdd.hhmmss UTC):
980106.053500
 Enter end   time for attitude solution processing (yymmdd.hhmmss UTC):
980106.060500

 Do you want only a sun-only solution (y/n)? n

 Enter minimum acceptable EMS pulse width (deg) : 10
 Enter maximum acceptable EMS pulse width (deg) : 30

 Enter nominal dihedral angle (sun-to-nadir deg): 300
 Acceptable tolerance on dihedral angle (deg)   : 30


-----------------  Attitude Data Grouping Statistics  -----------------

   Sun Sensor time tags read from input file:               1351
   Sun Sensor time tags eliminated because too early:        180
   Sun Sensor time tags eliminated because too late:           270
   Sun Sensor time tags eliminated by main frame merge:        451
   Sun Sensor time tags that are <no data>:                     90
   Sun Sensor time tags linked with invalid ASPIN value:         0
   Duplicate Sun Sensor time tags eliminated:                    0
   Sun Sensor time tags failing spin period test:                0
   Sun Sensor time tags remaining after filtering:             360

   Earth/Moon Sensor time tags read from input file:          8106
   Earth/Moon Sensor time tags eliminated because too early:  1080
   Earth/Moon Sensor time tags eliminated because too late:   1620
   Earth/Moon Sensor time tags that are <no data>:            3966
   Earth/Moon time tags before first sun pulse:                  0
   EMS time tags remaining after filtering:                   1440

   Sun sensor time tags before first EMS pulse:                  0
   Sun sensor time tags remaining after 2nd filtering:         360

   Total Earth/Moon Sensor pulses identified:                  359
   Earth/Moon Sensor pulses before first sun pulse:              0
   Earth/Moon Sensor pulses after last sun pulse:                0
   Narrow Earth/Moon Sensor pulses eliminated:                   0
   Fat    Earth/Moon Sensor pulses eliminated:                   0
   Earth/Moon Sensor pulses remaining after filtering:         359

   Sun Sensor        pulses eliminated during grouping:          1
   Earth/Moon Sensor pulses eliminated during grouping:          0
   Number of valid SS/EMS pulse groups:                        359

----------------------------------------------------------------------



   Data groups that occur before start of LP_emscal.dat:         0
   Data groups that occur after end of LP_emscal.dat:            0

   Raw      Attitude Data Groups written to file DEBUG_groups
   Filtered Attitude Data Groups written to file DEBUG_groups
```

```
               ***  Main Menu  ***

 1. Calculate attitude using Sun-Midscan       (SDM) method
 2. Calculate attitude using Sun-Chordwidth    (SDW) method
 3. Calculate attitude using Sun-Leading Edge  (SDI) method
 4. Calculate attitude using Sun-Trailing Edge (SDO) method
 5. Calculate attitude using Sun-Only          (SS)  method
 6. Quit

 Enter selection: 1


************************
****** SDM Method ******
************************


*** Attitude estimate 1  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  165.747 deg
       variance (analytic) =    0.546 deg (1-sigma)
       variance (numeric)  =    0.015 deg (1-sigma)

    Declination            =   65.225 deg
       variance (analytic) =    1.473 deg (1-sigma)
       variance (numeric)  =    0.012 deg (1-sigma)


*** Attitude estimate 2  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =   40.616 deg
       variance (analytic) =    0.748 deg (1-sigma)
       variance (numeric)  =    0.017 deg (1-sigma)

    Declination            =   60.176 deg
       variance (analytic) =    1.185 deg (1-sigma)
       variance (numeric)  =    0.021 deg (1-sigma)


Unit sphere view vector right ascension, declination (deg) : 180,45
To view the sphere, go to Matlab and enter the command "adsplot"
```

```
                ***  Main Menu  ***

 1. Calculate attitude using Sun-Midscan       (SDM) method
 2. Calculate attitude using Sun-Chordwidth    (SDW) method
 3. Calculate attitude using Sun-Leading Edge  (SDI) method
 4. Calculate attitude using Sun-Trailing Edge (SDO) method
 5. Calculate attitude using Sun-Only          (SS)  method
 6. Quit

 Enter selection: 2


************************
****** SDW Method ******
************************


*** Attitude estimate 1  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  166.826 deg
       variance (analytic) =    1.444 deg (1-sigma)
       variance (numeric)  =    0.021 deg (1-sigma)

    Declination            =   64.324 deg
       variance (analytic) =    4.052 deg (1-sigma)
       variance (numeric)  =    0.018 deg (1-sigma)


*** Attitude estimate 2  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  164.661 deg
       variance (analytic) =    1.528 deg (1-sigma)
       variance (numeric)  =    0.024 deg (1-sigma)

    Declination            =   66.059 deg
       variance (analytic) =    3.964 deg (1-sigma)
       variance (numeric)  =    0.018 deg (1-sigma)


Unit sphere view vector right ascension, declination (deg) : 180,45
To view the sphere, go to Matlab and enter the command "adsplot"
```

```
                ***  Main Menu  ***

 1. Calculate attitude using Sun-Midscan       (SDM) method
 2. Calculate attitude using Sun-Chordwidth    (SDW) method
 3. Calculate attitude using Sun-Leading Edge  (SDI) method
 4. Calculate attitude using Sun-Trailing Edge (SDO) method
 5. Calculate attitude using Sun-Only          (SS)  method
 6. Quit

 Enter selection: 3


************************
****** SDI Method ******
************************


*** Attitude estimate 1  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  160.903 deg
       variance (analytic) =    0.819 deg (1-sigma)
       variance (numeric)  =    0.026 deg (1-sigma)

    Declination            =   68.487 deg
       variance (analytic) =    1.791 deg (1-sigma)
       variance (numeric)  =    0.015 deg (1-sigma)


*** Attitude estimate 2  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  165.839 deg
       variance (analytic) =    0.787 deg (1-sigma)
       variance (numeric)  =    0.022 deg (1-sigma)

    Declination            =   65.149 deg
       variance (analytic) =    2.122 deg (1-sigma)
       variance (numeric)  =    0.017 deg (1-sigma)


Unit sphere view vector right ascension, declination (deg) : 180,45
To view the sphere, go to Matlab and enter the command "adsplot"
```

```
               ***  Main Menu  ***

 1. Calculate attitude using Sun-Midscan       (SDM) method
 2. Calculate attitude using Sun-Chordwidth    (SDW) method
 3. Calculate attitude using Sun-Leading Edge  (SDI) method
 4. Calculate attitude using Sun-Trailing Edge (SDO) method
 5. Calculate attitude using Sun-Only          (SS)  method
 6. Quit

 Enter selection: 4


************************
****** SDO Method ******
************************


*** Attitude estimate 1  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  165.643 deg
       variance (analytic) =    0.749 deg (1-sigma)
       variance (numeric)  =    0.019 deg (1-sigma)

    Declination            =   65.307 deg
       variance (analytic) =    2.014 deg (1-sigma)
       variance (numeric)  =    0.015 deg (1-sigma)


*** Attitude estimate 2  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  170.034 deg
       variance (analytic) =    0.540 deg (1-sigma)
       variance (numeric)  =    0.011 deg (1-sigma)

    Declination            =   61.118 deg
       variance (analytic) =    1.764 deg (1-sigma)
       variance (numeric)  =    0.012 deg (1-sigma)


Unit sphere view vector right ascension, declination (deg) : 180,45
To view the sphere, go to Matlab and enter the command "adsplot"
```

```
                ***  Main Menu  ***

 1. Calculate attitude using Sun-Midscan       (SDM) method
 2. Calculate attitude using Sun-Chordwidth    (SDW) method
 3. Calculate attitude using Sun-Leading Edge  (SDI) method
 4. Calculate attitude using Sun-Trailing Edge (SDO) method
 5. Calculate attitude using Sun-Only          (SS)  method
 6. Quit

 Enter selection: 5


*****************************
****** Sun-only Method ******
*****************************


Sun azimuth angle          =   -74.2904 deg
Sun azimuth rate           =     0.9528 deg/day
Sun angle                  =   102.0835 deg     (linear fit to data)
Sun angle rate             =     0.3589 deg/day (linear fit to data)
Sun angle rate uncertainty =     0.0451 deg/day (1-sigma)

*** Attitude estimate 1  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  166.098 deg
       variance (analytic) =    0.929 deg (1-sigma)
       variance (numeric)  =    N/A   deg (1-sigma)

    Declination            =   64.934 deg
       variance (analytic) =    2.547 deg (1-sigma)
       variance (numeric)  =    N/A   deg (1-sigma)


*** Attitude estimate 2  (Mean-ecliptic-of-J2000 reference frame)

    Right ascension        =  166.098 deg
       variance (analytic) =    0.929 deg (1-sigma)
       variance (numeric)  =    N/A   deg (1-sigma)

    Declination            =  -64.934 deg
       variance (analytic) =    2.547 deg (1-sigma)
       variance (numeric)  =    N/A   deg (1-sigma)


Unit sphere view vector right ascension, declination (deg) : 180,45
To view the sphere, go to Matlab and enter the command "adsplot"
```

```
                ***  Main Menu  ***

 1. Calculate attitude using Sun-Midscan       (SDM) method
 2. Calculate attitude using Sun-Chordwidth    (SDW) method
 3. Calculate attitude using Sun-Leading Edge  (SDI) method
 4. Calculate attitude using Sun-Trailing Edge (SDO) method
 5. Calculate attitude using Sun-Only          (SS)  method
 6. Quit

 Enter selection: 6



 Note: IEEE floating-point exception flags raised:
    Inexact;  Invalid Operation;
 See the Numerical Computation Guide, ieee_flags(3M)
feta%
```

Step 3.5 View the attitude unit sphere plots

After each attitude solution is calculated by ADS, it is recommended that the user display the attitude solution unit sphere using Matlab. To do this, invoke a second terminal window (if you don't know how, ask one of the Flight Controllers). Connect to the ADS directory by typing the UNIX command

cd ~/att/ads

Then type the command

matlab

When you receive the Matlab prompt (>>), type the command

adsplot

Matlab will display the attitude solution unit sphere. Superimposed on this sphere are a number of items:

- The celestial equator and the prime meridian are shown in green. The Line of Aries passes through the intersection of these two.
- The two attitude solutions are shown as red and lavender "+"s. If multiple attitude data groups survived the ADS filtering processing, then there will be multiple red and lavender "+"s.

- The sun is displayed as yellow “o”s. The central body (Earth or Moon) is displayed as white “o”s. A streak of white “o”s occurs if the spacecraft moves a significant distance during the attitude solution time span.

- The intersection of the sun cone with the unit sphere is displayed as a yellow dashed line.

- The intersection of the central body (Earth or Moon) cone with the unit sphere is displayed as a white dashed line. There will be two of these lines, reflecting the two possible cones.

To print the attitude solution unit sphere, type the command

print

at the Matlab prompt.

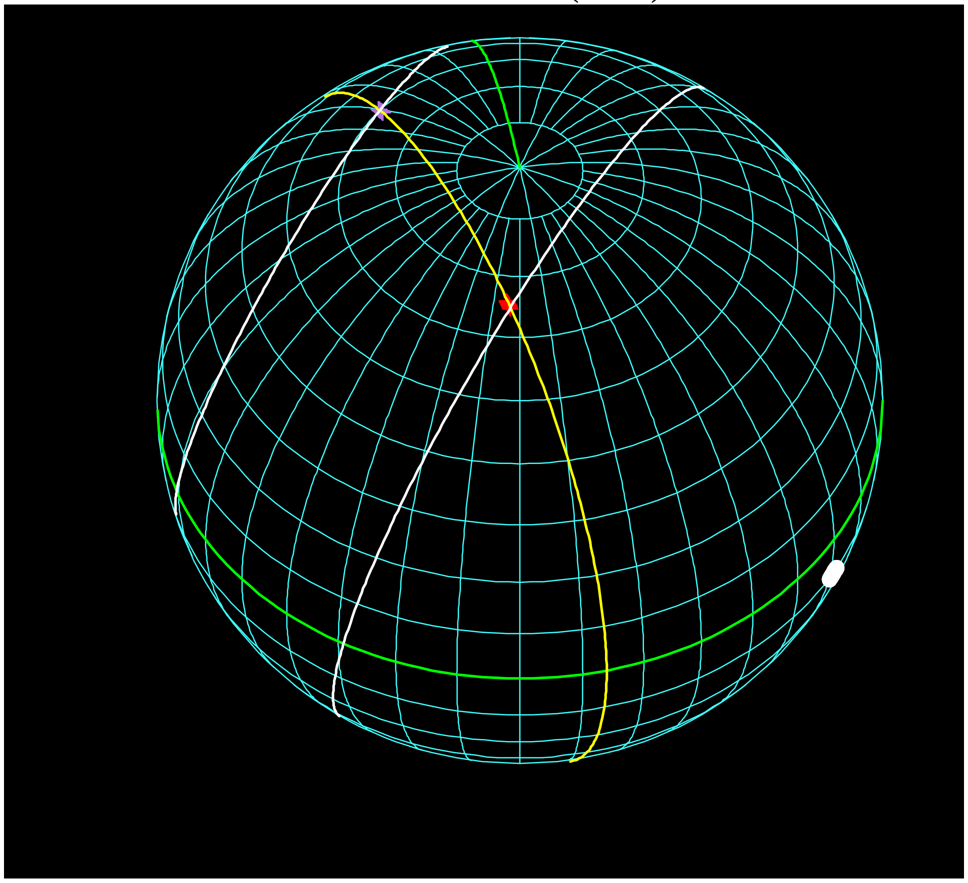
When you are done with ADS, quit out of Matlab by typing the command

quit

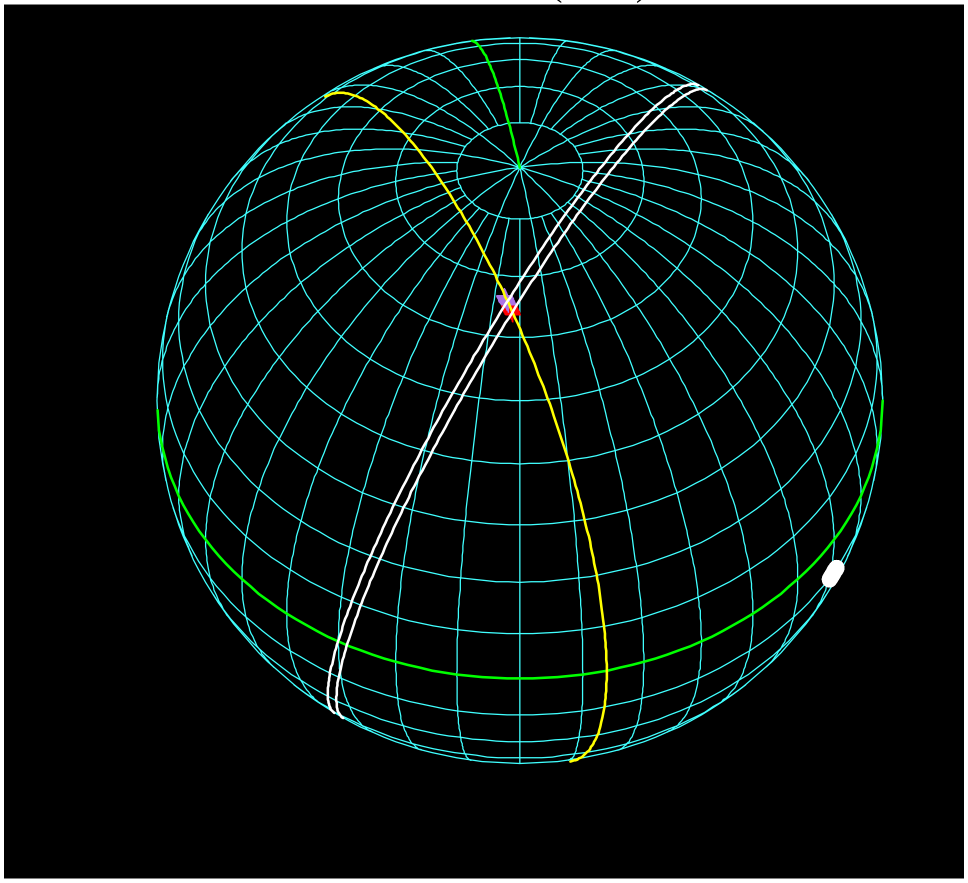
at the Matlab prompt.

Attitude unit sphere plots for each of the five attitude solution techniques are shown below for the example used in Step 3.4.

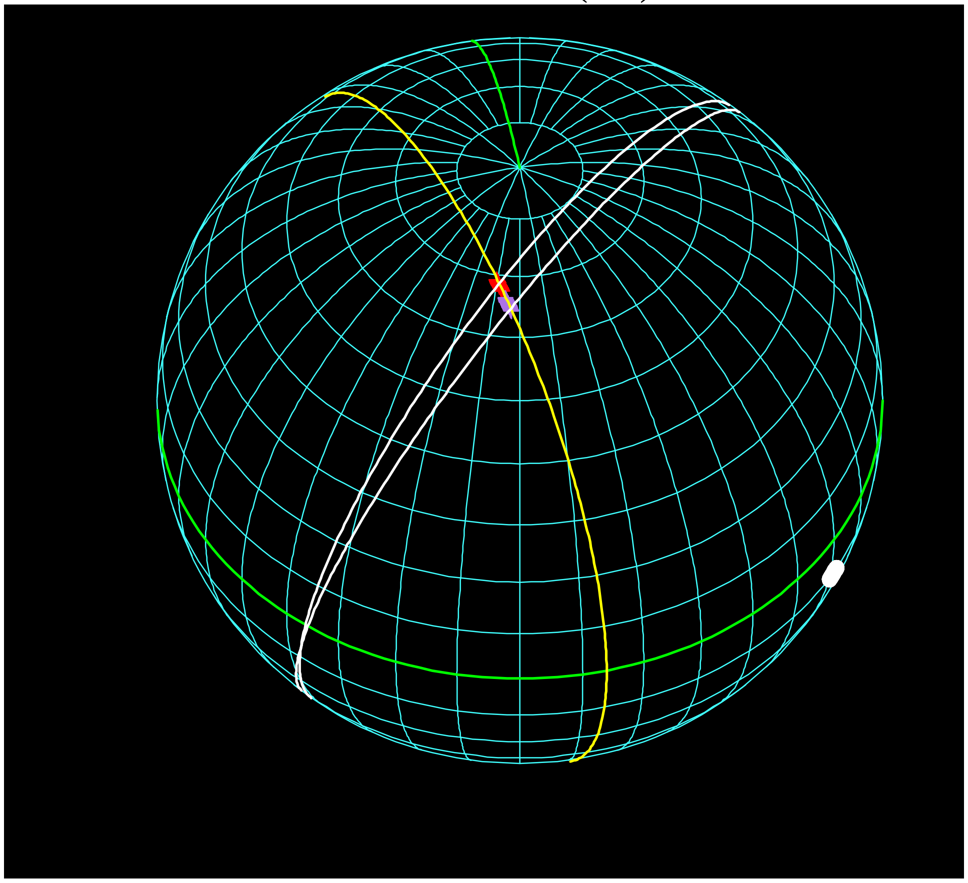
Sun - Disk Midscan (SDM) Method



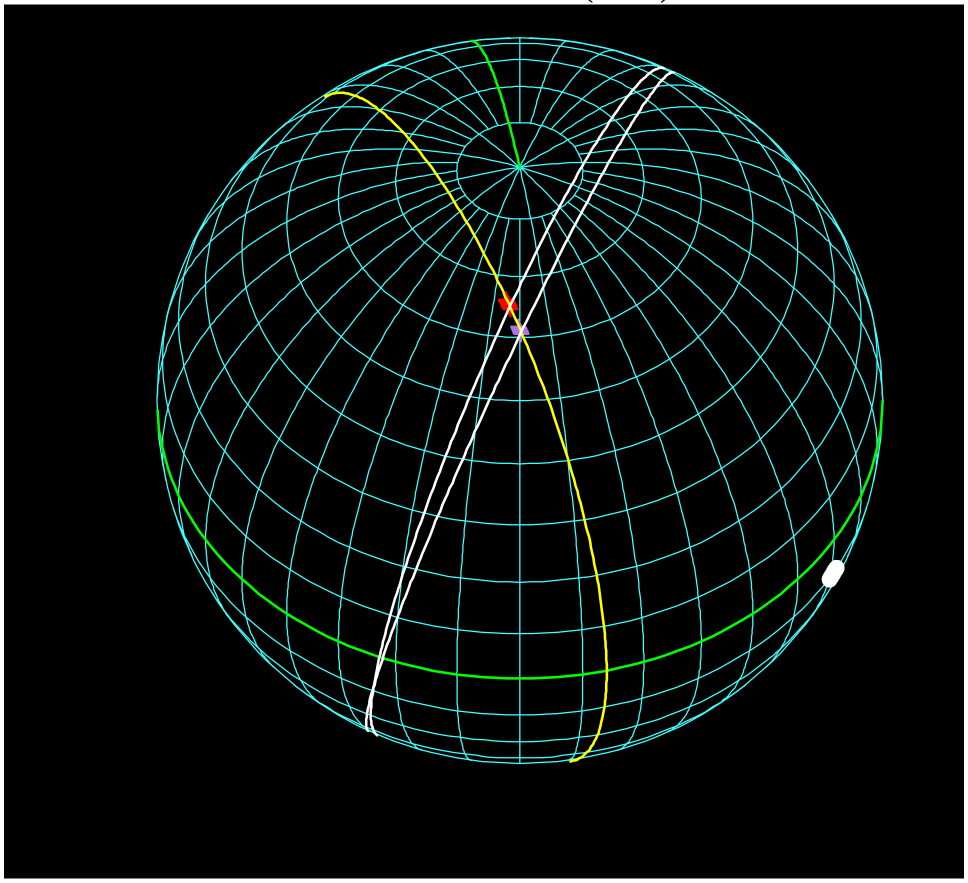
Sun - Disk Width (SDW) Method



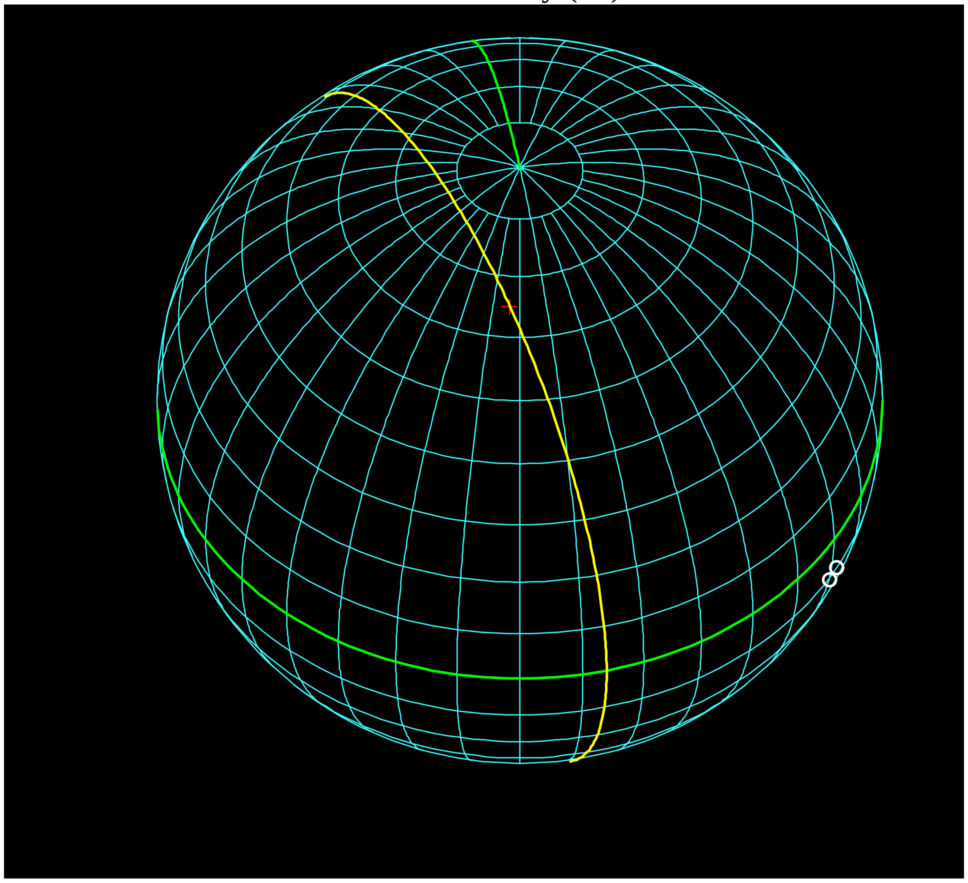
Sun - Disk Cross-In (SDI) Method



Sun - Disk Cross-Out (SDO) Method



Sun Sensor Only (SS) Method



Step 3.6 Validate the EMS calibration curves

If the estimated attitude that results from running ADS in Step 3.4 differs significantly from the attitude used in **LP_attitude.dat** in Step 1.3, the EMS calibration curves may not be correct. Update **LP_attitude.dat** with the new attitude estimate, then proceed through the attitude solution process again.

11 COMMAND GENERATION SOFTWARE (CGS)

Lunar Prospector supports three types of thruster maneuvers: spin rate change, spin axis reorientation, and delta-V. The Command Generation Software (CGS) is used to convert a maneuver into a sequence of spacecraft commands.

Location

The CGS executable, its source code, and its input files can all be found on each of the Lunar Prospector Mission Control Center (MCC) Sun workstations (feta, gouda, and gouda2) in the directory **~lunargrp/att/cgs/**. The executable file is named **cgs**.

Input Files

CGS requires the following input files in order to function properly.

File name	Description / usage
mn2000.dat	NASA SLP file. Provides vectors to Earth, Moon, Sun, and other planets. This file should never have to be changed, unless instructed to do so by the Flight Dynamics Facility (FDF) at GSFC.
LP_ephem.dat	Spacecraft ephemeris file. Provides vectors from the Earth or the Moon to the spacecraft as a function of time. This file is updated regularly by FDF, and should be replaced each time a new update is available. Updates are downloaded from the FDF website at http://fdd.gsfc.nasa.gov/lp
LP_attitude.dat	Spacecraft attitude file. Contains current best estimate of spacecraft spin rate and spin axis attitude (right ascension and declination relative to the ecliptic mean-of-J2000 reference frame). This file is generated by the Attitude Determination Software (ADS), or can be updated manually.
LP_sc.dat	Spacecraft description file. Contains current best estimate of a number spacecraft parameters. This file is created prior to launch, and may be updated manually if flight data indicates parameters differ from pre-flight estimates.
CGS_seqnum.dat	CGS sequence number. CGS numbers its output files with a unique 3-digit identifier. This file contains the most recent number used. If this file does not exist, CGS will create it and start numbering the output files from 001.
pulse1214_data_A1 pulse1214_data_A2	Thruster acceptance test performance data for the 0.2 on / 4.8 off duty cycle. These files should not be changed.

pulse1214_data_A3 pulse1214_data_A4 pulse1214_data_T1 pulse1214_data_T2	
pulse1260_data_A1 pulse1260_data_A2 pulse1260_data_A3 pulse1260_data_A4 pulse1260_data_T1 pulse1260_data_T2	Thruster acceptance test performance data for the 0.833 on / 4.167 off duty cycle. These files should not be changed.
pulse6060_data_A1 pulse6060_data_A2 pulse6060_data_A3 pulse6060_data_A4 pulse6060_data_T1 pulse6060_data_T2	Thruster acceptance test performance data for the 0.167 on / 0.833 off duty cycle. These files should not be changed.
ss_data_A1 ss_data_A2 ss_data_A3 ss_data_A4 ss_data_T1 ss_data_T2	Thruster acceptance test performance data for continuous burns. These files should not be changed.

Sample LP_attitude.dat file

This is a sample LP_attitude.dat file. Note the “last update” field is for information only – it is not used by CGS.

```
LUNAR PROSPECTOR ATTITUDE DATA FILE

last update                            yymmdd.hhmmss  =   970214.091223
current spin axis right ascension      deg            =      316.147
current spin axis declination          deg            =       82.392
current spin rate                      rpm            =       12.000
```

Sample LP_sc.dat file

This is a sample LP_sc.dat file. Note the “last updated” field is for information only – it is not used by CGS.

```
LUNAR PROSPECTOR S/C DATA FILE     last updated: 07-03-97

|-------------  ITEM DESCRIPTION  -----|- UNITS -|------  VALUE  ----|
S/C dry mass                            kg       =       167.0
Pressurant gas constant (PoVo/To)       lbf-in/K =      4478.
```

```
Propellant tank volume (total)          in^3      =     11306.           Per Irv
Benard
Dynamic pressure drop @ 450 psia, 1 REA psia      =         5.7

A1 moment arm length                    m         =         0.2289
A2 moment arm length                    m         =         0.2289
A3 moment arm length                    m         =         0.2404
A4 moment arm length                    m         =         0.2404
T1 moment arm length                    m         =         0.7455
T2 moment arm length                    m         =         0.7455

A1 cant angle                           deg       =        10.000
A2 cant angle                           deg       =        10.000
A3 cant angle                           deg       =        10.000
A4 cant angle                           deg       =        10.000
T1 cant angle                           deg       =         0.000
T2 cant angle                           deg       =         0.000

A1 azimuth angle from sun sensor        deg       =        90.000
A2 azimuth angle from sun sensor        deg       =       270.000
A3 azimuth angle from sun sensor        deg       =       270.000
A4 azimuth angle from sun sensor        deg       =        90.000
T1 azimuth angle from sun sensor        deg       =       180.000
T2 azimuth angle from sun sensor        deg       =       180.000
```

Sample LP_ephem.dat file

This is the first 100 lines from a sample LP_ephem.dat file.

```
EPHEMERIS FILE INFORMATION:
---------------------------

SATELLITE ID NUMBER =   1234567
RUN TITLE = Swingby trajectory file
TAPE IDENTIFIER =   EPHEMERIS START TIME (YYMMDD.)    =    971024.0000000000000000
EPHEMERIS START TIME (SEC OF DAY) =     43680.0000000000000000
EPHEMERIS END TIME (YYMMDD.)      =    971029.0000000000000000
EPHEMERIS END TIME (SEC OF DAY)   =     21600.0000000000000000
EPHEMERIS DELTA-T (SECONDS)       =        60.0000000000000000
ORBIT THEORY (COWELL OR BROUWER)  = SWINGBY
INTEGRATION STEP (2=FIXED)        =    1
COORDINATE SYSTEM           = 2000
COORD. SYSTEM INDICATOR     =    4
YYMMDD OF FILE CREATION     =       .0
HHMMSS.SSS OF FILE CREATION =       .000


INITIAL CONDITION INFORMATION:
------------------------------

ELEMENT EPOCH (MM/DD/YY  HH:MM:SS.SSS) =  10/24/97  12: 3: 6.604
INITIAL CARTESIAN ELEMENTS :    X (KM) =       6192.159421782063
                                Y (KM) =       2119.446935818197
                                Z (KM) =       483.9800811566426
                           VX (KM/SEC) =      -3.474204660876510
                           VY (KM/SEC) =       8.927902437407839
                           VZ (KM/SEC) =       5.273369238692247
INITIAL KEPLERIAN ELEMENTS :    A (KM) =       211075.3865486888
                                    E  =       .9689082382189328
                               I (DEG) =       29.20059862270360
                            RAAN (DEG) =       11.29180244261511
                              AP (DEG) =       8.756558435220825
                              MA (DEG) =       359.9997565447050
INITIAL BROUWER MEAN ELEMENTS : A (KM) =       .0000000000000000
                                    E  =       .0000000000000000
                               I (DEG) =       .0000000000000000
                            RAAN (DEG) =       .0000000000000000
                              AP (DEG) =       .0000000000000000
                              MA (DEG) =       .0000000000000000
CENTRAL BODY INDICATOR (1=EARTH)          =     1.0
DRAG PERTURBATION (0=YES,1=NO)            =     1.0
SOLAR RADIATION PERTURBATION (0=YES,1=NO) =      .0
SUN POINT MASS PERTURBATION (0=YES,1=NO)  =      .0
MOON POINT MASS PERTURBATION (0=YES,1=NO) =      .0
SPACECRAFT AREA (M**2)          =       1.750000000000000
SPACECRAFT MASS (KG)            =       290.0000000000000
DRAG COEFFICIENT                =       .1118707184331741+249
SOLAR REFLECTIVITY COEFFICIENT  =       .1237142423143447-310
ATMOSPHERIC DENSITY MODEL       =    RHO1  =       .0000000000000000
RHO2  =       .0000000000000000
RHO3  =       .0000000000000000
RHO4  =       .0000000000000000
```

```
 OTHER INFORMATION:
 ------------------

 ECCENTRIC ANOMALY AT EPOCH (DEG)     =       .0000000000000000
 TRUE ANOMALY AT EPOCH (DEG)          =       .0000000000000000
 ANOMALISTIC PERIOD (MINUTES)         =       .0000000000000000
 GREENWICH HOUR ANGLE AT EPOCH (RAD)  =      3.729912320667935
 INITIAL GREENWICH HOUR ANGLE (RAD)   =       .0000000000000000
 FINAL GREENWICH HOUR ANGLE (RAD)     =       .0000000000000000
 GEOCENTRIC SUN POSITION AT EPOCH (KM):  X  =   -127328450.81184146
                                         Y  =   -70610639.641960904
                                         Z  =   -30614463.207206012
 YYMMDD OF EARLIEST MEASUREMENT       =        .0
 HHMMSS.SSS OF EARLIEST MEASUREMENT   =        .000
 YYMMDD OF LATEST MEASUREMENT         =        .0
 HHMMSS.SSS OF LATEST MEASUREMENT     =        .000
 LEAP SECOND DURING EPHEM (2=YES)     =     1


 EPHEMERIS:
 ----------

TIME FROM EPOCH (SEC)        X (KM)               Y (KM)               Z (KM)             XDOT (KM/S)          YDOT (KM/S)          ZDOT (KM/S)
--------------------- --------------------- --------------------- --------------------- --------------------- --------------------- ---------------------
  .00000000000000E+00   .48298824177407E+04   .45645422859749E+04   .19729858593162E+04  -.56637057863063E+01   .76180323885110E+01   .47936771735154E+01
  .60000000000000E+02   .44800737346330E+04   .50117123020836E+04   .22562554120957E+04  -.59876946609165E+01   .72850544370187E+01   .46462519334718E+01
  .12000000000000E+03   .41123058685884E+04   .54386489645365E+04   .25303778814502E+04  -.62657584549349E+01   .69474777017087E+01   .44912865605401E+01
  .18000000000000E+03   .37290943026308E+04   .58453573179796E+04   .27950815626817E+04  -.65001324623619E+01   .66104795165608E+01   .43318727275832E+01
  .24000000000000E+03   .33330615647456E+04   .62320495209826E+04   .30501970820905E+04  -.66941923787711E+01   .62792069947271E+01   .41712098187818E+01
  .30000000000000E+03   .29265619139523E+04   .65990756315006E+04   .32956600890566E+04  -.68512858826169E+01   .59577414933563E+01   .40119074496443E+01
  .36000000000000E+03   .25115408493271E+04   .69472011181714E+04   .35316792327141E+04  -.69756268683110E+01   .56481066762390E+01   .38555366905197E+01
  .42000000000000E+03   .20900448430904E+04   .72771543067920E+04   .37584317562185E+04  -.70715002364134E+01   .53526811699607E+01   .37038717211853E+01
  .48000000000000E+03   .16634775959998E+04   .75898057335611E+04   .39762429506771E+04  -.71423628829411E+01   .50720324311808E+01   .35576002879126E+01
  .54000000000000E+03   .12333700500254E+04   .78861092014295E+04   .41854700964939E+04  -.71922444522253E+01   .48067320830684E+01   .34174744306113E+01
  .60000000000000E+03   .80080665307824E+03   .81669313888038E+04   .43864710670037E+04  -.72240179057173E+01   .45567729616430E+01   .32838025785932E+01
  .66000000000000E+03   .36679828624311E+03   .84332219526899E+04   .45796573532320E+04  -.72406460291880E+01   .43217192638157E+01   .31566670172430E+01
  .72000000000000E+03  -.67750215879742E+02   .86858673992675E+04   .47654172631012E+04  -.72447530735118E+01   .41011535971540E+01   .30361235111361E+01
  .78000000000000E+03  -.50229627876073E+03   .89256474422765E+04   .49441197815264E+04  -.72380492310645E+01   .38943355591298E+01   .29219511409373E+01
  .84000000000000E+03  -.93620841410028E+03   .91534093824362E+04   .51161613630041E+04  -.72225299834361E+01   .37004683825680E+01   .28139289028041E+01
  .90000000000000E+03  -.13688779823690E+04   .93699689718242E+04   .52819236168991E+04  -.72001034108871E+01   .35188144141681E+01   .27118588223437E+01
  .96000000000000E+03  -.18000474188351E+04   .95759358098255E+04   .54417130590641E+04  -.71716889333593E+01   .33486462509306E+01   .26154413655199E+01
  .10200000000000E+04  -.22294072494487E+04   .97719873586636E+04   .55958678293598E+04  -.71383822025427E+01   .31891282373661E+01   .25243366117101E+01
```

Output Files

In most instances, CGS generates three output files for each maneuver: a thruster parameter file (axv, rer, spn, or tnv), an execute file (exc), and an echo file (echo). These files are described below. In all cases the "xxx" represents the CGS sequence number.

File name	Description / usage
axvxxx.mne	File containing the MGDS script to uplink the thruster parameters for an axial delta-V maneuver.
rerxxx.mne	File containing the MGDS script to uplink the thruster parameters for a spin axis reorientation maneuver.
spnxxx.mne	File containing the MGDS script to uplink the thruster parameters for a spin rate change maneuver.
tnvxxx.mne	File containing the MGDS script to uplink the thruster parameters for a tangential delta-V maneuver.
excxxx.mne	File containing the MGDS script to start the maneuver execution, send the backup STOPFIRE command after the maneuver is completed, and safe the thruster logic using the SETCMDREG command.
echoxxx.cgs	File containing a complete playback of the complete CGS session. This file is to check the inputs and calculations used to generate the thruster parameter and execute files.

If a pulsed maneuver (reorientation or tangential delta-V) requires more than 255 pulses, then the maneuver will be broken into multiple 255 pulse segments, followed by a smaller segment containing the remaining pulses. Each segment will have its own parameter and execute files, but only one echo file will be produced.

If an axial delta-V maneuver uses thruster pair A3 and A4 and the total on-time exceeds the maximum allowable on-time for heating of the MGA radome, then the maneuver will be broken into multiple segments, each segment shorter than the maximum on-time limit. Each segment will have its own parameter and execute files, but only one echo file will be produced.

A vector burn maneuver has an axial component and a tangential component. Each component will have its own parameter and execute files (which could themselves be subdivided as described in the previous two paragraphs). Only one echo file will be produced.

What To Do With The Output Files

The “echo” file and the “.mne” files should be reviewed to verify the accuracy of all inputs and to ensure that the commands to be uplinked will indeed produce the desired maneuver. Once this check is complete, all of the “.mne” files produced for the maneuver should be ftp’d to the **~lunargrp/mnefiles/** directory on the MCC command console. Assuming that the command console is gouda and the .mne files are axv025.mne and exc025.mne, the following sequence of commands will work:

```
ftp gouda
(login as lunargrp, password lunar123)
cd mnefiles
send axv025.mne
send exc025.mne
quit
```

Using CGS For a Spin Rate Maneuver

This section describes how to use CGS for a spin rate change maneuver, using as an example the Lunar Prospector initial spin down maneuver from 57 rpm to 43 rpm.

Here are the contents of the “echo002.cgs” file, which is a complete playback of the interactive session between CGS and the user. User inputs are bold and underlined.

```
                   LUNAR PROSPECTOR
             Command Generation Software
                     Version 1.6

        Copyright 1997 Lockheed Martin Corporation
                  All rights reserved



          ***  Main Menu  ***

1. Generate spin rate maneuver commands
2. Generate reorientation maneuver commands
3. Generate vector burn maneuver commands (includes axial delta-V)
4. Quit

Enter selection : 1


          ***  Spin Rate Maneuver  ***

 Current spin rate is                         :   57.000 rpm
 Enter desired spin rate (rpm)                :   43.000
 Enter spacecraft spin inertia (kg-m^2)       :   64.300
 Enter delay from uplink till execution (mins) :    0.000
 Enter current tank pressure (psia)           :  439.000


 ----------  Spin Maneuver thruster parameters  ----------
```

```
   Current spin rate          =   57.000 rpm
   Target spin rate           =   43.000 rpm
   Current spin inertia       =   64.300 kg-m^2
   Tank pressure              =   439.00 psia
   REA feed pressure          =   433.58 psia

   Burn duration              =    3.598 sec
   Propellant consumed        =    0.055 kg

   ### THRUSTER T1 ###
       Moment arm                 =    0.746 m
       Mean thrust                =   35.140 N
       Isp                        =  234.021 sec


--------------  Maneuver Parameter Conversion Summary  --------------
SUN PULSE DELAY    =    0.0000 (sec)  =     0 (counts)  =  0000 (hex)
BURN DURATION      =    3.5985 (sec)  =  6477 (counts)  =  194D (hex)
HALFREV TIME       =    0.0000 (sec)  =     0 (counts)  =  0000 (hex)
EXECUTE DELAY      =      0.00 (min)  =     0 (counts)  =    00 (hex)
NUMBER OF PULSES   =                        1 (counts)  =    01 (hex)
THRUSTER COMMAND   =   T1D


---------  Maneuver Uplink Command List  ---------
DELAYSUN    0.0000         Confirmation  =  310000
SHORTDUR    3.5985         Confirmation  =  32194D
HALFREV     0.0000         Confirmation  =  330000
DELAYNUM    0.00    1      Confirmation  =  300001
T1D                        Confirmation  =  3C0050


MGDS parameter file for this maneuver is :  spn002.mne
MGDS execute   file for this maneuver is :  exc002.mne
```

The current spin rate is taken from the LP_attitude.dat file.

The desired spin rate is the target spin rate for the maneuver.

The spacecraft spin inertia is the z-axis inertia. Current estimates are that the spin inertia is about 64 kg-m^2 before boom deployment, and about 320 kg-m^2 after deployment. These numbers will be known more precisely after spacecraft spin balancing is completed in September.

The delay from uplink till execution is the FIREDELAY parameter. For all spin rate maneuvers in the nominal Lunar Prospector timeline this delay is zero.

The current tank pressure is read from the spacecraft telemetry (telemetry point TANKPRESS).

The “Spin Maneuver thruster parameters” summary is provided by CGS to allow the user to review the accuracy of the inputs and the maneuver duration calculations.

The “Maneuver Parameter Conversion Summary” provides the conversion of all maneuver parameters from engineering units to hex.

The “Maneuver Uplink Command List” provides the complete command mnemonic list with parameters to be uplinked to the spacecraft, and the LASTCMD confirmation of each mnemonic that will be seen in the downlink telemetry.

For this particular run, the thruster parameter file is “spn002.mne”, and the execute file is “exc002.mne”.

Here is the contents of the thruster parameter file “spn002.mne”. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:16:18.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt DELAYSUN,    0.0000
mst +00:00:32.0
xlt SHORTDUR,    3.5985
xlt HALFREV,     0.0000
xlt DELAYNUM,       0.0,   1
xlt T1D

$$EOS

$$EOF
```

Here is the contents of the execute file “exc002.mne”. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft. Note that the STOPFIRE command is issued approximately one-half second after the nominal completion of the maneuver.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
```

```
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:16:18.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt EXEC
mst +00:00:04.1
xlt STOPFIRE
mst +00:00:32.0
xlt SETCMDREG
xlt EXEC

$$EOS

$$EOF
```

Using CGS For a Reorientation Maneuver

This section describes how to use CGS for spin axis reorientation maneuver, using as an example the Lunar Prospector initial reorientation maneuver from injection attitude to trans-lunar cruise attitude.

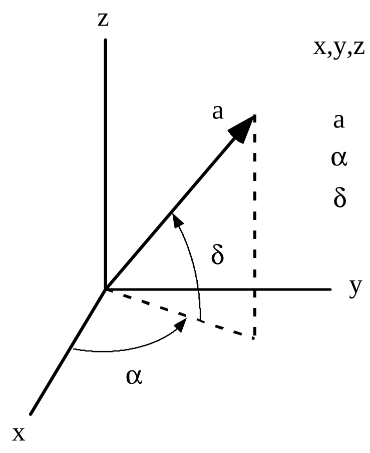
Here are the contents of the “echo001.cgs” file, which is a complete playback of the interactive session between CGS and the user. User inputs are bold and underlined.

```
                      LUNAR PROSPECTOR
                Command Generation Software
                        Version 1.6

         Copyright 1997 Lockheed Martin Corporation
                   All rights reserved



           ***  Main Menu  ***

1. Generate spin rate maneuver commands
2. Generate reorientation maneuver commands
3. Generate vector burn maneuver commands (includes axial delta-V)
4. Quit

Enter selection : 2


           ***  Reorientation Maneuver  ***

 NOTE: Attitude right ascension and declination are relative
       to the Mean-of-J2000 ecliptic reference frame

 Current spin rate is                 :   57.00 rpm  (342.000 deg/sec)
 Current spin axis right ascension is :  108.66 deg
 Current spin axis declination is     :    6.76 deg

 Enter desired spin axis right ascension (deg) :  316.147
 Enter desired spin axis declination (deg)     :   82.392
 Enter thruster pulse width (sec)              :    0.167

 Reorientation modes:
   1. A1
   2. A2
   3. A3
   4. A4
   5. A1 & A4
   6. A2 & A3
   7. A1A3 & A2A4
 Enter desired mode : 5

 Enter spacecraft spin inertia (kg-m^2)        :   64.300
 Enter current tank pressure (psia)            :  450.000
 Enter maneuver start time (yymmdd.hhmmss UTC) : 971024.122500
 Ignore pulse centroid delay? (answer "yes" ONLY for VES) : n


 --------------  Reorientation Maneuver attitude data  --------------
```

```
    Sun vector                       (J2000 ecliptic frame) =  -0.855640 -0.517572 -
0.000015
    Pre- maneuver spin axis attitude (J2000 ecliptic frame) =  -0.317696  0.940862
0.117676
    Post-maneuver spin axis attitude (J2000 ecliptic frame) =   0.095472 -0.091725
0.991197

    Post-maneuver spin axis azimuth (Sun-polar frame) =  -90.4318 deg
    Pre- maneuver sun angle  =  102.4233 deg
    Post-maneuver sun angle  =   91.9616 deg
    Rhumb line heading angle =  173.3397 deg
    Rhumb line length        =   90.2002 deg


 ------  Reorientation Maneuver thruster parameters  ------

    Current spin rate          =   57.000 rpm
    Current spin inertia       =   64.300 kg-m^2
    Tank pressure              =  450.000 psia
    REA feed pressure          =  444.300 psia
    Current angular momentum   =  383.808 N-m-s
    Number of pulses           =  227
    Maneuver duration          =  238.947 sec
    Propellant consumed        =    1.162 kg

    ### THRUSTER A1 ###
        Azimuth wrt sun sensor     =   90.000 deg
        Moment arm                 =    0.229 m
        Thrust                     =   35.648 N
        Specific impulse           =  231.941 sec
        Pulse centroid fraction    =    0.766

    ### THRUSTER A4 ###
        Azimuth wrt sun sensor     =   90.000 deg
        Moment arm                 =    0.240 m
        Thrust                     =   35.094 N
        Specific impulse           =  238.754 sec
        Pulse centroid fraction    =    0.832


 --------------  Maneuver Parameter Conversion Summary  --------------
 SUN PULSE DELAY    =    0.1157 (sec)  =   208 (counts)  =  00D0 (hex)
 BURN DURATION      =    0.1670 (sec)  =   301 (counts)  =  012D (hex)
 HALFREV TIME       =    0.3483 (sec)  =   627 (counts)  =  0273 (hex)
 EXECUTE DELAY      =      0.00 (min)  =     0 (counts)  =    00 (hex)
 NUMBER OF PULSES   =                      227 (counts)  =    E3 (hex)
 THRUSTER COMMAND   =   A14REOR


 ---------  Maneuver Uplink Command List  ---------
 DELAYSUN    0.1157         Confirmation  =  3100D0
 SHORTDUR    0.1670         Confirmation  =  32012D
 HALFREV     0.3483         Confirmation  =  330273
 DELAYNUM    0.00  227      Confirmation  =  3000E3
 A14REOR                    Confirmation  =  380048


 MGDS parameter file for this maneuver is :  rer001.mne
 MGDS execute   file for this maneuver is :  exc001.mne
```

The current spin rate and spin axis attitude are taken from the LP_attitude.dat file.

The desired spin axis right ascension and declination are the target attitude for this maneuver. These <u>must</u> be entered relative to the mean-of-J2000 ecliptic reference frame. The right ascension is the angle from the reference frame x-axis to the projection of the attitude vector on to the reference frame xy-plane (measured in the right-hand rule sense about the +z axis of the reference frame). The declination is the angle from the xy-plane to the desired attitude vector, measured positive toward the +z axis of the reference frame.



x,y,z	=	basis vectors of mean-of-J2000 ecliptic reference frame
a	=	attitude vector
α	=	attitude right ascension
δ	=	attitude declination

The thruster pulse width is the duration of each thruster pulse. For the initial reorientation from injection attitude to translunar cruise attitude, this pulse width is 0.167 seconds. For all other reorientation maneuvers the pulsewidth is 0.200 seconds.

There are seven commandable combinations of thrusters that can reorient the spacecraft. In the nominal timeline Lunar Prospector will use pair A1&A4 (option 5) for all reorientation maneuvers. Should either A1 or A4 fail, the backup thruster set for reorientation maneuvers is A2&A3.

The spacecraft spin inertia is the z-axis inertia. Current estimates are that the spin inertia is about 64 kg-m^2 before boom deployment, and about 320 kg-m^2 after deployment. These numbers will be known more precisely after spacecraft spin balancing is completed in September.

The current tank pressure is read from the spacecraft telemetry (telemetry point TANKPRESS).

The maneuver start time is the best estimate of when the “EXEC” in the execute file will be radiated to the spacecraft. This can be controlled to the nearest 0.1 second by

MGDS, but such precision isn't necessary for reorientation maneuvers. Accuracy to within half an hour is sufficient.

The pulse centroid delay is the difference between the actual and ideal midpoint of each thruster pulse. This delay is estimated using the thruster Acceptance Test performance data. For flight operations this delay must never be ignored (always answer NO for flight). The Vehicle & Environment Simulator (VES) used for operator training does not know about centroid delays, so if CGS is being used to generate a maneuver that will be simulated in the VES, answer YES to this question.

The "Reorientation Maneuver attitude data" and the "Reorientation Maneuver thruster parameters" are provided by CGS to allow the user to review the accuracy of the inputs and the maneuver duration and pulse count calculations. The sun vector is calculated using the SLP data (from mn2000.dat). The sun angles are measured from the spin axis to the sun vector. To convert these to the Sun Equatorial Angle (SEA) seen in telemetry, subtract the sun angle from 90 (ie, SEA = 90 – sun angle). The rhumb line length is the total arc through which the spin axis will move.

The "Maneuver Parameter Conversion Summary" provides the conversion of all maneuver parameters from engineering units to hex.

The "Maneuver Uplink Command List" provides the complete command mnemonic list with parameters to be uplinked to the spacecraft, and the LASTCMD confirmation of each mnemonic that will be seen in the downlink telemetry.

For this particular run, the thruster parameter file is "rer001.mne", and the execute file is "exc001.mne".

Here is the contents of the thruster parameter file "rer001.mne". This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:13:49.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS
```

```
$$DATA
xlt DELAYSUN,    0.1157
mst +00:00:32.0
xlt SHORTDUR,    0.1670
xlt HALFREV,     0.3483
xlt DELAYNUM,       0.0, 227
xlt A14REOR

$$EOS

$$EOF
```

Here is the contents of the execute file “exc001.mne”. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft. Note that the STOPFIRE command is issued approximately 10 pulse periods after the nominal completion of the maneuver.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:13:49.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt EXEC
mst +00:04:09.5
xlt STOPFIRE
mst +00:00:32.0
xlt SETCMDREG
xlt EXEC

$$EOS

$$EOF
```

Using CGS For a Vector Burn Maneuver

This section describes how to use CGS for a vector burn maneuver, using as an example the Lunar Prospector first mid-course correction maneuver. Maneuvers that are purely axial, such as the Lunar Orbit Insertion (LOI) burns, are a subclass of the vector burn maneuvers with the delta-V vector pointed along the spin axis.

CGS builds separate parameter and execute files for the axial and tangential components of the tangential burn. CGS assumes that the axial component of the maneuver is executed before the tangential component, meaning that the spacecraft is more massive and the thrust level is higher during the axial burn than during the tangential burn. The maneuvers should also be executed in this order, unless both components of the maneuver are less than a few meters per second.

Here are the contents of the "echo004.cgs" file, which is a complete playback of the interactive session between CGS and the user for the first mid-course correction maneuver. User inputs are bold and underlined.

```
          ***  Main Menu  ***

1. Generate spin rate maneuver commands
2. Generate reorientation maneuver commands
3. Generate vector burn maneuver commands (includes axial delta-V)
4. Quit

Enter selection : 3


          ***  Vector Burn Maneuver  ***

 NOTE: Delta-V right ascension and declination are relative
       to the Mean-of-J2000 ecliptic reference frame

 Enter delta-V magnitude (m/sec)                   :   70.000
 Enter right ascension of the delta-V vector (deg) :  255.000
 Enter declination of the delta-V vector (deg)     :    5.000
 Enter tangential thruster pulse width (sec)       :    0.833
 Enter current tank pressure (psia)                :  438.000
 Enter current average tank temperature (deg C)    :   20.000
 Enter estimated propellant mass remaining (kg)    :  136.724
 Will maneuver be executed in the dark (y/n)?      : n
 Enter delay from uplink till execution (mins)     :    0.000
 Enter maneuver start time (yymmdd.hhmmss UTC)     : 971024.160800


 -------------------  Vector Decomposition  -------------------

    Spin axis right ascension (J2000 ecliptic frame) =  316.15 deg
    Spin axis declination     (J2000 ecliptic frame) =   82.39 deg

    Spin axis attitude (J2000 ecliptic frame) =   0.095472 -0.091725  0.991197
    Sun vector         (J2000 ecliptic frame) =  -0.854118 -0.520079 -0.000056
    Delta-V vector     (J2000 ecliptic frame) =  -0.257834 -0.962250  0.087156
```

```
   Spin axis to sun     angle   =   91.9425 deg
   Spin axis to delta-V angle   =   81.3711 deg
   Delta-V axial      component =   10.5024 m/s
   Delta-V tangential component =   69.2077 m/s


----------  Axial Maneuver thruster parameters  ----------

   S/C dry mass                      =  167.000 kg
   Prop mass at start of burn        =  136.724 kg
   S/C mass at start of burn         =  303.724 kg

   Axial delta-V magnitude           =   10.502 m/s
   Burn duration                     =     46.0 sec
   Average axial thrust              =   69.164 N  (includes cant angle loss)
   Average Isp                       =  237.129 sec
   Propellant consumed               =    1.389 kg

   Tank pressure at start of burn    =   438.00 psia
   Tank pressure at end of burn      =   421.71 psia
   REA feed pressure at start of burn=   416.40 psia
   REA feed pressure at end of burn  =   401.69 psia

   A3/A4 on-time limit               =     20.8 sec

   ### THRUSTER A1 ###
       Cant angle                    =   10.000 deg
       Thrust at start of burn       =   35.773 N
       Thrust at end of burn         =   34.702 N
       Isp at start of burn          =  237.188 sec
       Isp at end of burn            =  236.718 sec

   ### THRUSTER A2 ###
       Cant angle                    =   10.000 deg
       Thrust at start of burn       =   35.847 N
       Thrust at end of burn         =   34.762 N
       Isp at start of burn          =  237.787 sec
       Isp at end of burn            =  237.139 sec


--------------  Maneuver Parameter Conversion Summary  --------------
SUN PULSE DELAY    =    0.0000 (sec)  =     0 (counts)  =  0000 (hex)
BURN DURATION      =   46.0146 (sec)  =   460 (counts)  =  01CC (hex)
HALFREV TIME       =    0.0000 (sec)  =     0 (counts)  =  0000 (hex)
EXECUTE DELAY      =      0.00 (min)  =     0 (counts)  =    00 (hex)
NUMBER OF PULSES   =                        1 (counts)  =    01 (hex)
THRUSTER COMMAND   =   A12LOI


---------  Maneuver Uplink Command List  ---------
DELAYSUN    0.0000         Confirmation  =  310000
LONGDUR    46.0146         Confirmation  =  3201CC
HALFREV     0.0000         Confirmation  =  330000
DELAYNUM    0.00    1      Confirmation  =  300001
A12LOI                     Confirmation  =  3F0056


MGDS parameter file for part 1 of this maneuver is :  axv004.mne
MGDS execute   file for part 1 of this maneuver is :  exc004.mne




----------  Tangential Maneuver thruster parameters  ----------
```

```
   S/C dry mass                      =  167.000 kg
   Prop mass at start of burn        =  135.335 kg
   S/C mass at start of burn         =  302.335 kg

   Tangential delta-V magnitude      =   69.208 m/s
   Tangential delta-V heading angle  =   42.737 deg (wrt sun pulse)
   Spin rate                         =   12.000 rpm   (  72.000 deg/sec)
   Number of pulses                  =      411
   Cumulative pulse time             =  342.139 sec
   Total maneuver time               = 2055.000 sec

   Pulse efficiency factor           = 0.954965
   Average tangential thrust         =   63.038 N
   Average Isp                       =  233.348 sec
   Average centroid                  =    0.541 (fraction of pulse)
   Propellant consumed               =    9.421 kg

   Tank pressure at start of burn    =   421.71 psia
   Tank pressure at end of burn      =   356.41 psia
   REA feed pressure at start of burn=   401.69 psia
   REA feed pressure at end of burn  =   342.11 psia

   ### THRUSTER T1 ###
       Azimuth angle                 =  180.000 deg
       Thrust at start of burn       =   33.700 N
       Thrust at end of burn         =   29.559 N
       Isp at start of burn          =  235.943 sec
       Isp at end of burn            =  234.536 sec
       Centroid at start of burn     =  0.53775 (fraction of pulse)
       Centroid at end of burn       =  0.54004 (fraction of pulse)

   ### THRUSTER T2 ###
       Azimuth angle                 =  180.000 deg
       Thrust at start of burn       =   33.753 N
       Thrust at end of burn         =   29.638 N
       Isp at start of burn          =  231.924 sec
       Isp at end of burn            =  231.067 sec
       Centroid at start of burn     =  0.54010 (fraction of pulse)
       Centroid at end of burn       =  0.54411 (fraction of pulse)


--------------  Maneuver Parameter Conversion Summary  --------------
SUN PULSE DELAY    =    0.1432 (sec)  =   258 (counts)  =  0102 (hex)
BURN DURATION      =    0.8330 (sec)  =  1499 (counts)  =  05DB (hex)
HALFREV TIME       =    1.6670 (sec)  =  3001 (counts)  =  0BB9 (hex)
EXECUTE DELAY      =      0.00 (min)  =     0 (counts)  =    00 (hex)
NUMBER OF PULSES   =                      255 (counts)  =    FF (hex)
THRUSTER COMMAND   =   T12VEC


---------  Maneuver Uplink Command List  ---------
DELAYSUN    0.1432        Confirmation  =  310102
SHORTDUR    0.8330        Confirmation  =  3205DB
HALFREV     1.6670        Confirmation  =  330BB9
DELAYNUM    0.00  255     Confirmation  =  3000FF
T12VEC                    Confirmation  =  3B004E


MGDS parameter file for part 2 of this maneuver is :  tnv005.mne
MGDS execute   file for part 2 of this maneuver is :  exc005.mne



--------------  Maneuver Parameter Conversion Summary  --------------
SUN PULSE DELAY    =    0.1432 (sec)  =   258 (counts)  =  0102 (hex)
```

```
BURN DURATION      =    0.8330 (sec)  =  1499 (counts)  =  05DB (hex)
HALFREV TIME       =    1.6670 (sec)  =  3001 (counts)  =  0BB9 (hex)
EXECUTE DELAY      =      0.00 (min)  =     0 (counts)  =    00 (hex)
NUMBER OF PULSES   =                     156 (counts)  =    9C (hex)
THRUSTER COMMAND   =   T12VEC


---------  Maneuver Uplink Command List  ---------
DELAYSUN    0.1432         Confirmation  =  310102
SHORTDUR    0.8330         Confirmation  =  3205DB
HALFREV     1.6670         Confirmation  =  330BB9
DELAYNUM    0.00  156      Confirmation  =  30009C
T12VEC                     Confirmation  =  3B004E


MGDS parameter file for part 3 of this maneuver is :  tnv006.mne
MGDS execute   file for part 3 of this maneuver is :  exc006.mne
```

The delta-V magnitude, right ascension, and declination are taken from the maneuver planning sheet provided by the Goddard Space Flight Center Flight Dynamics Facility (FDF). The desired spin axis right ascension and declination must be entered relative to the mean-of-J2000 ecliptic reference frame. See “Using CGS For a Reorientation Maneuver” for the definition and sense of the right ascension and declination.

The tangential thruster pulse width is the duration of each thruster pulse during the tangential segment of the burn. This pulse width is 0.833 seconds for all maneuvers in the baseline Lunar Prospector timeline. If contingency operations require executing a tangential delta-V at a spin rate above 24 rpm, then 0.167 seconds should be used instead.

The current tank pressure is read from the spacecraft telemetry (telemetry point TANKPRESS).

The average tank pressure is calculated from spacecraft telemetry:

avg tank temperature = (TANK1TMP + TANK2TMP + TANK3TMP) / 3

The estimated propellant mass remaining can be taken either from the propellant mass logbook maintained in the mission control center, or by multiplying the telemetry derived monitor FUEL by the propellant mass loaded on the spacecraft at launch (which at the time this is written is estimated to be 138 kg):

propellant remaining = 138 * (FUEL / 100)

The tangential burn logic in the C&DH is triggered by the sun pulse from the sun sensor. If the spacecraft is in eclipse (“in the dark”) this logic will not function, and the spacecraft will raise an error flag. To be able to execute tangential burns in eclipse, a special MGDS command file can be built by CGS that uses the ability of MGDS to precisely time commands to the spacecraft. Answering “yes” to the question “Will maneuver be executed in the dark (y/n)?” triggers this special CGS logic. For most vector burn maneuvers the correct answer is “no”. Response of CGS to a “yes” answer is described in the next section.

The delay from uplink till execution is the FIREDELAY parameter. This delay should be zero for all vector burn maneuvers in the nominal Lunar Prospector timeline except for the first Lunar Orbit Insertion (LOI) burn. For this burn the delay should be 60.0 minutes. The FIREDELAY is applied only to the axial segment of the vector burn. The spacecraft will not accept a non-zero delay prior to a tangential burn.

The maneuver start time is the best estimate of when the “EXEC” in the execute file will be radiated to the spacecraft for the axial component of the maneuver. For a large maneuver it will probably be necessary to calculate the start time iteratively, using the maneuver midpoint provided by FDF:

1) Run CGS using the maneuver midpoint as the start time
2) Get the duration of the axial and tangential components from the echoxxx.cgs file
3) Rerun CGS using the maneuver midpoint minus half the combined duration of the two components as the start time

The “Vector Decomposition” is provided by CGS to allow the user to review the accuracy of the decomposition of the delta-V vector into axial and tangential components. The spin axis attitude is take from the LP_attitude.dat file. The sun vector is calculated using the SLP data (from mn2000.dat) and the input maneuver start time. The sun angles are measured from the spin axis to the sun vector. To convert these to the Sun Equatorial Angle (SEA) seen in telemetry, subtract the sun angle from 90 (ie, SEA = 90 – sun angle).

The “Axial Maneuver thruster parameters” are provided by CGS to allow the user to allow the user to review the accuracy of the inputs and the maneuver duration calculations.

The “Maneuver Parameter Conversion Summary” provides the conversion of all maneuver parameters from engineering units to hex.

The “Maneuver Uplink Command List” provides the complete command mnemonic list with parameters to be uplinked to the spacecraft, and the LASTCMD confirmation of each mnemonic that will be seen in the downlink telemetry.

For this particular run, the thruster parameter file for the axial component of the maneuver is “axv004.mne”, and the execute file for the axial component is “exc004.mne”.

Here is the contents of the thruster parameter file “axv004.mne”. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:24:08.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt DELAYSUN,    0.0000
mst +00:00:32.0
xlt LONGDUR ,   46.0157
xlt HALFREV,     0.0000
xlt DELAYNUM,       0.0,   1
xlt A12LOI

$$EOS

$$EOF
```

Here is the contents of the execute file “exc004.mne”. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft. Note that the STOPFIRE command is issued approximately 5 seconds after the nominal completion of the maneuver.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:24:08.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS
```

```
$$DATA
xlt EXEC
mst +00:00:51.0
xlt STOPFIRE
mst +00:00:32.0
xlt SETCMDREG
xlt EXEC

$$EOS

$$EOF
```

The “Tangential Maneuver thruster parameters” are provided by CGS to allow the user to allow the user to review the accuracy of the inputs and the maneuver duration and pulse count calculations.

The “Maneuver Parameter Conversion Summary” provides the conversion of all maneuver parameters from engineering units to hex.

The “Maneuver Uplink Command List” provides the complete command mnemonic list with parameters to be uplinked to the spacecraft, and the LASTCMD confirmation of each mnemonic that will be seen in the downlink telemetry.

For this particular run, the tangential component requires more than the maximum 255 pulses allowed per maneuver. Hence CGS has divided the maneuver into two separate thruster parameter files, “tnv005.mne” and “tnv006.mne”, and the corresponding two execute files, “exc005.mne” and “exc006.mne”.

Here is the contents of the thruster parameter file “tnv005.mne”. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:24:08.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt DELAYSUN,    0.1432
mst +00:00:32.0
```

```
xlt SHORTDUR,    0.8330
xlt HALFREV,     1.6670
xlt DELAYNUM,       0.0, 255
xlt T12VEC

$$EOS

$$EOF
```

Here is the contents of the execute file “exc005.mne”. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft. Note that the STOPFIRE command is issued approximately 10 pulse periods after the nominal completion of the maneuver.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:24:08.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt EXEC
mst +00:22:05.0
xlt STOPFIRE
mst +00:00:32.0
xlt SETCMDREG
xlt EXEC

$$EOS

$$EOF
```

Here is the contents of the thruster parameter file “tnv006.mne”. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:24:08.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
```

```
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt DELAYSUN,    0.1432
mst +00:00:32.0
xlt SHORTDUR,    0.8330
xlt HALFREV,     1.6670
xlt DELAYNUM,       0.0, 156
xlt T12VEC

$$EOS

$$EOF
```

Here is the contents of the execute file "exc006.mne". This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft. Note that the STOPFIRE command is issued approximately 10 pulse periods after the nominal completion of the maneuver.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-232T07:24:08.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt EXEC
mst +00:13:50.0
xlt STOPFIRE
mst +00:00:32.0
xlt SETCMDREG
xlt EXEC

$$EOS

$$EOF
```

Using CGS For a Vector Burn Maneuver in the Dark

This section describes how to use CGS for a vector burn maneuver when the spacecraft is in eclipse. It uses as an example an arbitrary Orbit Maintenance Maneuver. The commands for the axial component of a vector maneuver are independent of whether or not the maneuver occurs in the eclipse, so the delta-V right ascension and declination for the example shown here were specifically selected to produce zero axial delta-V.

Here are the contents of the "echo068.cgs" file, which is a complete playback of the interactive session between CGS and the user for this example Orbit Maintenance Maneuver. User inputs are bold and underlined.

```
                    LUNAR PROSPECTOR
              Command Generation Software
                      Version 1.6

         Copyright 1997 Lockheed Martin Corporation
                    All rights reserved



           ***  Main Menu  ***

1. Generate spin rate maneuver commands
2. Generate reorientation maneuver commands
3. Generate vector burn maneuver commands (includes axial delta-V)
4. Quit

Enter selection : 3


           ***  Vector Burn Maneuver  ***

 NOTE: Delta-V right ascension and declination are relative
       to the Mean-of-J2000 ecliptic reference frame

 Enter delta-V magnitude (m/sec)                   :    3.000
 Enter right ascension of the delta-V vector (deg) :   90.000
 Enter declination of the delta-V vector (deg)     :    0.000
 Enter tangential thruster pulse width (sec)       :    0.833
 Enter current tank pressure (psia)                :  140.000
 Enter current average tank temperature (deg C)    :   20.000
 Enter estimated propellant mass remaining (kg)    :   30.000
 Will maneuver be executed in the dark (y/n)?      : y
 Enter light delay from ground to spacecraft (sec) :    1.330
 Enter reference sunpulse time (yymmdd.hhmmssf UTC): 971029.015814


 -------------------  Vector Decomposition  -------------------

    Spin axis right ascension (J2000 ecliptic frame) =    0.00 deg
    Spin axis declination     (J2000 ecliptic frame) =   90.00 deg

    Spin axis attitude (J2000 ecliptic frame) =   0.000000  0.000000  1.000000
    Sun vector         (J2000 ecliptic frame) =  -0.811365 -0.584540 -0.000166
    Delta-V vector     (J2000 ecliptic frame) =   0.000000  1.000000  0.000002
```

```
   Spin axis to sun     angle   =   90.0095 deg
   Spin axis to delta-V angle   =   89.9999 deg
   Delta-V axial      component =    0.0000 m/s
   Delta-V tangential component =    3.0000 m/s


----------  Axial Maneuver thruster parameters  ----------

   S/C dry mass                      =  167.000 kg
   Prop mass at start of burn        =   30.000 kg
   S/C mass at start of burn         =  197.000 kg

   Axial delta-V magnitude           =    0.000 m/s
   Burn duration                     =      0.0 sec
   Average axial thrust              =   19.997 N  (includes cant angle loss)
   Average Isp                       =  172.484 sec
   Propellant consumed               =    0.000 kg

   Tank pressure at start of burn    =   140.00 psia
   Tank pressure at end of burn      =   138.33 psia
   REA feed pressure at start of burn=   137.79 psia
   REA feed pressure at end of burn  =   136.18 psia

   A3/A4 on-time limit               =     44.8 sec

   ### THRUSTER A1 ###
       Cant angle                    =   10.000 deg
       Thrust at start of burn       =   10.317 N
       Thrust at end of burn         =   10.237 N
       Isp at start of burn          =  177.574 sec
       Isp at end of burn            =  177.305 sec

   ### THRUSTER A2 ###
       Cant angle                    =   10.000 deg
       Thrust at start of burn       =   10.158 N
       Thrust at end of burn         =   10.068 N
       Isp at start of burn          =  167.831 sec
       Isp at end of burn            =  167.662 sec


!!! NO AXIAL BURN REQUIRED !!!  Axial burn duration less than 0.050 sec


----------  Tangential Maneuver thruster parameters  ----------

   S/C dry mass                      =  167.000 kg
   Prop mass at start of burn        =   30.000 kg
   S/C mass at start of burn         =  197.000 kg

   Tangential delta-V magnitude      =    3.000 m/s
   Tangential delta-V heading angle  = -125.771 deg (wrt sun pulse)
   Spin rate                         =   12.000 rpm   (  72.000 deg/sec)
   Number of pulses                  =       26
   Cumulative pulse time             =   21.346 sec
   Total maneuver time               =  130.000 sec

   Pulse efficiency factor           = 0.954965
   Average tangential thrust         =   28.972 N
   Average Isp                       =  222.950 sec
   Average centroid                  =    0.554 (fraction of pulse)
   Propellant consumed               =    0.283 kg

   Tank pressure at start of burn    =   140.00 psia
   Tank pressure at end of burn      =   138.08 psia
   REA feed pressure at start of burn=   137.79 psia
   REA feed pressure at end of burn  =   135.93 psia

   ### THRUSTER T1 ###
       Azimuth angle                 =  180.000 deg
       Thrust at start of burn       =   14.744 N
       Thrust at end of burn         =   14.576 N
       Isp at start of burn          =  226.532 sec
       Isp at end of burn            =  226.266 sec
       Centroid at start of burn     =  0.55027 (fraction of pulse)
       Centroid at end of burn       =  0.55045 (fraction of pulse)

   ### THRUSTER T2 ###
       Azimuth angle                 =  180.000 deg
       Thrust at start of burn       =   14.903 N
       Thrust at end of burn         =   14.732 N
       Isp at start of burn          =  223.590 sec
       Isp at end of burn            =  223.239 sec
       Centroid at start of burn     =  0.55584 (fraction of pulse)
       Centroid at end of burn       =  0.55610 (fraction of pulse)


--------------  Maneuver Parameter Conversion Summary  --------------
SUN PULSE DELAY    =    2.7917 (sec)  =  5025 (counts)  =  13A1 (hex)
BURN DURATION      =    0.8330 (sec)  =  1499 (counts)  =  05DB (hex)
HALFREV TIME       =    1.6670 (sec)  =  3001 (counts)  =  0BB9 (hex)
EXECUTE DELAY      =      0.00 (min)  =     0 (counts)  =    00 (hex)
NUMBER OF PULSES   =                       26 (counts)  =    1A (hex)
THRUSTER COMMAND   =   T12VEC


---------  Maneuver Uplink Command List  ---------
DELAYSUN    2.7917         Confirmation  =  3113A1
SHORTDUR    0.8330         Confirmation  =  3205DB
HALFREV     1.6670         Confirmation  =  330BB9
DELAYNUM    0.00   26      Confirmation  =  30001A
T12VEC                     Confirmation  =  3B004E


MGDS  script file for this maneuver is :  tnv068.mne
```

Most of the input parameters are described in the previous section. The differences occur after the user responded “yes” to the question “Will maneuver be executed in the dark (y/n)?”.

The light delay is the one-way light travel time from the ground station to the spacecraft. This information can be obtained from DSN.

The reference sunpulse time is the UTC time at which the spacecraft saw its last valid sun pulse. This is the last sun pulse before the spacecraft enters penumbra, not umbra. This pulse time is calculated by from the ground receipt time of the sun pulse as follows:

reference sunpulse = ground receipt time stamp – light delay – 2 seconds – delta

where delta = MOD(2*VCDU – SSTIM + 32, 32)

VCDU = frame counter (from telemetry)
SSTIM = sun pulse time tag (from telemetry)

Unlike all other maneuvers, tangential maneuvers in the dark have only one MGDS file, which includes both the parameters and the execute. This is necessary because the ground must load the parameters and command the execute rapidly and very precisely in time.

Here is the contents of the file “tnv068.mne” for this example. This is a complete MGDS script, which MGDS will translate into bit patterns to be radiated to the spacecraft.

```
CCSD3ZF0000100000001NJPL3KS0L015$CMDMNE$
MISSION_NAME = LUNAR_PROSPECTOR;
MISSION_ID = 18;
SPACECRAFT_NAME = LUNAR_PROSPECTOR;
SPACECRAFT_ID = 25;
DATA_SET_ID = CMD_MNE;
PRODUCER_FULL_NAME = 'Lunar Prospector Command Generation Software v1.6';
PRODUCT_CREATION_TIME = 1997-238T16:07:29.000;
DATA_TYPE = TELECOMMAND;
CCSD$$MARKER$CMDMNE$NJPL3IF0037600000001
$$HEADER
BIT1_RADIATION_TIME=1997-302T01:57:58.0
ACQUISITION_SEQ=8,AA
CLTU_START_SEQ=2,EB90
CLTU_TAIL_SEQ=5,C5C5C5C579
$$EOS

$$DATA
xlt DELAYSUN,    0.0000
mst +00:00:01.0
xlt SHORTDUR,    0.8330
xlt HALFREV,     1.6670
xlt DELAYNUM,       0.0,   1
mst 1997-302T01:58:12.3
xlt T12VEC
mst 1997-302T01:58:14.3
xlt EXEC
mst 1997-302T01:58:17.3
xlt T12VEC
mst 1997-302T01:58:19.3
xlt EXEC
mst 1997-302T01:58:22.3
xlt T12VEC
mst 1997-302T01:58:24.3
xlt EXEC
mst 1997-302T01:58:27.3
xlt T12VEC
mst 1997-302T01:58:29.3
xlt EXEC
mst 1997-302T01:58:32.3
xlt T12VEC
mst 1997-302T01:58:34.3
xlt EXEC
mst 1997-302T01:58:37.3
xlt T12VEC
```

```
mst 1997-302T01:58:39.3
xlt EXEC
mst 1997-302T01:58:42.3
xlt T12VEC
mst 1997-302T01:58:44.3
xlt EXEC
mst 1997-302T01:58:47.3
xlt T12VEC
mst 1997-302T01:58:49.3
xlt EXEC
mst 1997-302T01:58:52.3
xlt T12VEC
mst 1997-302T01:58:54.3
xlt EXEC
mst 1997-302T01:58:57.3
xlt T12VEC
mst 1997-302T01:58:59.3
xlt EXEC
mst 1997-302T01:59:02.3
xlt T12VEC
mst 1997-302T01:59:04.3
xlt EXEC
mst 1997-302T01:59:07.3
xlt T12VEC
mst 1997-302T01:59:09.3
xlt EXEC
mst 1997-302T01:59:12.3
xlt T12VEC
mst 1997-302T01:59:14.3
xlt EXEC
mst 1997-302T01:59:17.3
xlt T12VEC
mst 1997-302T01:59:19.3
xlt EXEC
mst 1997-302T01:59:22.3
xlt T12VEC
mst 1997-302T01:59:24.3
xlt EXEC
mst 1997-302T01:59:27.3
xlt T12VEC
mst 1997-302T01:59:29.3
xlt EXEC
mst 1997-302T01:59:32.3
xlt T12VEC
mst 1997-302T01:59:34.3
xlt EXEC
mst 1997-302T01:59:37.3
xlt T12VEC
mst 1997-302T01:59:39.3
xlt EXEC
mst 1997-302T01:59:42.3
xlt T12VEC
mst 1997-302T01:59:44.3
xlt EXEC
mst 1997-302T01:59:47.3
xlt T12VEC
mst 1997-302T01:59:49.3
xlt EXEC
mst 1997-302T01:59:52.3
xlt T12VEC
mst 1997-302T01:59:54.3
xlt EXEC
mst 1997-302T01:59:57.3
xlt T12VEC
mst 1997-302T01:59:59.3
```

```
xlt EXEC
mst 1997-302T02:00:02.3
xlt T12VEC
mst 1997-302T02:00:04.3
xlt EXEC
mst 1997-302T02:00:07.3
xlt T12VEC
mst 1997-302T02:00:09.3
xlt EXEC
mst 1997-302T02:00:12.3
xlt T12VEC
mst 1997-302T02:00:14.3
xlt EXEC
mst 1997-302T02:00:17.3
xlt T12VEC
mst 1997-302T02:00:19.3
xlt EXEC

$$EOS

$$EOF
```

12 CmdX PROGRAMMER'S GUIDE

The C application program CmdX was developed by LMMS to package the command selected by the operator at the OASIS console into the format required by the spacecraft. CmdX performs the required BCH encoding of the command data. CmdX then assembles the uplink command into a 200 bit serial stream which is transmitted to st_vme_up, which in turn effects the uplink command transmission via the VME hardware.

CmdX calls the function BCH_Encode, contained in the file bch.c, which performs BCH encoding per the required CCSDS Telecommand (TC) Codeblock Encoding Procedure. The encoding algorithm is a modified Bose-Chaudhuri-Hocquenghem (BCH) code which generates 7 parity check bits from 32 TC Codeblock bits. Reference the Lunar Prospector Mission Ground System Software Requirements & Design, LMSC P304S001 for additional information.

CmdX calls the socket functions contained in socks.c for reading and writing data over socket interfaces.

The included files are socks.h, which contains the function prototype definitions for the socket functions, and ccsds.h, which contains general purpose type definitions, global declarations, and function prototype definitions used by cmdx.c.

CmdX has three socket interfaces:

1. CmdX is a server to st_vme_up for writing, i.e. transmitting the assembled, serial uplink command to st_vme_up

2. CmdX is a client to OASIS for writing, i.e. echoing back the required command information to OASIS for command verification by the operator via comparison with telemetry

3. CmdX is a client to OASIS for reading, i.e. receiving the OASIS command information

13 st_vme_up PROGRAMMER'S GUIDE

The C application program st_vme_up was developed by LMMS to affect the uplink command transmission via the VME hardware. In addition, st_vme_up reads the IRIG-B time registers and writes the command and time into a log file for later analysis.

Operation of the VME hardware is achieved by implementing the pseudocode provided in section 4.2.2 of the Avtec Systems' DQMFS and PCMSIM Firmware manual, dated 12/22/94. PCMSIM is the firmware component that supports the serial output function of the FSIO2 VME card. The program is essentially composed of two parts. In the main procedure, all setup tasks required to set up the uplink hardware are performed by utilizing the National Instruments "VXI for VME" functions (reference the comments in the code listing of st_vme_up.c). Next, the socket interface with CmdX is opened and the VME_Uplink_Func function is called. VME_Uplink_Func is essentially an infinite loop to read, process, and send commands via the VME hardware. Consult the code listing comments for additional information.

st_vme_up calls the socket functions contained in socks.c for reading and writing data over socket interfaces.

The included files are socks.h, which contains the function prototype definitions for the socket functions, and /opt/NICsbmxi/include/nivxi.h, a header file containing type definitions and function prototype definitions for the NI-VXI library function calls.

st_vme_up has one socket interface:

1. st_vme_up is a client to CmdX for reading, i.e. receiving the assembled, serial uplink command

See section 14.1 for reference material pertinent to the st_vme_up application program.

14 st_vme_down PROGRAMMER'S GUIDE

The C application program st_vme_down was developed by LMMS to read the downlink telemetry data via the VME hardware. st_vme_down appends the IRIG-B time prior to transmission to FrameX, in order to make the interface with FrameX identical to the flight configuration - in this manner the same version of FrameX can be used for system test as well as operations.

Operation of the VME hardware is achieved by implementing the pseudocode provided in section 3.2.2 of the Avtec Systems' DQMFS and PCMSIM Firmware manual, dated 12/22/94. DQMFS is the firmware component that supports the serial input function of the FSIO2 VME card. The program is essentially made up of two parts. In the main procedure, all setup tasks required to set up the downlink hardware are performed by utilizing the National Instruments "VXI for VME" functions (reference the comments in the code listing of st_vme_down.c). Next, the socket interface with FrameX is opened and the VME_Downlink_Func function is called. VME_Downlink_Func is essentially an infinite loop to read downlink telemetry data via the VME hardware, read the IRIG-B time, process the data and time into the format required by FrameX, display the data and time in a text-based format, and send the data and time to FrameX over the socket. Consult the code listing comments for additional information.

st_vme_down calls the socket functions contained in socks.c for reading and writing data over socket interfaces.

The included files are socks.h, which contains the function prototype definitions for the socket functions, and /opt/NICsbmxi/include/nivxi.h, a header file containing type definitions and function prototype definitions for the NI-VXI library function calls.

st_vme_down has one socket interface:

1. st_vme_up is a server to FrameX for writing, i.e. transmitting the downlink data and time to FrameX

See section 14.1 for reference material pertinent to the st_vme_down application program.

14.1 References for st_vme_up and st_vme_down Application Programs

1. Avtec Systems, Frame Synchronizer (DQMFS) and PCM Simulator (PCMSIM) Firmware, Programmer's Reference Guide, December 22, 1994

2. Avtec Systems, TM-CRC / TM-SYNTH / TM-CRC2 Transition Module Functional Description, Version 1.1, March 1994

3. Avtec Systems, Frame Synchronizer / Telemetry Output Interface Board (FSIO2) Functional Description, Version 1.1, January 12, 1994

4. National Instruments, Getting Started with Your VME-SB2020 and the NI-VXI Software for Solaris, February 1994 Edition, Part Number 320338-01

5. National Instruments, NI-VXI Text Utilities Reference Manual, October 1993 Edition, Part Number 320321-01

6. National Instruments, NI-VXI C Software Reference Manual for VME, November 1993 Edition, Part Number 320389-01

7. Datum Inc., bc635VME / bc350VXI Time and Frequency Processor User's Guide, August, 1996

15 FrameX PROGRAMMERS'S GUIDE

The C application program FrameX was developed by LMMS to decode downlink telemetry data. FrameX validates the downlink telemetry frame, archives the downlink data, and forwards the telemetry data to OASIS for further processing and display to the operator.

FrameX reads one byte at a time from the st_vme_down socket looking for the EB90 (hex) sync pattern at the beginning of the frame. Once the sync has been found, FrameX then reads the remaining 890 data bytes, 6 time tag bytes and 8 bytes of ground station data from the socket. FrameX copies the raw incoming telemetry data to a binary log file. The log file name is either provided by the user at startup, or is created by default by combining the current date and time. The original frame is then split into a packet containing prior data and a packet containing delayed data. Minor changes are made to the new frame headers in order to reduce the processing load on OASIS. More specifically, the spacecraft ID, VCID, and telemetry rate ID are replaced by a packet ID and packet length. Both 470 byte long packets are finally forwarded to OASIS and are also copied to binary files.

FrameX calls the socket functions contained in socks.c for reading and writing data over socket interfaces.

FrameX has two socket interfaces:

1. FrameX is a server to OASIS for writing, i.e. transmitting the prior and delayed science and engineering packets to OASIS

2. FrameX is a client to st_vme_down for reading, i.e. receiving the downlink data and time from st_vme_down

3. In Mission Operations, FrameX is also a client to OASIS for the purpose of receiving a segment command. The user can send the segment command "CHANGE FRAMEX" which will close the current raw binary file and open a new file.

16 VES PROGRAMMER’S GUIDE

16.1 Overview

The Lunar Prospector Vehicle and Environment Simulator (VES) is a software training tool for mission operations personnel designed to provide an uncomplicated yet realistic simulation of spacecraft and orbital dynamics as well as command and telemetry functionality. In particular, the VES comprises three software modules. The command and data handling simulation (C&DHS) is written in C and models the decoding and encoding of spacecraft telemetry and command streams and performs error checking and command. The vehicle and orbital dynamics simulation (VODS) is implemented in FORTRAN and models spacecraft orbit and attitude based on initial conditions and uplink commands (e.g. thruster firings). VODS also models the thruster catalyst bed temperatures as well as the outputs of the sun sensor and earth moon limb crossing sensor. An independent software entity named TlmMod was developed using C++ and the class framework supplied by zApp to fulfill the VES requirement that anomalous data be inserted in the downlink telemetry stream for operator training. TlmMod provides the capability to view and modify telemetry downlink parameters in real time.

C&DHS, VODS, and TlmMod comprise the VES and are integrated into the remainder of the ground software system as illustrated in Figure 16.1-1.

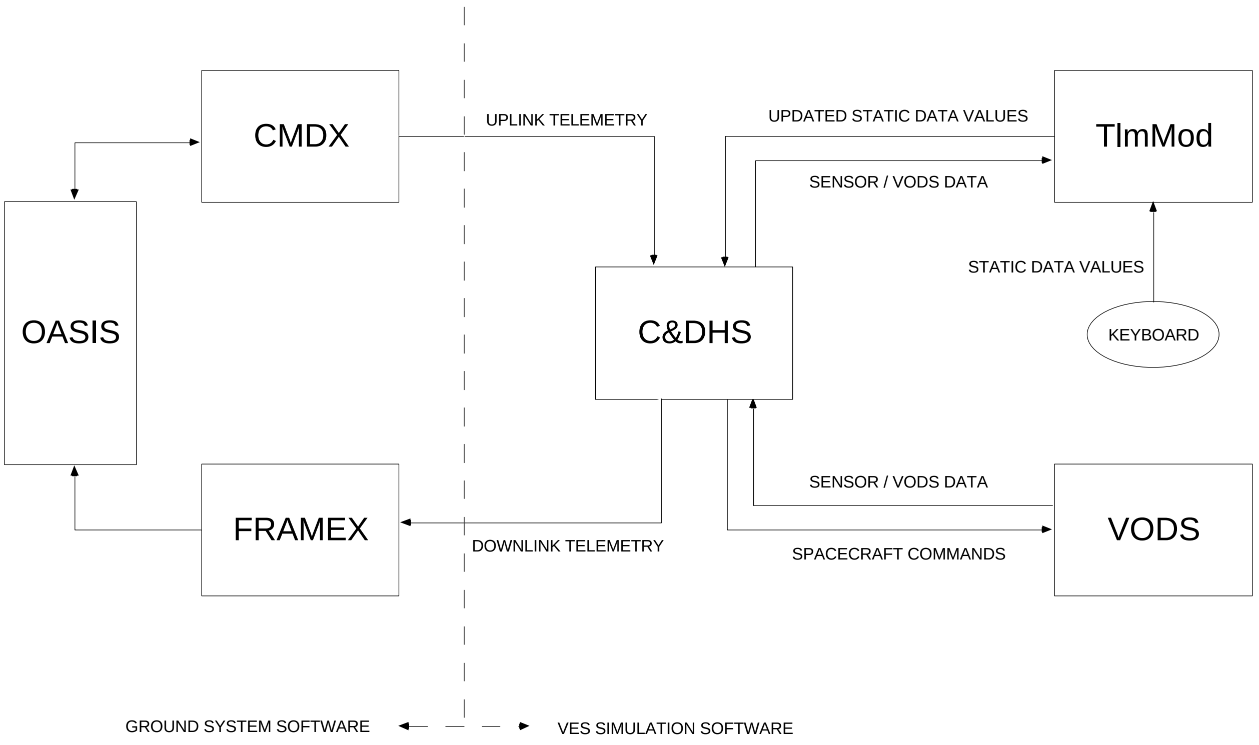


Figure 16.1-1 Ground Software Architecture

16.2 Interfaces

Since the VES substitutes for the actual spacecraft hardware during training, it must support all of the normal spacecraft command and telemetry interfaces. These include the asynchronous command uplink interface (i.e. CmdX to C&DHS) and the synchronous telemetry downlink (i.e. C&DHS to FrameX), both of which use TCP / IP Ethernet sockets for data transfer. Unique to the VES are the interfaces to the VODS and TlmMod. Information is exchanged between the C&DHS and VODS software using a standard function call, while data transfer between the C&DHS and TlmMod occurs via sockets.

16.3 Timing

The core of the VES software is a timing routine contained in the C&DHS which controls the real-time execution of the telemetry downlink and the VODS calculations, as well as the data transfer between the C&DHS and TlmMod.

The timer is implemented as a signal handler in conjunction with a UNIX itimer having a pulse interval of 100 milliseconds. The VODS is called by the signal handler every timer pulse to ensure accurate dynamics calculations. The routine to encode and transmit downlink telemetry is called at two second intervals (every 20 timer pulses). Similarly, the data transfer to and from TlmMod also occurs at two second intervals. The sequence of events within the signal handler is shown in Figure 14.3-1.

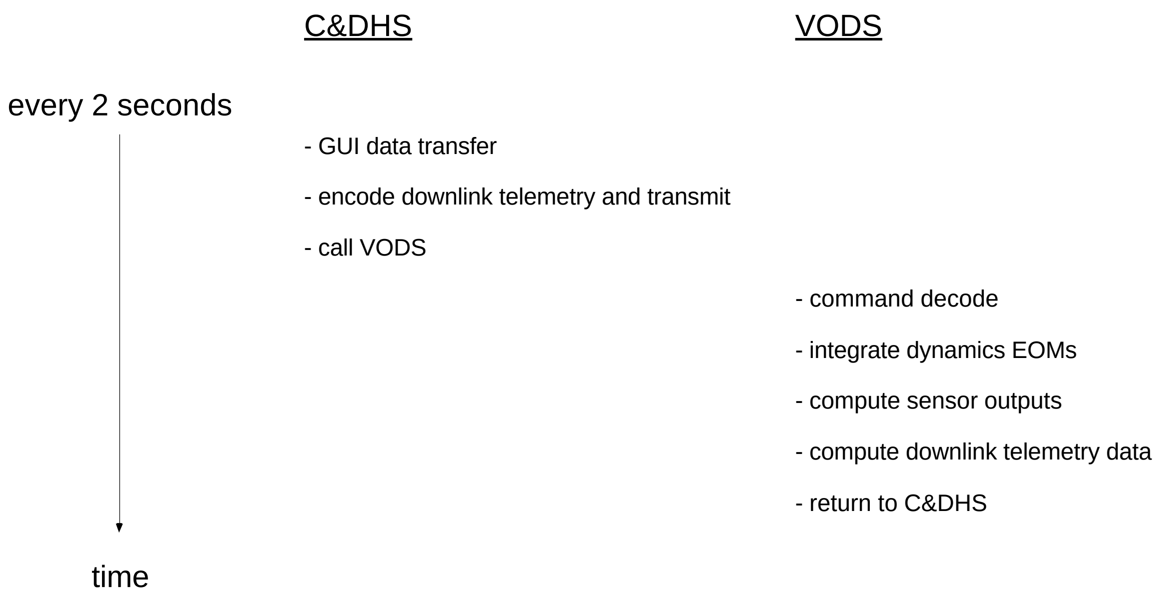


Figure 16.3-1 C&DHS Signal Handler Timing Sequence

The asynchronous processing of uplink commands is performed by the main program using a blocking socket. Therefore, during the majority of the time which the VES is running, the main program loop is waiting at the CmdX socket read waiting for command data while the signal handler is executing the downlink telemetry and TlmMod functions every 100 milliseconds.

TlmMod is implemented as an event driven process, with zApp zTimer class objects providing the necessary real-time capability to synchronize the transfer of information to and from the C&DHS. Data transfer occurs at 2 second intervals. Also, every open data visualization window has its own timer to update the parameter's value every two seconds.

16.4 Data Structures

In order to facilitate the transfer of information among various software entities, several global data structures are defined. In particular, both the C&DHS and TlmMod use a two dimensional array to store the contents and status of all downlink telemetry parameters. The type definitions listed in Figure 16.4-1 are contained in the C&DHS file "typedef.h" and in the TlmMod file "guidefs.h."

```
typedef enum {orig, tlm, persist} numidx;

typedef enum {
  A1TMP, A2TMP, A3TMP, A4TMP, APSM25V, BATCUR, BATTMP, BATVLT, BLI1, BLI10,
  BLI11, BLI12, BLI2, BLI3, BLI4, BLI5, BLI6, BLI7, BLI8, BLI9, BURNDUR_msb,
  BURNDUR_lsb, BUSVLT, CCBTMP, CDHTMP, CHKSUM, CTLRST, DAMPTMP, DELAYSUN_msb,
  DELAYSUN_lsb, DWNLINKST, EMDETTMP, EMELXTMP, EMSTHR, EMTHRESHLEV_msb,
  EMTHRESHLEV_lsb, EMTIM1_msb, EMTIM1_lsb, EMTIM2_msb, EMTIM2_lsb, EMTIM3_msb,
  EMTIM3_lsb, EMTIM4_msb, EMTIM4_lsb, EMTIM5_msb, EMTIM5_lsb, EMTIM6_msb,
  EMTIM6_lsb, ER12V, ER28V, ER5V, ERHTRPWR, ERHVCUR, ERHVVLT, ERM5V, ERSWPCUR,
  ERSWPVLT, ERTMP, FIREDELAY, FRM_SEQ_NUM, GRSHV1, GRSHV2, GRSTMP, HALFREV_msb,
  HALFREV_lsb, LASTCMD_msb, LASTCMD, LASTCMD_lsb, LDCUR, LOOPSTR, MAGHTRPWR,
  MAGTMP, MER12V, MER5V, MERHTRPWR, MERTMP, NSAPSTMP, NSHV1, NSHV2, NUMTIM,
  RCVSECVLT, RCVSIGSTR, SA1TMP, SA2TMP, SA3TMP, SACUR, SAI_10, SAI_9, SES12V,
  SES28VCUR, SES5V, SESTMP, SSANG, SSBIASV, SSCOSV, SSSINV, SSTIM_msb,
  SSTIM_lsb, STRCUR, T1TMP, T2TMP, TANK1TMP, TANK2TMP, TANK3TMP, TANKPRESS,
  UPLKST, V15, V5, VCID, VM15, VTERR, XMTPWROUT, XMTSECV12, XMTSECV20, XMTTMP
  } eng_enum;
```

Figure 16.4-1 VES Global Type Declarations

The global variable declarations specific to the C&DHS are listed in Figure 16.4-2. The three integer variables are used by various functions to indicate the status of the current telemetry mode and uplinked command. The two unsigned char variables are used to initialize the downlink telemetry arrays with default values of engineering and science data. The unsigned char array uses the type definitions from Figure 16.4-2 to represent the 107 bytes which comprise the sum total of the uncommutated telemetry parameters. For the short dimension, "orig" reflects the true telemetry data computed by the C&DHS and VODS. The contents of the "tlm" dimension are the combination of "orig" data and any modified values entered through TlmMod. It is the contents of the "tlm" dimension of the "eng_data" array which are output to FrameX. The "persist" dimension contains a flag set by TlmMod to tell the C&DHS that the modified value should be inserted in the downlink telemetry stream.

```
int execute = FALSE, tlm_rate = ENG_RATE_ID, cmd_reject = FALSE;
unsigned char eng_default, sci_default;
unsigned char eng_data[persist+1][XMTTMP+1];
```

Figure 16.4-2 C&DHS Global Variable Declarations

The global variable declarations specific to TlmMod are listed in the Figure 16.4-3. The downlink telemetry buffer named "numdata" is identical in dimension to the corresponding "eng_data" array in the C&DHS. For the short dimension, "orig" reflects the true telemetry data computed by the C&DHS and VODS. The contents of the "tlm" dimension represent the modified values entered through TlmMod. The "persist" dimension contains a flag set by TlmMod to tell the C&DHS that the modified value should overwrite the existing value in the downlink telemetry stream. The "tlmNames" array contains the names (i.e. mnemonics) of all 94 downlink telemetry parameters and is used to create captions for the data input windows. The “tlmItems” array contains the mnemonic, description, data type, output format, and other information for each of the 94 downlink telemetry parameters. The “calItems” array contains the mnemonic, calibration type, numerical coefficients, and other information pertaining to the data number to engineering unit (dn-to-eu) calibrations performed on each telemetry parameter by TlmMod.

```
unsigned char numdata[persist+1][XMTTMP+1];

typedef char tlmName[16];
typedef enum {Binary, Octal, Hexadecimal, Integer, Float, Exponential, Text}
enumFormat;
typedef enum {unsegmented_exp, unsegmented_5d, segmented_3d} enumCalType;
typedef enum {Short_Analog, Long_Analog, Command_Type} enumDataType;
typedef enum {F, T} enumFT;

typedef struct {
  tlmName mnemonic;
  int calIndex;
  int dataIndex;
  char description[48];
  tlmName displayUnits;
  enumDataType dataType;
  enumFormat format;
} tlmItem;

typedef struct {
  tlmName mnemonic;
  enumCalType calType;
  int segmentBound;
  enumFT lastSegment;
  float c0, c1, c2, c3, c4, c5;
} calItem;

char* tlmNames[200];
tlmItem tlmItems[200];
calItem calItems[200];
```

Figure 16.4-3 TlmMod Global Variable Declarations

16.5 Software Modules

The VES software source code is broken up into various files for ease of organization and debugging. Figure 16.5-1 describes the filenames and their contents. Since TlmMod code was developed using the zApp zFactory application builder, numerous other files in addition to those listed below are required to create the application. However, these are mostly window, menu, and dialog resource files created automatically by zFactory and do not contain any functionality specific to the VES.

FILENAME	DESCRIPTION
CDH	

bch.c	bch error code calculation
cdh.c (main)	initializations, timer functions
cdhgui.c	data transfer over TlmMod socket, manage true/modified telemetry data
define.h	#defines for C&DHS
init.c	reads initial telemetry values from file
init.dat	contains initial telemetry values
socks.c	library of socket functions
tel_dec.c	accepts and processes command data from CmdX socket
tel_enc.c	encodes telemetry data and transmits to FrameX socket
typedef.h	global type definitions
ves.c	routes commands, provides interface to VODS
vods.f	dummy VODS routine
TlmMod	
caltable.cxx	reads channel calibration and display data from initialization file
guidefs.h	global type, variable declarations for tlmmod.cxx
socks.cxx	library of socket functions
socks.h	#includes and #defines for socket library
socks.hpp	class, variable definitions for socks.cxx
tlmmod.cxx (main)	window routines, user data input, data transfer over TlmMod socket
tlmmod.hpp	class, variable definitions for tlmmod.cxx
ves_cal.out	initialization file of telemetry monitor calibrations (SYBASE dump)
xref.dat	initialization file relating CDH and TlmMod telemetry data structures

Figure 16.5-1 VES Software Source Files and Descriptions

bch.c

<u>unsigned char BCH_Encode(unsigned char *Input)</u>

This piece of code takes 32 bits starting at the address pointed to by Input, computes the eight bit BCH checksum as its return value. See the "CmdX Programmer's Guide" section above for a reference to the BCH algorithm.

cdh.c

<u>void alarm_handler()</u>

This is the routine which controls the execution of the synchronous functions of the VES. It is called every 100 milliseconds as a result of the itimer initialized in main().

Upon receipt of an interrupt, the global variable "pulse" is incremented. This variable rolls over upon reaching 20 which amounts to two seconds in real time. If a specific pulse is a two second pulse, the global variable "counter" is incremented. This corresponds to the VCDU counter on board the spacecraft and rolls over upon reaching 2^{24} or roughly a year in real time. In order to achieve accurate dynamics results, the function "VesSimulation" is called every 100 milliseconds. At two second intervals the functions "GuiTransfer" and "EncodeTelemetry" are called following "VesSimulation" to provide the required downlink telemetry and operator intervention capabilities.

main()

This is the main program which is responsible for starting up and running the C&DHS and VODS portions of the VES. First, the file "init.dat" is read and the contents used as the initial values of the downlink telemetry stream. Next, a client socket is created which connects with CmdX to allow uplink commands to be sent to the C&DHS. A server socket is then created which is used to send telemetry to FrameX. An itimer is initialized to create a SIGALRM interrupt and call the "alarm_handler" interrupt handler every 100 milliseconds. In order to process the command data from CmdX, the function "DecodeTelemetry" is called. Only in case of error does the program return from "DecodeTelemetry" in which case the CmdX and FrameX sockets are closed and the itimer is switched off before exiting.

cdhgui.c

int OpenGuiSocketAsServer()

This function opens a socket used to transfer data between the C&DHS and TlmMod. The socket, bind, and listen calls are made here, but the accept call must be made in the calling function in order to support non-blocking sockets. Upon successful completion, the function returns TRUE (1), otherwise an error message is printed to the screen and the application terminates.

void CloseGuiSocket()

This function closes the socket used to transfer data between the C&DHS and TlmMod.

int GuiTransfer(int counter)

This function is called every two seconds by "alarm_handler." Its purpose is to write current telemetry data to TlmMod, receive user-modified data from TlmMod, and update the downlink telemetry stream to reflect any changes.

First, the default action is to copy the current (true) telemetry data into the downlink telemetry buffer. In order that TlmMod may be started up and shut down without affecting the operation of the VES, the function next performs one of the following three actions.

1. If the socket has not yet been bound, the “OpenGuiSocketAsServer” function is called.

2. If the socket has been bound but the client is not yet present, the accept call is repeated in an attempt to complete the socket connection.

3. If the client has been successfully connected, current telemetry data is written to TlmMod. A failed socket write indicates that TlmMod client has been terminated in which case the server socket is also shut down. Next, the modified telemetry data is read from TlmMod and copied into the downlink telemetry buffer.

The process by which true (C&DHS) or modified (TlmMod) data is copied into the telemetry buffer deserves more explanation. The C&DHS sends its current true information to TlmMod every two seconds whenever the socket is operational. Likewise, the C&DHS receives a stream of modified data and flags from TlmMod every two seconds. The C&DHS then checks each telemetry parameter to see whether TlmMod has sent a modified value. If so, it inserts the modified value in the downlink buffer, and if not it inserts the current true value in the downlink buffer. The data path through TlmMod cannot supply both true and modified values because of timing delays. More specifically, the delay time between the transfer of true data to TlmMod and the time the combination of true and modified data returns from TlmMod would be between two and four seconds. To avoid this delay, the true C&DHS data (i.e. not passed through TlmMod) is used in the telemetry stream along with the modified data from TlmMod.

define.h

This file contains various #defines for global constants.

init.c

<u>void InitTelemetry()</u>

This routine is called by “main” during VES startup in order to initialize the contents of the downlink telemetry buffer. The file “init.dat” is opened and read with the assumption that the ordering of the “parameter description, hex value” pairs is equivalent to the order of the enum type “eng_col_enum” defined in “typedef.h.”

init.dat

This file contains “parameter description, hex value” pairs read by “InitTelemetry” to initialize the downlink telemetry buffer.

socks.c

<u>int write_n_bytes(int socket_fd, int n, unsigned char *data)</u>

This function writes “n” bytes of data beginning at the location specified by the pointer “data” to the socket identified by “socket_fd.” If the function terminates successfully, the return value is the number of bytes written. If the write fails, a message is printed to the screen with the error code from the system file “errno.h”, and the value ERROR (-1) is returned to the calling function. Since the VES application makes use of itimer interrupts, this function is designed to retry the socket write and not return with error in the event that it is interrupted by the itimer. Also, to avoid having the application be terminated if the receiving socket process quits, the function ignores the SIGPIPE interrupt.

<u>int read_n_bytes(int socket_fd, int n, unsigned char *data)</u>

This function reads “n” bytes of data from the socket identified by “socket_fd” and stores the information beginning at the location specified by the pointer “data.” If the function terminates successfully, the return value is the number of bytes read. If the read fails, a message is printed to the screen with the error code from the system file “errno.h”, and the value ERROR (-1) is returned to the calling function. Since the VES application makes use of itimer interrupts, this function is designed to retry the socket read and not return with error in the event that it is interrupted by the itimer.

tel_dec.c

<u>int OpenCmdxSocketAsClient()</u>

This function opens a client socket so that the C&DHS can receive uplink commands from the CmdX server. Upon successful completion, the function returns TRUE (1). Otherwise an error message is printed to the screen and the application terminates.

<u>static int FindHeader()</u>

This function is called by “DecodeTelemetry” and searches the uplink command data stream byte by byte for the hexadecimal sync pattern “EB90.” When successful, this function returns the value of the sync word. Otherwise, an error message is printed to the screen and the value FALSE (0) is returned.

<u>int DecodeTelemetry()</u>

This function is called by “main” and is responsible for accepting and validating uplink commands from CmdX. First, “FindHeader” is called to search for the sync pattern. If unsuccessful, the function terminates and the value FALSE (0) is returned. Otherwise, three five byte blocks are read from the CmdX socket. BCH checksums are computed (see “BCH_Encode”) for the first four bytes of the first two blocks and compared with the checksum already present as the fifth byte of each block. If the checksums do not match, the function prints an error message, returns FALSE (0), and terminates. Otherwise, the contents of the third block (tail sequence) are checked against the fixed hexadecimal bit pattern “C5C5C5C579.” If not identical, the function prints an error message, returns FALSE (0), and terminates. Otherwise, various other telemetry bits are checked to see if they match the required fixed values. If there is a discrepancy, the downlink telemetry command reject flag is set to TRUE (1), otherwise it is set to FALSE (0). Next, if the VCID and command bit pattern match those of the “execute” command and the command has not been rejected, the execute flag is set to TRUE (1). If the command is valid but not “execute,” the frame sequence number, VCID, and 24 bit command sequence are copied into the global engineering data array “eng_data.” If the command does not require an “execute” in order to be carried out, the action is performed immediately. Since this function comprises an infinite loop and does not exit unless an error occurs, there is no return value for successful completion.

<u>void CloseCmdxSocket()</u>

This function closes the socket used to transfer data between CmdX and the C&DHS.

tel_enc.c

<u>int OpenFramexSocketAsServer()</u>

This function opens a server socket so that the C&DHS can transmit downlink telemetry to the FrameX client. Upon successful completion, the function returns TRUE (1). Otherwise an error message is printed to the screen and the application terminates.

<u>int EncodeTelemetry(int counter)</u>

This function is called by the timing function “alarm_handler” and builds and transmits the appropriate telemetry frame to the FrameX socket. In order to support both engineering and science telemetry modes, the telemetry information is structured as low rate engineering data, high rate engineering data, and science data. The low rate engineering data contains the health and status information of the satellite and is transmitted in both telemetry modes, while the high rate engineering data and science data contain spare bytes and science instrument data not simulated by the VES. Thus, the simulation accurately models the low rate engineering data, but simply fills the high rate engineering and science data with the default values specified in “init.dat.”

The contents of the telemetry frame are determined by the telemetry mode (i.e. engineering or science), default telemetry values, uplink command data, the results of the VODS sensor outputs, and user-modified data from TlmMod. The frame contents are also a function of the mainframe counter which provides commutation of the data.

In order to properly construct each telemetry frame, the function uses a state machine approach to insert the relevant information at the appropriate time. For science data, all information is inserted every two seconds, so no state transitions are required. However, since a telemetry frame in engineering mode contains data over 12 two second periods, the state must have knowledge of how far along it is in the sequence and when to close the frame and send it to FrameX. One difference in the operation of the VES compared with the actual hardware is that the simulation sends bursts of information to FrameX every two seconds while the C&DH hardware continually transmits information at either 300 or 3600 bps. Since FrameX waits to accept the entire frame before passing it on to OASIS, the console operator should not notice any difference in the refresh rate of his OASIS displays.

<u>void CloseFramexSocket()</u>

This function closes the socket used to transfer data between the C&DHS and FrameX.

typedef.h

This file contains the definitions of various enumerated types used to define the structure of the downlink telemetry buffer.

ves.c

<u>void VesSimulation(unsigned char pulse, int counter)</u>

This function acts as a command interpreter and an interface to the VODS routines. Its functions are to execute the current command (if any), provide the VODS with parameters for its dynamics routines, and to insert the resulting VODS sensor outputs into the downlink telemetry buffer. Currently, simple on/off toggle commands are simulated in “VesSimulation” itself while thruster firing commands are passed on to the VODS.

vods.f

```
SUBROUTINE VODS       (PULSE, MAINFRAME, COMMAND, PARAM1, PARAM2,
                      PARAM3, PARAM4, EMS_TIME1, EMS_TIME2,
                      EMS_TIME3, EMS_TIME4, EMS_TIME5, EMS_TIME6,
                      EMS_TYPE12, EMS_TYPE34, EMS_TYPE56, SS_TIME,
```

SS_SIN, SS_COS, SS_BIAS, SS_ANGLE, CATBED1,
CATBED2, CATBED3, CATBED4, CATBED5, CATBED6)

Currently this is a dummy FORTRAN routine which inserts predefined values into the various VODS output parameters for testing purposes. All of the parameters between PARAM1 and SS_TIME inclusive are 2 byte values. All other parameters are single byte values. This function will be replaced by the full-blown VODS routines in the final version of the code.

caltable.cxx

<u>int readCalTable()</u>

This function reads two tab delimited files and assembles the data into three arrays used to store telemetry channel calibration information. The first input file is called "cal.dat" and contains a dump of the SYBASE project database information pertaining to telemetry channel mnemonic, data type, calibration type and numerical coefficients, etc. The second file is named "xref.dat" and contains a list of the telemetry channel mnemonics in the order in which they appear in "ves_cal.out", as well as an index indicating the byte number at which each mnemonic appears in the C&DH simulation software telemetry array.

The three output arrays are defined as follows. The first, named “tlmNames,” is simply an array of telemetry mnemonic strings from which the GUI user selects a parameter to modify. The second, named “tlmItems,” contains all of the telemetry channel-specific information such as mnemonic, description, display units, data type (length), data format (text, integer, float, hex, etc.), as well as an index into the third array named “calItems.” “calItems” contains the information required to perform the conversion from data number to engineering units. Array fields include mnemonic, calibration type, segment bound (if required), and numerical coefficients.

This function returns a value of zero if it completes successfully. If there are any missing or inconsistent data fields in the input files, the function prints an error message to the screen noting the filename, line number and type of error. The function then returns a value of -1 at which point the TlmMod process is terminated.

guidefs.h

This file contains the definitions of various enumerated types used to define the data structures used by TlmMod. In addition, abbreviations and descriptions of all telemetry parameters are defined here.

socks.cxx

int SocketIO::readNBytes(int socketId, int n, unsigned char* data)

This function reads “n” bytes of data from the socket identified by “socketId” and stores the information beginning at the location specified by the pointer “data.” The return value is the number of bytes read if the function terminates successfully. If the read fails, a message is printed to the screen with the error code from the system file “errno.h”, and the value ERROR (-1) is returned to the calling function. This function is designed to retry the socket read and not return with error in the event that it fails due to an interrupt.

int SocketIO::writeNBytes(int socketId, int n, unsigned char* data)

This function writes “n” bytes of data beginning at the location specified by the pointer data to the socket identified by “socketId.” The return value is the number of bytes written if the function terminates successfully. If the write fails, a message is printed to the screen with the error code from the system file “errno.h”, and the value ERROR (-1) is returned to the calling function. This function is designed to retry the socket write and not return with error in the event that it fails due to an interrupt. Also, to avoid having the application terminate if the receiving socket process quits, the function ignores the SIGPIPE interrupt.

SocketClient::SocketClient(char* h, char* p)

This is the constructor for objects of the SocketClient class. It initializes the host and port data members of the class and sets the status to IDLE (0).

int SocketClient::getStatus()

This function returns the value of the “status” data member which describes the current status of the socket connection.

int SocketClient::openSocket(long int arg)

This function attempts to open a client socket in either blocking (arg = 0) or non-blocking (arg != 0) mode by making the socket, ioctl, and connect system calls. If any of these calls fail, an error message is written to the screen and the value ERROR (-1) is returned. If the function completes successfully, the “status” data member is set to OK (2) and the value of the “socketId” data member is returned.

int SocketClient::readNBytes(int dumy, int n, unsigned char* data)

This function simply calls SocketIO::readNBytes using the SocketClient socket descriptor which is unknown to the SocketIO class.

<u>int SocketClient::writeNBytes(int dumy, int n, unsigned char* data)</u>

This function simply calls SocketIO::writeNBytes using the SocketClient socket descriptor which is unknown to the SocketIO class.

<u>void SocketClient::closeSocket()</u>

The connection with the server is severed and the "status" data member is set to IDLE (0).

<u>SocketServer::SocketServer(char* h, char* p)</u>

This is the constructor for the SocketServer class. It initializes the host and port data members of the class and sets the "status" data member to IDLE (0).

<u>int SocketServer::getStatus()</u>

This function returns the value of the "status" data member which describes the current status of the socket connection.

<u>int SocketServer::openSocket(long int arg)</u>

This function attempts to open a server socket in either blocking (arg = 0) or non-blocking (arg != 0) mode by making the socket, bind, listen, ioctl, and accept system calls. If any of these calls fails, an error message is written to the screen and the value ERROR (-1) is returned. If all of the calls are successful and the server is able to communicate with the client, the socket descriptor "socketId" is returned. However, in order to support non-blocking sockets, the condition of the socket is defined not only by the return value of the function but also by the "status" data member of the SocketServer object. Thus, "status" is set to IDLE (0) if an error occurred before reaching the accept call, BOUND (1) if the function successfully reached but failed the accept call (i.e. client not present), or OK (2) if the socket was successfully connected with the client. A diagram of this logic can be found in Figure 16.5-2.

beginning status	IDLE	IDLE	IDLE	BOUND	BOUND
socket, bind, listen, ioctl accept	fails (bypassed)	succeeds fails	succeeds succeeds	(bypassed) fails	(bypassed) succeeds

return value
ending status
	ERROR
	IDLE
		ERROR
		BOUND
			socketId
			OK
				ERROR
				BOUND
					socketId
					OK

Figure 16.5-2 SocketServer::SocketServer Function Logic

int SocketServer::readNBytes(int dumy, int n, unsigned char* data)

This function simply calls SocketIO::readNBytes using the SocketServer socket descriptor which is unknown to the SocketIO class.

int SocketServer::writeNBytes(int dumy, int n, unsigned char* data)

This function simply calls SocketIO::writeNBytes using the SocketServer socket descriptor which is unknown to the SocketIO class.

void SocketServer::closeSocket()

The connection with the client is severed and the “status” data member is set to IDLE (0).

socks.h

This is the header file containing the variable and class definitions required by the socket functions.

socks.hpp

This is the header file containing the #includes required by the socket functions.

tlmmod.cxx

WWindow1::WWindow1(zWindow *w, const char *title)

This is the constructor for objects of the Window1 class. The window created by an object of this class provides the user with a menu containing the update and exit options. A SocketClient object for data transfer with the C&DHS is created and initialized to “idl” (i.e. not yet bound), and a zTimer object with a two second period is created to synchronize this data transfer.

WWindow1::~WWindow1()

This is the destructor for the WWindow1 class. It frees the memory occupied by the SocketClient and zTimer objects by deleting their pointers.

int WWindow1::cmdMenu1update(zCommandEvt* ev)

This function has no VES-specific code and merely calls the constructor for a DDialog1 object.

int WWindow1::cmdMenu1exit(zCommandEvt* ev)

This function is executed when the user clicks “exit” on the main menu in order to terminate the program. The “sockstatus” member of the SocketClient object is set to “con_SD” such that the “SocketClient::socketTransfer” member function will close the socket connection and terminate the application.

void WWindow1::socketTransfer()

This is the function which is called by the zTimer object upon expiration of the two second interval timer. Its purpose is to manage the socket communications with the C&DHS and to create the data structure of modified downlink telemetry values to be sent to the C&DHS.

If the socket has not yet been bound, “SocketClient::openSocket” is called with the intent of connecting to the C&DHS server. If successful, the “sockstatus” member of the WWindow1 object is set to “con” (connected) and the socket I/O commences the next time the function is called. If the attempt to connect the client fails, the socket remains “idl” (idle) and the next attempt is made the next time the function is called. If the socket is connected and “sockstatus” becomes “con_SD” as a result of the user exiting the program, the function “SocketClient::closeSocket” is called to sever the communications link with the C&DHS. Immediately thereafter, the application is terminated.

As described in the “Data Structures” section above, a global two dimensional array of downlink telemetry parameters named “numdata” exists with the modified value in the first column and a persistence flag in the second column. The flag determines whether the C&DHS will downlink its current telemetry values or those modified values supplied by TlmMod (see “GuiTransfer”). The flag can have a value of zero, one, or two and is determined by the intent of the user when modifying data (see “DDialog1::clickedOK”). A zero flag means that this parameter has not been modified while a non-zero flag means that this parameter has been modified and that the modified value should appear in the current downlink telemetry frame. The difference between flag values of one and two is the persistence of the modified value. A value of one signifies that the parameter’s value has been modified but shall only be valid for a single telemetry frame before reverting to its true value. As a result, any parameter whose persistence flag equals one has both its value and its flag set to zero after being transmitted to the C&DHS. A value of two means that this parameter has been modified and that a modified value shall remain in the downlink stream until the flag becomes zero.

DDialog1::DDialog1(zWindow *w, const zResId& rid)

This is the constructor for objects of the DDialog1 class. If the user wishes to modify a telemetry parameter's value, the window created by an object of this class provides accept and cancel buttons as well as a list box containing all available parameter names. The VES-specific code adds the list of names to the empty list box and highlights the first element in the list.

int DDialog1::clickedOK(zNotifyEvt* ev)

This function is called when the user has selected a parameter to view or modify and has clicked the accept button. Once the identity of the selected parameter has been stored, a DDialog2 window is created.

DDialog1::~DDialog1()

This is the destructor for the DDialog1 class. It frees the memory occupied by the various window controls by deleting their pointers.

DDialog2::DDialog2(zWindow *w, const zResId& rid, int item)

This is the constructor for objects of the DDialog2 class. The window created by an object of this class enables the user to view a downlink telemetry parameter's current value in real time and also to insert a modified value into the downlink stream with two levels of persistence. The VES-specific code initializes a zTimer with a two second interval used to refresh the parameter's current value. The parameter abbreviation which was selected in the previous window is used as the caption of this window. The appropriate parameter description, input range for raw data numbers, and engineering units for calibrated values are assigned to their proper list boxes. The persist check box reflects the current persistence state of the parameter.

int DDialog2::clickedAccept(zNotifyEvt* ev)

This function is called when the user clicks the accept button in the DDialog2 window in order to send modified telemetry data to the C&DHS. First, the alphanumeric string entered by the user is converted into an integer value. If the conversion is successful, the integer data number is calibrated using SYBASE calibration tables and converted back into a formatted string to be displayed in both the current value and new value fields of the DDialog2 window. The new data number is stored in the global “numdata” telemetry data array and the value of the persistence flag is set according to the position of the persist check box (see WWindow1::socketTransfer). If an error occurs during either conversion, an appropriate error string is displayed instead of a calibrated numerical value.

int DDialog2::clickedConvert(zNotifyEvt* ev)

This function is called when the user clicks the convert button in the DDialog2 window in order to convert a new raw data number into its equivalent in engineering units. First, the alphanumeric string entered by the user is converted into an integer value. If the conversion is successful, the integer data number is calibrated using SYBASE calibration tables and converted back into a formatted string to be displayed in the new value field of the DDialog2 window. If an error occurs during the conversion, an appropriate error string is displayed instead of a calibrated numerical value.

void DDialog2::dnStrToDnInt()

This function converts a user input alphanumeric string into an integer.

void DDialog2::dnIntToEuStr()

This function performs a dn-to-eu calibration using the integer result of the dnStrToDnInt function. The function then converts the calibrated numerical result into a formatted string.

void DDialog2::output()

This function is called upon expiration of the two second zTimer created as part of a DDialog2 object. It updates the parameter’s current value with the value being transmitted to the C&DHS for inclusion in the downlink telemetry stream.

DDialog2::~DDialog2()

This is the destructor for the DDialog2 class. It frees the memory occupied by the various window controls and timers by deleting their pointers.

tlmmod.hpp

This is the header file containing the variable and class definitions for TlmMod.

ves_cal.out

This is an initialization file created using a SYBASE extraction tool and contains display and calibration information for VES telemetry monitors. This information is read by "readCalTable" and is used by TlmMod to compute calibrated values for the GUI interface.

xref.dat

This file contains "mnemonic name, byte position" pairs read by "readCalTable" to coordinate the array positions of telemetry monitors in Cdh and TlmMod data structures.

17 PostProcTlm PROGRAMMER'S GUIDE

17.1 Software Modules

To maintain PostProcTlm, first go to the program's development directory, /home/lunargrp/postproctlm, which at a minimum contains the source code postproc_tlm.c, the include and lib directories, and the makefile. Because the program's source code, the header file postproc_tlm.h, the PostProcTlm function library's source codes, and their respective makefiles are under SCCS control, you must then check out the file or files that you want to edit.

17.2 Known Limitations

The program has a number of limitations. If the user enters a character string at the prompt for the number of mnemonics for postprocessing, qwerty, for instance, the program enters an infinite loop. To break it, the user has to press the control and "C" keys. This action will also quit the program. If the user enters more than four digits (or characters in general) at the prompt for the year of the OASIS-CC output, the program writes to unallocated memory, without crashing. To see this explicitly, run the program from debugger with Run Time Checking (RTC) for memory access and memory use turned on. If the user lists the mnemonics at the prompt for a mnemonic for postprocessing, the program will always display twenty-one of them at a time regardless of the height of the window; and if the program displays a More prompt if there are more mnemonics to display, the user can only advance to the next twenty-one by pressing the return or enter key. If the user enters 'Y' at the Yes/No prompt for entering a specific interval of the OASIS-CC output's time span, the program does not display the time span to help the user enter an interval that at the least overlaps the time span. If the user enters a string with an even number of characters at a Yes/No prompt, the program will stop accepting even the correct entries "Y" and "N", unless the user presses the return or enter key at the prompt, or enter another string with an even number of characters, which ends with "N".

18 ATTITUDE DETERMINATION SOFTWARE PROGRAMMER'S GUIDE

19 COMMAND GENERATION SOFTWARE PROGRAMMER'S GUIDE

Appendix A - Lunar Prospector CSTOL Commands

Command Name	**Mnemonic/Macro Syntax**
Fire A1-Mode1	A1
Fire A1&A3+A2&A4-Mode3	A1234REOR
Fire A1&A2-Mode6	A12LOI
Fire A1&A4-Mode2	A14REOR
SAED	A1DETON 0 0 1 0 0
Thruster A1 Cat Bed Htrs	A1HTR
Fire A2-Mode1	A2
Fire A2&A3-Mode2	A23REOR
Thruster A2 Cat Bed Htrs	A2HTR
Fire A3-Mode1	A3
Fire A3&A4-Mode6	A34VEC
Thruster A3 Cat Bed Htrs	A3HTR
Fire A4-Mode1	A4
Thruster A4 Cat Bed Htrs	A4HTR
AHI	AHI
ALO	ALO
SAEWL	ALOENGY 100
APS Sensor On	AON
All Cat Bed Heaters Off	CATSOFF
PCM_Channel_Select_Off	CHANOFF
PCM_Channel_Select	CHANSEL 10
Clear Uplink Header-bit Errors	CLRERR
Coherent Mode Off	COHEROFF
Coherent Mode On	COHERON
Delay & Count Set	DELAYNUM 10.0 1
Sun Delay Set	DELAYSUN 10.0
Earth / Moon Sensor Off	EMOFF
Earth / Moon Sensor On	EMON
EMS_Sensor_Threshold	EMSTHR 100.0
Convolutional Encoder Off	ENCODOFF
Convolutional Encoder On	ENCODON
Mag / Er-ER Cover Open	ERCOV
EXECUTE	EXEC
Set_To_FAST_Rate	FAST
Set_To_Frame_Sequence	FRMSEQ
SBC454WH	GACSHI 100
SBC454WL	GACSLO 100

PCM_Gain2	GAIN2 100
SBGOWH	GBGOHI 100
SBGOWL	GBGOLO 100
SETH	GEARLYHI 100
SETL	GEARLYLO 100
SGH1LVL	GHV1LEV 100
D_SGH1CTL	GHV1OFF
SGH1CTL	GHV1ON
SGH2LVL	GHV2LEV 100
D_SGH2CTL	GHV2OFF
E_SGH2CTL	GHV2ON
SLTH	GLATEHI 100
SLTL	GLATELO 100
GRS Sensor On	GON
Half Rev Duration Set	HALFREV 10.0
Mode 1&6 Burn Duration Set	LONGDUR 10.0
Release Mag / Er	MAGBOOM
MBBYTE	MBBYTE 100
MBCOMMIT	MBCOMMIT 50 100
MBPROM	MBPROM
MBWORD	MBWORD 100
MDADDR	MDADDR 100
MDBYTE	MDBYTE 100
MDUMP	MDUMP 100
MDUMPR	MDUMPR 100
MDWORD	MDWORD 100
MEECKSM	MEECKSM
MBEEPROM	MEEPROM
MEEWEOFF	MEEWEOFF
MEEWEON	MEEWEON
Mag / Er-Power Both Off	MERABOFF
Mag / Er-Power A On / B Off	MERAON
Mag / Er-Power B On / A Off	MERBON
Mag / Er-Heater Power Off	MERHTROFF
Mag / Er-Heater Power On	MERHTRON
MERLVOFF	MERLVOFF
MERLVON	MERLVON
MEXEC	MEXEC 100
Xmtr to MGA	MGA
MLADDR	MLADDR 100
MRESET	MRESET
Master Reset all circuitry	MRRST

MTADDR	MTADDR 5 100
MTDUMP	MTDUMP 5 100
MTDUMPR	MTDUMPR 5 100
MTLOADB	MTLOADB 5 10 100
MTLOADW	MTLOADW 5 10 100
SHECDWH	NCDHI 100
SHECDWL	NCDLO 100
SNH1LVL	NHV1LEV 100
D_SNH1CTL	NHV1OFF
E_SNH1CTL	NHV1ON
SNH2LVL	NHV2LEV 100
D_SNH2CTL	NHV2OFF
E_SNH2CTL	NHV2ON
NS Sensor On	NON
No Op 1	NOOP1
No Op 2	NOOP2
SHESNWH	NSNHI 100
SHESNWL	NSNLO 100
Xmtr to Omni	OMNI
PCM_Offset_Gain1	OSGAIN1 10 100
Propellant Isolation Valve Closed	PIVCLOSE
Propellant Isolation Valve Open	PIVOPEN
Pressure XDCR Power Off	PRESSOFF
Pressure XDCR Power On	PRESSON
Pri Heater Bus Off	PRIHTROFF
Pri Heater Bus On	PRIHTRON
Simplify_Power_Down	PWRSM
Ranging Off	RANGOFF
Ranging On	RANGON
Release Booms	RELBOOMS 100.0
SDATA	SEBYTE12 100 200
SCHK	SECHECK 100
Red Heater Bus Off	SECHTROFF
Red Heater Bus On	SECHTRON
SCLR	SECLRERR
SEXECMD	SECMDEXEC
SNUM	SEDATCMD 10 100
SDUMP	SEDMPADDR 100
SEEPROMLD	SEEPROMLD
Power to Spectrometers-Off	SEOFF
Power to Spectrometers-On	SEON
SEROMLD	SEROMLD

SADD	SESTADDR 100
Reset Thruster Parameter registers	SETCMDREG
Reset 32-sec vehicle timer	SETTIM
D_SWDG	SEWDOGOFF
E_SWDG	SEWDOGON
Mode 2-5 Pulse Width Set	SHORTDUR 10.0
Display_Bytes	SHOWBYTE 100 200
SHVDIS	SHVDIS
Spare_Ext_28V_Pulse1	SP28V1
Spare_Ext_28V_Pulse10	SP28V10
Spare_Ext_28V_Pulse2	SP28V2
Spare_Ext_28V_Pulse3	SP28V3
Spare_Ext_28V_Pulse4	SP28V4
Spare_Ext_28V_Pulse5	SP28V5
Spare_Ext_28V_Pulse6	SP28V6
Spare_Ext_28V_Pulse7	SP28V7
Spare_Ext_28V_Pulse8	SP28V8
Spare_Ext_28V_Pulse9	SP28V9
All Sensors Off	SPECSOFF
Reset Thruster Execution	STOPFIRE
Sub-Carrier Oscillator Off	SUBOFF
Sub-Carrier Oscillator On	SUBON
Set_To_SubframeID	SUBSEQ
Sun Sensor Off	SUNOFF
Sun Sensor On	SUNON
Fire T1&T2-Mode4	T12VEC
Fire T1-Mode5	T1D
Thruster T1 Cat Bed Htrs	T1HTR
Thruster T2 Cat Bed Htrs	T2HTR
Fire T2-Mode5	T2UP
TLM to Engrg only format-300bps	T300
TLM to Science format-3600bps	T3600
Display_Tail	TAILBYTE
Stop_Time_Filling	TIMOFF
Start_Time_Filling	TIMON
Reset TLM circuitry-Relay open	TLMOFF
Reset TLM circuitry-Relay closed	TLMON
Charge Control V/T #1	VT1
Charge Control V/T #2	VT2
Charge Control V/T #3	VT3
Charge Control V/T #4	VT4
Transmitter Off	XMTOFF

Transmitter On	XMTON

Appendix B - Lunar Prospector Command Argument Types

Name	Size (bits)	Range
Integer_3	3	0 to 7
Integer_4	4	0 to 15
Integer_5	5	0 to 31
Integer_7	7	0 to 127
Short_Analog	8	0 to 255
Short_Integer	16	0 to 65535

Appendix C - Sample Lunar Prospector CSTOL Command Procedure

```
proc test

        write “entering sample OASIS procedure”
        wait ::10.0

        T3600
        write “Select Spacecraft Sci_Fmt command sent”
        write “check telemetry to validate command”
        write “enter GO at CSTOL prompt to send EXECUTE command”
        wait

        EXEC
        write “Execute command sent”
        write “pausing 10 seconds . . .”
        wait ::10.0

        write “now automatically sending commands 30 seconds apart”

        RANGON
        write “Ranging On command sent”
        wait ::5.0
        EXEC
        wait ::25.0

        HALFREV 10.0
        write “Half Rev Duration command sent with argument of 10.0 sec”
        wait ::30.0

        SHORTDUR 5.0
        write “Mode 2-5 Pulse Width Set cmd sent with argument of 5.0 sec”
        wait ::30.0

        DELAYNUM 10.0 1
        write “Delay & Count Set cmd sent with argument of 10.0 sec and 1 count”
        wait ::30.0

        DELAYSUN 30.0
        write “Sun Delay Set command sent with argument of 30.0 sec”
```

```
          wait ::30.0

          PIVOPEN
          write “Propellant Isolation Valve Open command sent”
          wait ::5.0
          EXEC
          wait ::25.0

          T2UP
          write “Fire T2-Mode 5 comand sent”
          wait ::5.0
          EXEC
          wait ::25.0

          T300
          write “TLM to Engrg only format command sent”
          wait ::5.0
          EXEC
          wait ::25.0

          write “exiting sample OASIS procedure”

end proc
```