Performing LET Memory Dumps

Robert Radocinski

February 12, 2007

After LET on the AHEAD spacecraft locked-up during the P4 manuever, it was decided that a dump of LET's memory would be useful in trying to diagnose the cause of the problem. The purpose of this document is to describe the method which was used to perform this dump and process the resulting data.

1 Dumping the LET Memory to Telemetry

The first step in dumping the LET memory is to verify that the SEP payload packets are being logged. The second step is uploading the FORTH file `letdump2.fth` to SEP Central, if `LETDUMP` is not already in the dictionary for SEP Central. A copy of the FORTH file can be found in appendix A. After the upload completes, the user must send `OK` to SEP Central to load the new word, `LETDUMP`, into the dictionary. After `LETDUMP` is in the dictionary, the user needs to send the command `LETDUMP` to SEP Central; this will begin the dump of the LET memory.

The dump process takes approximately 95 minutes. During that time, the 16 packets which are normally allocated to LET data are filled with dump data. To distinguish them from regular LET packets, they are assigned the ApID number 581. If monitoring the SEP payload packets, the user can detect the end of the dumping process when packets with ApID number 581 cease to appear in the telemetry. When the dump is finished, LET will not return to its normal processing until it is rebooted.

2 Processing the LET Dump in the Telemetry

After the dump is captured in telemetry, the user can use the two IDL procedures, `createLetDumpFile` and `printLetDumpFile` to create a dump listing.

The procedure `createLetDumpFile` creates a file with the LET dump records which are extracted from the payload telemetry file. `CreateLetDumpFile` takes two arguments. The first argument is a string with name of the input file with the payload telemetry packets. The second is a string that contains the name of the output file which will contain the LET dump records. If `createLetDumpFile` is run using a payload telemetry file capture by the ground system, the user should set the keyword argument `STD`; if the input payload telemetry file is coming from the leve0 data, then the user should not set the keyword argument `STD`.

The procedure `printLetDumpFile` is used to make an ASCII listing of the dump records. `PrintLetDumpFile` takes a single argument. The argument is a string of file containing the LET dump records. `PrintLetDumpFile` will print the ASCII listing of the dump records to the console. If the user wants to capture the ASCII listing in a file, the keyword `OUTPUT_FILE` should be set to a string containing the name of file to hold the ASCII listing.

A Listing of letdump2.fth

```
( small program to dump LET memory )

DECIMAL  ( requires HERE . 20079 )
VARIABLE STARTADR
VARIABLE CODELEN
: ?TRAN DUP >R RR8 RR8 2/ 2/ $3F AND 2 6 WITHIN
I STARTADR @ 6 +  = OR
I STARTADR @ $E + = OR
I STARTADR @ $68 + = OR
I STARTADR @ $6E + = OR
R> $800C0 = NOT AND ;

: RELOC
DUP STARTADR !
- 1- >R
STARTADR @ R>
FOR
DUP @ ?TRAN
IF CR DUP DUP . @ DUP . STARTADR @ - 1+ DUP .
OVER !
THEN
1+
NEXT DROP
STARTADR @ $7F + DUP @ ( kluge the else in < )
STARTADR @ - 1+ OVER ! DROP ;

VARIABLE FUDGE

CREATE HEADER $800C0 , $109 , 0 , 0 ,
CREATE PDATA 86 ALLOT

VARIABLE PSEQ#
VARIABLE CHKSUM
CODE * ( n n -- n )
   sta zero
   mul mul mul mul
   mul mul mul mul
   mul mul mul mul
   mul mul mul mul
   mul mul mul mul
   mul mul mul mul
   drop drop lda
   ret
: < - 2* -IF DROP -1 ELSE DROP 0 THEN ;
CODE ZFILL zero com add push sta anew zero stp times nop ret ;
: MOVE 1- FOR OVER @ OVER ! 1+ >R 1+ R> NEXT DROP DROP ;  ( src dest cnt -- )
: CLR-FRM 1 [ 9 ] G! ;
: WFRM CLR-FRM 0 BEGIN 1+ g0@ $40000 AND UNTIL ;
: WSEC59 WFRM DROP BEGIN WFRM 20000 < UNTIL ;
```

```
: MSECS 1- FOR 3198 FOR NEXT NEXT ;
: 3WSEC WFRM DROP WFRM DROP WFRM DROP 60 MSECS ;
: PBSEND $FF AND DUP CHKSUM +!
BEGIN g0@ $20000 AND UNTIL RR8 RR8 [ 2 ] G! ;
: PWSEND DUP RR8 RR8 PBSEND DUP RR8 PBSEND PBSEND ;
: PSEND 0 CHKSUM !
HEADER @ PWSEND
HEADER 1+ @ PWSEND
HEADER 2 + @ PWSEND
0 PBSEND 0 PBSEND
PDATA 85 FOR DUP @ PWSEND 1+ NEXT DROP
0 PBSEND 0 PBSEND CHKSUM @ NEGATE PBSEND ;
: APID! $7FF AND RR8 RR8 HEADER @ $F800FF AND XOR HEADER ! ;
: SEQ#!
DUP $3F00 AND RR8 HEADER @ $FFFFC0 AND XOR HEADER !
$FF AND RR8 HEADER 1+ @ $FFFF AND XOR HEADER 1+ ! ;
: SEQ!1+ PSEQ# @ SEQ#! PSEQ# 1+! ;
: P588 588 APID! 0 SEQ#! PDATA 86 ZFILL PSEND ;
: P589 589 APID! 0 SEQ#! PDATA 86 ZFILL PSEND ;
: P581 PSEQ# @ 86 * PDATA 86 MOVE 581 APID! SEQ!1+ PSEND ;
: 1MIN WSEC59 3WSEC P588 3WSEC P589
15 FOR 3WSEC P581 NEXT ;
: LD 0 PSEQ# ! 0 95 FOR 1MIN NEXT ;

PDATA 86 ZFILL
HERE FUDGE - CODELEN !
HERE FUDGE HEX RELOC DECIMAL
' LD FUDGE - 1+ FUDGE !

: LETDUMP LETCSEL CUINIT LETRSH 1MS LETRSL 1MS FUDGE CODELEN @ 2 + SENDRAW 1000 TCRCV ;
```