r e p o r t

Kavli Institute, MIT NE83-545
Building 300, One Kendall Square
Cambridge MA, 02139–4307
Tel: +1-617-253-7277
Fax: +1-617-253-8084

To: ACIS Science Operations Team

From: Peter Ford, NE83-545 <pgf@space.mit.edu>

Date: May 3rd 2017

Subject: Repairing Bit Errors in Compressed ACIS Packets (v 1.0)

1. Introduction

ACIS bias maps and raw image frames are usually compressed within the BEP before being sent to the spacecraft's RCTU for storage and/or immediate transmission. The compression scheme uses the Huffman First-Difference algorithm which subtracts each 12-bit pixel from its row neighbor and converts the result to a varying length bit string using a fixed translation table. ACIS flight software gives the user a choice of 5 tables, each created by an entropy coding algorithm from bias and raw frames in pre-launch testing.

It is a feature of Huffman encoding that each successive bit in the compressed record is either the end of a string in the translation table or isn't, and is therefore part of a longer string. While a row of pixels is being decompressed, there is no way of determining whether the data has been corrupted in transmission since all bit patterns are equally valid. Only after the entire packet has been decompressed can corruption be detected: if more or less than the expected number of 12-bit pixels result.

Nevertheless, bit errors in bias maps and raw frames are usually quite easy to spot, *e.g.*, using *ds9* with "scale" and "color" functions, although not easy to correct. This report describes how to identify and correct single bit errors in ACIS telemetry.

2. Locating the Bad Packet

Bias maps and raw frames are compressed row-by-row into a series of telemetry packets, respectively *dataTeBiasMap* and *dataTeRaw*. Each *dataTeBiasMap* packet begins with an 11-word header (of 32-bit words), followed by 1–1012 words of compressed bias map, typically containing 8–10 complete rows. Similarly, each *dataTeRaw* packet begins with a 6-word header followed by 1–1017 words of compressed data, containing 2–5 raw rows. The following discussion will concentrate on recovering damaged *dataTeBiasMap* packets.

Having determined the range of bad `CHIPY` values from *ds9*, the next step is to locate the damaged packet. In the current processing phase directory, the packet headers are found in "psci/acis*phase.run*.bias.log.gz", *e.g.*,

```
dataTeBiasMap[1546210,8672,82890:016,145:22302] = {
  telemetryLength               = 816
  formatTag                     = 14 # TTAG_SCI_TE_BIAS
  sequenceNumber                = 8601
  biasStartTime                 = 0x84e61df6
  biasParameterId               = 0x006e6034
  ccdId                         = 1 # CCD_I1
  fepId                         = 3 # FEP_3
  dataPacketNumber              = 97
  initialOverclocks             =   654   713   487   843
  pixelsPerRow                  = 1023
  rowsPerBias                   = 1023
  ccdRow                        = 53
  ccdRowCount                   = 9
  compressionTableSlotIndex     = 1
  compressionTableIdentifier    = 0xfffffffe
  pixelCount                    = 10240
  data                          = [805]
}
```

Bias packets are filled from the highest row number downwards, *i.e.*, starting at row 1023 (or less in subframe mode). This particular packet contains 10 rows (*i.e.*, one more than the *rowCount* value), starting at row 53 (one less than its `Image Y` value in *ds9*) and ending at row 44.

3. Locating the Bad Bit

This is made easy by the *find-bias-bit.pl* script, which flips every bit in the damaged packet until the result "seems" reasonable, *i.e.*, until it results in a valid decompression and shows no gross difference between the recompressed rows and those of the preceding and following packets. The options are as follows:

```
find-bias-bit.pl
    [-P dir]       # preserve temporary files []
    [-2]           # two-bit search []
    [-b]           # byte search []
    [-c ccd]       # this ccdId []
    [-f fep]       # this fepId []
    [-h file]      # Huffman table [/nfs/acis/h4/tools/lib/huff.dat]
    [-i id]        # this biasParameterId []
    [-n val]       # column variance limit [0]
    [-r from[,to]] # range of bit offsets [0]
    [-t file]      # ACIS command table [/nfs/acis/h1/www/bin/current.dat]
    [-v]           # verbose []
    ccdRow         # the ccdRow of the bad packet []
    [file...]      # input EHS file(s)
```

This *perl* script reads the ACIS telemetry that includes the bad bias packet and the *dumpedTeBlock* that controls the run that created it. The packet is identified by its *ccdRow* and, if necessary, by the *fepId*, *ccdId*, and/or *biasParameterId* of its bias map. If no EHS file is specified, the input to *stdin* must contain ACIS packets, *e.g.*, as output from *getnrt*, *getPackets*, or *getp*. *find-bias-bit.pl* will start the *psci* program and feed it sets of 5 or 6 packets:

1	*dumpedTeBlock*	A duplicate of the latest *dumpedTeBlock* preceding the damaged bias packet
2	*dataTeBiasMap*	The bias packet preceding the damaged one
3	*dataTeBiasMap*	The damaged packet with one bit "flipped"
4	*dataTeBiasMap*	The bias packet following the damaged one
5	*dataTeBiasMap*	The last bias packet in the damaged map
6	*scienceReport*	A "generic" end-of-run packet reporting no errors

If the damaged packet is either the first or last packet in the map, packets 2 and 4 are taken from the two nearest the damaged one. Packet 5 is needed to force *psci* to write a partially filled bias map. The group of packets is repeatedly sent to *psci*, flipping each bit in turn in the data portion of the damaged packet.

The script monitors the *stderr* output from *psci*. If a, "unpacking failure" is reported, the corresponding bias map is ignored. Otherwise, the bias map is read and the pixels corresponding to packets 2, 3, and 4 identified. Then, for each column, the difference is taken between the value of each pixel in the "bad" packet and the average of the pixels from the "good" packets in that same column. These differences are squared and summed over all columns, to form a measure that is minimized over all possible bit flips to identify the most likely damaged bit.

The result is the following pair of lines, the first detailing the input parameters and the second the result of the computation, which typically takes about 10 minutes on a fast processor.

```
ccdId 1 fepId 3 biasId 0x006e6034 ccdRow 53 npkt 97 sigma 0 bits 25760
varmin 9.35913 bitmin 9604 vcdu 82890:016 seqnum 8601 pkt[311] 0x8353e153 -> 0x8353e143
```

The crucial numbers are (`from+bitmin`), the bit offset of the error within the `dataTeBiasMap.data[]` array, and the hexadecimal values of the original and corrected words. `From` is zero unless supplied as the first or only argument of the `-r` option. If the `-b` flag is used, the eight least significant 8 bits of `bitmin` contain the corrected value of the byte at bit offset `(from+8*(bitmin>>8))` from the start of the compressed data array. Similarly, if the `-2` flag is used, the two corrupt bits are located at bit offsets `(from+(bitmin/(to-from+1)))` and `(from+(bitmin%(to-from+1)))`.

4. Correcting the EHS File

Having located the bad bit, byte, or bits, the next step is to find the corresponding data in the EHS telemetry file. This is made complicated by the segmentation applied by the Chandra telemetry system into separate minor frames and, within those frames, to separate blocks of science data. To assist in locating the bits, the *getnrt* script has a special `–W` option that instructs it to report the location of a word in an ACIS packet, *e.g.*,

```
# getnrt -W 0x8353e153 2016_145_0404_145_1142_Dump_EM_51198.gz >/dev/null
0x8353e153 located in EHS block 82890:016 at byte +34160703
```

If multiple copies of the word are reported, the VCDU count reported by *find-bias-bit.pl* can be used to distinguish between them. Unless the ACIS word (*e.g.*, `0x8353e153` in our example) is split between EHS segments, the byte can be easily replaced. Since ACIS uses LSB (little endian) format, the byte at offset `+34160703` contains the bad `0x53` value, `+34160703` contains `0xe1`, etc., so `0x53` can be replaced by `0x43` as follows:

```
# gunzip -c 2016_145_0404_145_1142_Dump_EM_51198.gz |\
  perl -e 'read(STDIN,$_,9e9); substr($_,34160703,1) = pack("C",0x43); print' |\
  gzip > 2016_145_0404_145_1142_Dump_EM_51198-a.gz
```

However, if the replacement isn't in the first (lowest) byte of the word, it is safer to check that the word isn't split into separate EHS blocks or segments, as illustrated in the *Makefile* in section 7. If it is, it is best to dump the ACIS packet and the individual EHS block separately to locate the byte to be replaced.

5. Conclusion

Single bit errors in bias map or raw image packets can be repaired with relatively little work but it isn't possible to automate the process in all circumstances because the corruption could involve more than one bit, the "fixed" packet may be hard to distinguish from the damaged one, and the packed 32-bit word may be split between separate EHS blocks or segments.

6. References

1. User Interface to the ACIS Instrument, "*acistools.pdf*", pp. 2-15, revised January 30, 2015.

2. ACIS IP&CL Structure Definition Notes, MIT Report 36–53206.0204, Revision N (2003).

7. Example

The following *Makefile* was used to locate the one-bit bias error in OBSID 18294. It includes all of the steps described in this report with the addition of a final test to recreate the full bias map.

```
#
#         Fix a bad bit in a TE bias packet
#
OBSID     = 18294
DIR       = /nfs/maax/r2/eC
EHS       = 2016_145_0404_145_1142_Dump_EM_51198.gz
PKTS      = $(DIR)/acis91a/pkts/run-34.pkts.gz
HUFF      = $$ACISTOOLSDIR/lib/huff.dat
DAT       = /nfs/acis/h1/www/bin/current.dat
FIND      = find-bias-bit.pl -h$(HUFF) -t$(DAT)
IN        = $(DIR)/ssr/$(EHS)

# identification of the bad dataTeBiasMap packet
CCDID     = 1
FEPID     = 3
BIASID    = 0x006e6034
CCDROW    = 53
SIGMA     = 0
V         = -v
```

Continued overleaf

```
# Bad and replacement word extracted from $(OBSID).txt, in LSB order
OLDVAL    = 0x8353e153
NEWVAL    = 0x8353e143

# Byte offset of bad byte in uncompressed EHS file, original and repaired byte values.
REPL      = 34160703,0x53,0x43
# To replace two-bit errors, use -2 flag and 6 comma-separated values in $(REPL)

all:      $(OBSID).test

# Locate the bad bit within its dataTeBiasMap packet
$(OBSID).txt:    $(IN)
          $(FIND) $V -c$(CCDID) -f$(FEPID) -i$(BIASID) -n$(SIGMA) $(CCDROW) $? > $@

# locate the word in the EHS file
$(OBSID).loc:    $(OBSID).txt
          @getnrt -W $(OLDVAL) $(IN) >/dev/null 2>$@

# Create a copy of the SSR file with the damaged byte fixed
$(OBSID).log:    $(OBSID).loc
          @gunzip -c $(IN) |\
          perl \
            -e '$$f="$(EHS): byte 0x%02x at +%d %s 0x%02x\n",' \
            -e 'read(STDIN,$$_,9e9) > 0 || die "$(EHS): $$!\n";' \
            -e '@P = ($(REPL)) ;' \
            -e 'while(($$o,$$a,$$b,@P) = @P) {' \
            -e '  ($$n=unpack("C",substr($$_,$$o,1))) == $$a ||' \
            -e '  die sprintf($$f,$$n,$$o,"!=",$$a);' \
            -e '  substr($$_,$$o,1) = pack("C",$$b);' \
            -e '  warn sprintf($$f,$$a,$$o,"replaced by",$$b);' \
            -e '}; print' 2>$@ |\
          gzip >./$(EHS)

# Run psci once again to check that the EHS file is fixed
$(OBSID).test:   $(OBSID).log
          @getnrt -O ./$(EHS) |\
          psci -q -t $(DAT) -h $(HUFF) -l foo 2>$@ && \
          for i in foo.*.bias.log ; do \
            j=`expr "$$i" : 'foo\.\([0-9]*\)\.bias\.log'` ;\
            grep "biasParameterId *= $(ID)" $$i >/dev/null && \
            gzip -cf foo.$$j.$(FEP).bias.fits > test.$(FEP).bias.fits.gz && \
            ls -l test.$(FEP).bias.fits.gz >>$@ && rm -f foo.* && break ;\
          done

# Extract the damaged dataTeBiasMap packet to assist in locating the damaged byte
NSEQ      = 8601
$(OBSID).pkt:    $(IN)
          @getnrt $(IN) | fpkt -c $(CCD) -s $(NSEQ) 14 >$@

# Extract the VCDU range in EHS format to assist in locating the damaged byte
VCDU1     = 82890:016
VCDU2     = 82890:024
$(OBSID).ehs:    $(IN)
          @set -- `ehs2mnf -V $(IN) |\
          awk '/ $(VCDU1) / {s=$$3} / $(VCDU2) / {print s,$$3-s}'` ;\
          gunzip -c $(IN) |\
          perl -e "read(STDIN,\$$_,$$1);print if read(STDIN,\$$_,$$2)" >$@
```

Note the optional `$(OBSID).pkt` target that extracts the damaged *dataTeBiasMap* packet and `$(OBSID).ehs` that extracts one or more EHS blocks. Hexadecimal dumps of these files may be useful in locating a replacement byte if it is located in a 32-bit word that is split between telemetry segments.