		ENGINEERING CHANGE ORDER		<u>ECO No.</u> <u>36-1035</u>
CENTER FOR SPACE RESEARCH MASSACHUSETTS INSTITUTE OF TECHNOLOGY				
DWG. NO.	NEW REV.	DRAWING TITLE		
36-58010	C	Flight Software Standard Patch Release C		
REASON FOR CHANGE: Addition of standard patches <i>tlmbusy</i> and <i>buscrash</i> . These patches address software problem reports 138 and 140. No new optional patches are added.				
DESCRIPTION OF CHANGE: The optional patches—release C—are unchanged. The new set of standard—release C—patches is compiled and loaded into a common address space so that each optional patch can be loaded independently of the others, provided the load order defined in <i>PatchRelease.spec</i> is maintained. Patches <i>eventhist</i> , <i>ctireport1</i> , and <i>ctireport2</i> require that <i>smtimedlookup</i> is also loaded; similarly, the engineering patches <i>deaeng</i> , <i>dearepl</i> , and <i>printswhouse</i> require the <i>tlmio</i> patch. <i>deaeng</i> and <i>dearepl</i> must not be loaded at the same time.				
	SIGNATURE	DATE	REMARKS:	
ORIGINATOR	Peter Ford	08/09/07	Approved	
MECHANICAL				
ELECTRICAL				
SOFTWARE				
STRUCTURE				
FABRICATION				
SCIENCE				
SYSTEMS ENG.				
QUALITY				
PROJ. ENGINEER				
DEPUTY PM				
PROJ. MANAGER				

Existing ACIS Flight Software Patches

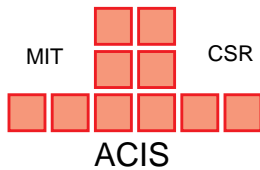
	Name	Rev	Size	Part	ECO	SPR
Standard Release C						
1	corruptblock	A	16	36-58030.01	994	113
2	digestbiaserror	A	64	36-58030.02	995	116
3	histogramvar	A	16	36-58030.03	999	115
4	biastiming	A	1612	36-58030.04	993	117
5	rquad	A	16	36-58030.14	1000	121
6	histogrammean	A	156	36-58030.15	996	123
7	zaplexpo	A	64	36-58030.16	997	122
8	condoclk	A	472	36-58030.17	1012	127
9	fepbiasparity2	A	504	36-58030.19	1015	130
10	cornermean	A	32	36-58030.21	1017	128
11	tlmbusy	A-	328	36-58030.29	1033	138
12	buscrash	A-	280	36-58030.30	1034	140
Optional Release C						
1	eventhist	B	5908	36-58030.05	1025	N/A
2	cc3x3	B	4636	36-58030.06	1018	120,124,126
3	teignore	A	36	36-58030.09	1003	N/A
4	ccignore	A	36	36-58030.10	1004	N/A
5	smtimedlookup	A	3712	36-58030.24	1025	N/A
6	ctireport1	A	5452	36-58030.25	1026	N/A
7	ctireport2	A	2784	36-58030.26	1026	N/A
8	compressall	A	2368	36-58030.27	1027	134
9	untricklebias	A	1612	36-58030.28	1028	133
10	reportgrade1	A	816	36-58030.22	1021	131,132
Under Development						
1	hybrid	03	6104	36-58030.13	1010	N/A
2	fepbiasparity1	02		36-58030.18	1014	N/A
3	squeegy	06	4412	36-58030.23	1023	N/A
4	forcebiastrickle	01	N/A	36-58030.29	1024	133
Engineering Unit Utility Patches						
1	tlmio	02	10312	36-58030.07	1010	N/A
2	printshouse	01	7224	36-58030.08	986	N/A
3	deaeng	02	2604	36-58030.11	1010	N/A
4	dearepl	02	556	36-58030.12	1010	N/A

Status of Patch Release C, Optional Revision C

Name	Part Number	Description	Typos ^a	RIDs ^b	Status
<i>tlmbusy</i>	36-58030.29	Prevent BEP telemetry packet loss			Awaiting review
<i>buscrash</i>	36-58030.30	Prevent BEP bus crash on FEP powerdown			Awaiting review
S/W Review	36-58020 C-C-D	Documentation accompanying the individual patch ECOs			Awaiting review
Certification	36-58021.03	Documentation describing the multi-patch certification tests			Awaiting review

a. typographical errors in the documentation

b. review item discrepancies—requiring changes to the patch code and/or test procedures



ACIS SOFTWARE PROBLEM REPORT

CENTER FOR SPACE RESEARCH
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FOR: Part Number			Used on hardware:	
Rev:	Sub-Section Name:	DEA Rev:	Human Interface:	
36-54002.08	1.5	SW ACIS FLT 1.5		
Originator:		Phone:	Date:	RCTU Rev:
P. Ford		x3-7277	04/07/05	Front End HW:

Description of Problem: (should be sufficiently complete to be duplicated by engineering):

Missing data packet during OBSID 5645

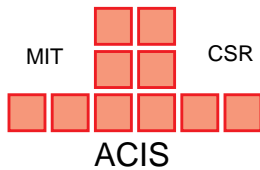
On 30 March 2005, the CXO DS reported that OBSID couldn't be processed because a parameter block was missing from downlink. It was subsequently discovered that, although the parameter block had been transmitted, a small number of bad packets within the event-processing phase of this run were responsible for terminating all data processing. The identical anomaly was present in both R/T telemetry and in the subsequent SSR dump (2005_088_2214_089_0250_Dump_EM_93236). In place of a single exposureTeVeryFaint packet, the telemetry stream contained three packet headers: one dataCcGraded and two dataTeVeryFaint. All three packet lengths were wrong, *i.e.*, they were not followed by the number of data words defined in their headers: the dataCcGraded packet was actually only 8 bytes long, and the dataTeVeryFaint packets were actually 2048 bytes long.

Corrective Action:

The problem was traced to a feature of the BEP telemetry manager. Telemetry packets are enqueued via a call to the `TlmManager::post()` method. If this routine is called by one task while it is still processing a call from another task, the first packet will be truncated and a random block of data from within the telemetry buffer area will be telemetered in place of the second packet.

The problem can be prevented by cancelling task switching while `TlmManager::post()` is running, and has been implemented as the `tlmbusy` standard patch. This was verified by running the ACIS engineering unit under very heavy load, causing 5 tasks to post telemetry packets "simultaneously" (see ACIS report SPR138-1.0, September 30, 2005).

Problem closed on:	Date:	Refer to ECO #:	Refer to Patch ID:
	08/09/2007	36-1033	tlmbusy
Problem ID: M05040701	Status: Closed		Sheet: 138 of 140



ACIS SOFTWARE PROBLEM REPORT

CENTER FOR SPACE RESEARCH
 MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FOR: Part Number			Rev:	Sub-Section Name:	Used on hardware: DEA Rev:	Human Interface:
36-54002.08		1.5	SW ACIS FLT 1.5		Flight	
Originator:		Phone:	Date:		RCTU Rev:	Front End HW:
P. Ford		x3-7277	12/13/06			

Description of Problem: (should be sufficiently complete to be duplicated by engineering):

ACIS BEP experienced a bus error during SCS107

When ACIS was halted by an SCS107 (high-radiation shut-down) command on 12/13/2006, the BEP suffered a bus error and watchdog reboot. Studying previous occasions, it was discovered that bus errors occurred whenever the SCS107 was issued while the ACIS FEPs were computing their bias maps (3 instances) but never while they were writing those maps to telemetry or processing event data (64 instances) or raw frames (1 instance).

Corrective Action:

The BEP flight code was examined to determine whether the science thread was correctly examining the power-on status of FEPs before accessing their command mailboxes. It was found that the code that marks bad pixels and columns in the FEP bias maps was not protected against a FEP power-down.

A patch (`buscrash`) was generated that calls `fepManager.isEnabled()` to determine whether to update the bias maps. The patch was run on the ACIS engineering unit, and was found to prevent the bus crash.

Problem closed on:	Date: 08/09/2007	Refer to ECO #: 36-1034	Refer to Patch ID: buscrash
Problem ID: M06121301	Status: Closed		Sheet: 140 of 140

Massachusetts Institute of Technology
 Kavli Institute for Astrophysics & Space Research
 1 Hampshire Street
 Cambridge, MA 02139-4307



Tel: 617-253-7277
 Fax: 617-253-8084

report

Date: September 30, 2005 5:41 PM
To: ACIS Instrument Team
From: Peter G. Ford, NE80-6071 <pgf@space.mit.edu>
Subject: Version 1.0 of a report relating to SPR 138 (M05040701):
 “Missing data packet during OBSID 5645”
Cc: Chandra SOT <sot@head-cfa.cfa.harvard.edu>

1. Introduction

The CXCDs was unable to process OBSID 5645 at the first attempt on March 30, 2005, because of the presence of a single data packet of type *dataCcGraded* (see Applicable Document 2). Since the observation was being conducted in Very Faint Timed Exposure mode, this unexpected packet caused the processing program to halt. At the time that the packet was generated, Chandra was in real-time contact with a DSN station. The real-time and recorded data were identical, implicating either the instrument itself or the RCTU that relays its telemetry to the on-board data system. After examining a dump of the telemetry stream, CXCDs personnel removed 8 consecutive telemetry minor frames, and OBSID 5645 was successfully processed.

This anomaly is being tracked by DDTs as **OCCcm07109** and by the ACIS SI team as **SPR 138**.

2. Preliminary Analysis

After filing a Software Problem Report (Applicable Document 1), the first task of the ACIS team was to examine the logs from quick-look *psci* processing. The messages generated during OBSID 5645 are illustrated in Fig. 1. While merging real-time and SSR data, any mismatch between the two data streams would have been reported. It is therefore the presence of the three “bad ACIS serial byte” messages and the single “missing sequenceNumber” message, *without* an accompanying “missing minor frame(s)” or “R/T-SSR mismatch” message that constitutes the ACIS anomaly.

The relevant SSR dump file (2005_088_2214_089_0250_Dump_EM_93236) was examined in some detail. It spanned frames 67910:044 – 68416:035, but the anomalous packets were restricted to the seven minor frames 068390:076 – 068390:082. Earlier in that file, a number of minor frame headers began with anomalous sync codes,¹ viz.

```
bad sync 0xe53003e2 replaced at vcdv 67911:058
bad sync 0xe53003e5 replaced at vcdv 67912:075
bad sync 0xe6cffc1d replaced at vcdv 67913:068
bad sync 0x1acff862 replaced at vcdv 67913:092
```

and there had been numerous similarly garbled sync words in the previous SSR file, but all frame headers contributing to OBSID 5645 appeared normal and the anomaly was restricted to that portion of the minor frames allocated to output from ACIS.

1. Each minor frame should begin with the synchronization pattern, 0x1acffc1d.

```

068257:016 starting ACIS science run 53
068380:022 R/T missing 215 minor frame(s) byte
068390:080 bad ACIS serial byte 0x52
068390:080 bad ACIS serial byte 0x8e
068390:080 bad ACIS serial byte 0x2a
068390:080 dataTeVryFaint.sequenceNumber=31834 != 31833
068398:071 R/T missing 4 minor frame(s) byte +7334042
068412:080 2005_089_0248_089_1138_Dump_EM_93423 starts at 2005/089/10115.400
068412:080 SSR extra 468 minor frame(s) byte +537498
068416:035 2005_088_2214_089_0250_Dump_EM_93236 ends at 2005/089/10235.325
068435:015 R/T missing 4 minor frame(s) byte +12704428
068440:031 raw_telem_2005-03-29_18:33:27 ends at 2005/089/11021.425
068440:032 raw_telem_2005-03-29_22:03:44 starts at 2005/089/11021.500
068471:091 R/T missing 4 minor frame(s) byte +4626158
068484:127 R/T missing 105902 minor frame(s) byte +7963056
069312:051 R/T missing 264 minor frame(s) byte +8275716
069314:060 R/T missing 6 minor frame(s) byte +8280314
069349:087 raw_telem_2005-03-29_22:03:44 ends at 2005/089/40851.025
069349:088 raw_telem_2005-03-30_06:20:54 starts at 2005/089/40851.000
069349:099 R/T missing 4 minor frame(s) byte +13782
069379:016 2005_089_1136_089_2210_Dump_EM_93700 starts at 2005/089/41816.600
069379:016 SSR extra 474 minor frame(s) byte +542092
069382:105 2005_089_0248_089_1138_Dump_EM_93423 ends at 2005/089/41938.575
069386:047 R/T missing 4 minor frame(s) byte +5388762
069421:045 R/T missing 132410 minor frame(s) byte +12271034
070463:107 raw_telem_2005-03-30_06:20:54 ends at 2005/089/77395.375
070463:108 raw_telem_2005-03-30_16:29:58 starts at 2005/089/77395.300
070481:075 R/T missing 4 minor frame(s) byte +2609392
070484:069 R/T missing 2 minor frame(s) byte +3041232
070518:023 R/T missing 4 minor frame(s) byte +7984376
070535:024 2005_089_2208_090_0402_Dump_EM_93819 starts at 2005/089/79735.400
070535:024 2005_089_2208_090_0402_Dump_EM_93819 starts at 2005/089/79735.400
070535:024 SSR extra 468 minor frame(s) byte +537498
070538:107 2005_089_1136_089_2210_Dump_EM_93700 ends at 2005/089/79855.375
070554:099 R/T missing 4 minor frame(s) byte +13359356
070555:055 raw_telem_2005-03-30_16:29:58 ends at 2005/089/80399.625
070555:056 raw_telem_2005-03-30_17:20:02 starts at 2005/089/80399.600
070580:026 R/T missing 70726 minor frame(s) byte +4570814
071041:120 ending ACIS science run 53

```

Figure 1: The *psci* log for OBSID 5645: messages are tagged by their major:minor frame indices. The only indication of the anomaly is highlighted: a single data packet, sequence number 31833, is missing, but there is no accompanying “missing minor frame(s)” report. Byte offsets are reported from the start of the (uncompressed) EHS file. Times are UTC.

The anomalous ACIS data were output at 2:36:29 UTC on March 30, 2005. ACIS was running in the S configuration with 5 CCDs (I2, I3, S2-4) and had just finished reporting exposure 982. With no data yet available from exposure 983, the back-end processor (BEP) idled, outputting a stream of “pad” bytes (0xb7). The data packets and pads generated in this time interval are listed in Table 1. The readout times are those at which the packets begin to be inserted into telemetry frames. The three highlighted packets were anomalous in several respects:

1. The packet lengths recorded in their 5th and 6th header bytes did not correspond to the number of (non-pad) bytes before the next packet header.
2. The packet sequence numbers, recorded in their 7th and 8th header bytes, were out of order relative to the preceding and following packets.

Table 1: Data packets from ACIS exposure numbers 982 and 983[†]

31822	0.00000	exposureTeVeryFaint	18	18	3	3	981	
	0.01800	3693 pad bytes						
31823	1.26150	dataTeVeryFaint	333	333	7	1	982	0
31824	1.70700	exposureTeVeryFaint	18	18	7	1	982	
31825	1.73300	dataTeVeryFaint	123	123	8	4	982	0
31826	1.89600	exposureTeVeryFaint	18	18	8	4	982	
31827	1.92225	dataTeVeryFaint	283	283	6	2	982	0
31828	2.29350	exposureTeVeryFaint	18	18	6	2	982	
31829	2.31950	dataTeVeryFaint	123	123	2	5	982	0
31830	2.48275	exposureTeVeryFaint	18	18	2	5	982	
31831	2.50875	dataTeVeryFaint	183	183	3	3	982	0
31832	2.75600	exposureTeVeryFaint	18	18	3	3	982	
	2.77400	4703 pad bytes						
23572	4.35825	dataCcGraded	795	2	6	6		
31740	4.36025	dataTeVeryFaint	283	512	3	3	973	0
31630	5.04100	dataTeVeryFaint	253	512	6	2	963	0
31834	5.73650	dataTeVeryFaint	503	503	8	4	983	0
31835	6.40825	dataTeVeryFaint	163	163	2	5	983	0
31836	6.61950	exposureTeVeryFaint	18	18	2	5	983	
31837	6.64550	dataTeVeryFaint	283	283	7	1	983	0
31838	7.02475	exposureTeVeryFaint	18	18	7	1	983	
31839	7.05100	dataTeVeryFaint	183	183	6	2	983	0
31840	7.29000	exposureTeVeryFaint	18	18	6	2	983	
31841	7.32325	dataTeVeryFaint	53	53	3	3	983	0
31842	7.40025	exposureTeVeryFaint	18	18	3	3	983	
31843	7.41825	dataTeVeryFaint	23	23	8	4	983	1
31844	7.44925	exposureTeVeryFaint	18	18	8	4	983	
	7.47525	169 pad bytes						
31845	7.53350	dataTeVeryFaint	233	233	7	1	984	0

[†] The readout time—relative to the start of packet 31822—is determined by the location of the start of the packet (or padding) in a telemetry minor frame. The packet length (in 32-bit words) is recorded in each packet header; the “actual” length column shows the number of words until the next packet header. The “PktNo” field denotes a continuation packet, *i.e.*, there were more events from a given CCD than could fit into a single data packet.

3. The *dataCcGraded* packet was anomalously short: the minimum length for such a packet is 5 words; also, the packet type (recorded in header byte 6) was inconsistent with the parameter block used for this run (whose identifier was in each *exposureTeVeryFaint* packet.) In fact, the only instrument mode that could generate such a packet, Cc1x3 graded mode, has never been used since Chandra launch.
4. The two anomalous *dataTeVeryFaint* packets were too long—since each Very Faint event occupies 5 words, their maximum length is only 503 words (*e.g.*, packet 31834).
5. The contents of legitimate *dataTeVeryFaint* packets are tallied in *exposureTeVeryFaint* packets. The number of exposures per frame recorded in the latter was entirely consistent with the number of events in the former, leaving no possibility that the three anomalous packets would have been generated in normal ACIS operation.
6. The sequence numbers (31740 and 31630) recorded in the headers of the anomalous *dataTeVeryFaint* packets corresponded to packets that had already been telemetered

during this run. In fact, the contents of the anomalous packets were *identical* to those already telemetered, with the addition of “random” words to make up a total length of 2048 bytes before the next packet header.

3. A Missing Packet

Since packet sequence number 31833 was missing, it seemed reasonable to suppose that the BEP was trying to write that packet to the RCTU when the anomaly occurred. But what sort of packet was it? It could not have been a *dataTeVeryFaint* or *exposureTeVeryFaint* packet since these are all accounted for. It was unlikely to be a *commandEcho* packet since an inspection of the daily Chandra command load shows no commands being sent to ACIS during this time period. That leaves either *bepStartupMessage* or *fatalError* packets—both of which would have caused the BEP to reboot and to terminate the science run—or software or analog (DEA) housekeeping packets. Table 2 shows the sequence numbers of the housekeeping packets spanning the time of the anomaly, their creation times as measured by the BEP interrupt clock, and the time interval between successive housekeeping packets. This time interval is approximately 64 seconds for software housekeeping, and is selected by ground command for analog housekeeping (the current value is 17 seconds.) Note the anomalous interval between the pair of analog housekeeping packets with sequence numbers 31782 and 31898.

Table 2: Housekeeping Packets

Packet Sequence Number	Packet ID	Packet ID	Interval (s)	Packet Sequence Number	Packet ID	Interval (s)
31501	0x0cb72d01	0x0cb72f81	64.0	31721	0x0cb731c2	17.0
31720	0x0cb72f81	0x0cb73201	64.0	31782	0x0cb7326c	17.1
31940	0x0cb73201	0x0cb73481	64.0	31898	0x0cb733c0	34.0
32160	0x0cb73481	0x0cb73701	64.0	31961	0x0cb7346a	17.0

[†] Housekeeping packets contain the value of the 32-bit BEP interrupt counter at the time the packet was first created and (for software housekeeping only) the time at which the packet was written to telemetry. The interrupt counter is incremented whenever a 100 ms timer expires, but the true interval can be slightly longer when the interrupt handler has several tasks to perform, *e.g.*, when receiving lengthy commands or generating many short telemetry packets. In this table, the “elapsed” fields—time in seconds between housekeeping packets—are calculated on the assumption that the timer intervals are all 100 ms.

While the BEP software will prevent an analog housekeeping packet from being generated when no telemetry buffers are available, this cannot be the case in this instance since the buffers are being written almost as soon as they are filled, as evidenced by the numerous pad bytes that separate each exposure. This implies that the BEP tried to write the housekeeping packet, but that something went astray and that 1026 words (4104 bytes) were written from another location instead. The suspicion falls on the BEP hardware that copies telemetry packets to the RCTU, and on the software that controls it.

4. The DMA Interface

The BEP’s Downlink Transfer Controller (DTC) is a type of Direct-Memory-Access (DMA) device. It transfers buffers of data from the BEP’s bulk (*i.e.*, uncached) memory to the RCTU telemetry serial port interface logic which handles the insertion of ACIS time-stamps into the first 32-bits of science data of each Science Telemetry Frame. The logic also provides a 2-deep 32-bit word FIFO between the DTC and the RCTU. Once the last word of a block has been transferred, the logic generates a Downlink Interrupt. Assuming that peak transfer rate out of the FIFO is 128Kbps, the BEP software has about 0.5 milliseconds to handle the interrupt and start a new transfer. Otherwise, a gap is introduced, and the hardware will respond to RCTU requests with an 8-bit pattern whose value is 0xb7 in hexadecimal (see Appendix A and Applicable Document 3 for further details).

5. Similar Previous Anomalies

All ACIS output recorded on board and downlinked since March 2000¹ was scanned for similar anomalies, *i.e.*, gaps in packet sequence numbers and packets followed by unexpected bytes, but not associated with missing telemetry frames or ACIS reboots. The resulting anomalies are shown in Table 3.

Table 3: Missing packets not associated with telemetry gaps

1	d5/acis2c	180	873	2000-03-18 18:41:17.269	5522	1	0	0		Bad hdr
2	d6/acis3a	33	62084	2000-04-15 11:19:53.945	59475	0	0	4		Bad word
3	d6/acis3a	33	62084	2000-04-15 11:33:28.308	59570	1	1	0		Bad word
4	d6/acis3c	111	551	2000-05-14 12:15:34.470	23710	1	1	4		Bad word
5	d7/acis4a	32	326	2000-06-25 11:14:55.333	45910	1	1	8		
6	d8/acis5b	77	2306	2000-09-20 00:28:33.395	34007	34	1	385		Seq jump
7	df/acis8b	69	1988	2001-05-12 18:23:10.130	51136	1	8	14344	2048	
8	dg/acis9a	18	1622	2001-06-23 21:21:45.476	27148	1	1	0		Bad hdr
9	dg/acis9b	137	2017	2001-07-24 15:13:04.831	59445	2	1	152		
10	dk/acis13b	67	3344	2002-05-01 23:18:20.744	43676	2	1	36		
11	dk/acis13e	194	3008	2002-06-16 23:34:57.214	50320	1	4	7176	2048	
12	dm/acis15c	92	61019	2002-09-26 13:46:07.795	28883	1	1	0		Bad hdr
13	dm/acis15c	92	61019	2002-09-26 19:23:15.913	33616	1	1	0		Bad hdr
14	dm/acis15c	92	61019	2002-09-26 19:40:26.038	36866	1	1	0		Bad hdr
15	dn/acis16e	206	3744	2003-01-11 18:42:39.708	52385	1	5	14400	3088	
16	do/acis17a	2	4374	2003-01-17 22:09:25.029	25451	1	5	5112	1024	
17	dp/acis18a	46	3654	2003-03-28 21:03:40.769	25970	1	8	15368	2048	
18	dp/acis18a	85	60790	2003-04-02 16:06:48.885	44950	1	1	0		Bad hdr
19	dq/acis19a	13	4276	2003-06-06 16:00:34.217	4033	1	2	4096	2048	
20	ds/acis21b	90	3966	2003-11-30 05:19:07.289	8951	1	7	14308	2048	
21	dt/acis22d	152	5015	2004-03-02 09:54:56.044	37016	1	2	3080	2048	
22	dv/acis24b	52	4505	2004-06-04 21:30:56.570	62456	1	1	0		Bad hdr
23	dx/acis25c	76	5357	2004-08-14 15:01:51.517	48423	1	1	2048	2048	
24	dx/acis25d	114	4816	2004-08-31 10:06:50.794	28178	1	4	7176	2048	
25	dA/acis28a	20	6180	2005-01-14 00:53:43.873	35534	1	1	0		Bad hdr
26	dB/acis29b	53	5645	2005-03-30 02:57:57.423	31832	1	3	4104	2048	

Of the 26 anomalies, 12 occurred at times when a housekeeping packet (either DEA or software) was expected but not output. Two other anomalies, cases 15 and 16 in Table 3, were recognized by their garbled output, but were not accompanied by gaps in DEA or Software housekeeping. All 14 anomalies show strong similarities to case 26, *i.e.*,

- the BEP output from 2 to 3482 4-byte words, beginning at a seemingly random 4-byte address within the BEP storage area reserved for telemetry buffers.
- within the anomalous output, packet headers occurred at intervals of either 1024, 2048, or 3088 bytes, as they do within the telemetry buffers assigned to DEA housekeeping, science, and software housekeeping, respectively (see Table 4). It is perhaps relevant

1. This was the most recent time that the Chandra OBC was rebooted, and the VCDU counter reset. Earlier telemetry could be processed, but with a separate time fence-post file after each OBC reboot.

that, when a housekeeping packet is missed, the anomalous output comes from within science buffers, whereas cases 15 and 16, in which no housekeeping is missed, the output comes from within housekeeping buffers.

Table 4: BEP telemetry buffer assignments

bepStartupMessage	Startup	4096	1
commandEcho	CmdManager	2048	4
bepExecuteReply, bepReadReply fepExecuteReply, fepReadReply pramReadReply, sramReadReply	MemoryServer	4096	4
swHousekeeping	SwHousekeeper	3088	8
deaHousekeepingData	DeaHousekeeper	1024	8
dataCcBiasMap, dataTeBiasMap	BiasThief	4096	20
<i>everything else</i>	ScienceManager	2048	400

- ignoring the anomalous output, the event counts within the expected science packets were consistent between data and exposure packets—no science packets appeared to be missing.

In case 6, (“Seq jump” in the rightmost column of Table 3) the packet sequence numbers jump from 34007 to 34042 and the exposure numbers jump from 6917 to 6920 without any telemetry gap. The last packet before the gap is garbled—it is a *dataTeVeryFaint* packet whose events are not in ascending *ccdRow* order, so its “real” length is suspect. Including that packet, there are only 385 bytes of telemetry before the (totally normal) packet numbered 34042 and the truncated packet 34043. The following 36 minor frames are missing, so it is tempting to treat this as an extreme case of telemetry corruption.

The remaining 11 anomalies were caused by corruptions in telemetry, either to fields in a packet header (*ibid.* “Bad hdr”) or to the pad bytes following a packet (*ibid.* “Bad word”).

Three anomalies, cases 13, 14, and 26, occurred at times when Chandra was in real-time contact with the ground. In the case of item 26, the recorded and real-time data were identical, and it was this that originally persuaded us that the anomaly was confined to ACIS itself, rather than mere telemetry corruption. In the case of cases 13 and 14, the packets with damaged headers in recorded telemetry were undamaged in real-time telemetry, but there were frequent recorded-*vs.*-real-time mis-matches around that time, and in a majority of cases it was the real-time data that seemed to have been corrupted. In any case, the corruption in cases 13 and 14 was unlikely to have originated within ACIS.

6. Flight Software Analysis

A preliminary version of this Memo, consisting of sections 1–5, was circulated to the ACIS SI team, including several engineers who were responsible for the original ACIS design (for details, see Appendices A–C). Jim Francis, the BEP flight software architect, responded as follows:

From: francis@payload.com
 Subject: Re: ACIS anomaly during OBSID 5645
 Date: April 25, 2005 8:46:12 AM EDT
 To: pgf@space.mit.edu
 Cc: dag@elfelectronics.com

It's looking to me like TelemetryManager::post()/serviceDevice() isn't properly protected against re-entry when there is nothing actively being

transmitted by the telemetry system (i.e. `TelemetryManager::curPkt` is initially NULL).

If a thread calls `TelemetryManager::post()` when the DMA is idle, `TelemetryManager::curPkt` is null. So it calls `serviceDevice()`. If the 1st thread is preempted by another thread before it has a chance to assign `TelemetryManager::curPkt` (see `TelemetryManager::serviceDevice()`), and the 2nd thread calls `TelemetryManager::post()`, the 2nd thread will fall-through to advance the sequence number and initiate a telemetry transfer. If the 2nd thread then suspends, the original 1st thread will grab the next sequence number, and clobber the ongoing downlink transfer with the new one. I'm not sure of the hardware behavior in this situation. If the hardware behaves well, then this is probably a red-herring. If the hardware gets confused, it might produce the garbage you're observing.

Our "startTransfer()" function:

- clears the CTL_DNLKENB bit in the BEP's control register
- sets the DTC Start register to the starting page of the buffer to send
- sets the DTC End register to the end of the buffer we want to send (looks like it's aligned to page boundary)
- sets the CTL_DNLKENB bit in the BEP's control register

I believe that downlink interrupts are enabled throughout the activity.

After reading this, Dorothy Gordon, the BEP hardware architect, concurred:

```
From:      dag@elfelectronics.com
Subject:   Re: ACIS anomaly during OBSID 5645
Date:      April 27, 2005 9:26:25 PM EDT
To:        pgf@space.mit.edu, francis@payload.com
```

If two software threads collide, one of the actions might be that the "Page Register" is changed mid-Transfer. This may result in some of the symptoms that you're seeing (i.e. the jumbling of science and engineering data).

Another action might be that the end-register is modified mid-transfer. This would result in the packet header length field not matching the actual packet length. Another thing to consider might be single event upsets. But if none of the other state machines and registers appear to randomly upset, I doubt that just this one would.

In further correspondence, Dorothy stated that the maximum DMA output length was 16 kbytes (12 bit length of 4-byte words), although the specification called for 8 kbytes. Also, the input address is mapped into the BEP GP memory. This implies that the worst that could happen is for no more than 16 kbytes to be output from GP memory, which will have no further impact on the instrument. The two serious failure modes are thereby totally excluded—one in which the contents of memory-mapped FEP memory would be read, possibly causing a FEP latch-up¹, and the other in which the DMA is asked to output an enormously long "packet".

7. A Software Patch

Assuming that the anomaly was caused by the non-reentrant routine described in Section 6., a *tlmbusy* patch (see Fig. 2) was prepared that prevents the `TelemetryManager::post()` routine being called by two tasks simultaneously. The `taskManager.forbidPreempt()` and `taskManager.permitPreempt()` calls will prevent a second task from calling `post()` until after the first has determined that `curPkt` is NULL, i.e., that the DMA is quiescent, and has returned from `serviceDevice()` with `curPkt` initialized.

1. Simultaneous FEP event-processing, pixel thresholding, and memory-mapped reading from the BEP can trigger an anomaly in FEP firmware that puts a FEP FPGA into a "latch-up" state which continues until FEP power is cycled (see Applicable Document 4).

```

void Test_TlmManager::post(TlmPkt*pkt)
{
    DebugProbe probe;

    // ---- Place packet onto queue ----
    sendQueue.enqueuePkt (pkt);

    // ---- Prevent task preemption ----
    taskManager.forbidPreempt();

    // ---- If no transfers in progress, start one up ----
    if (curPkt == 0)
    {
        serviceDevice (0);
    }

    // ---- Allow task preemption again ----
    taskManager.permitPreempt();
}

```

Figure 2: Changes (in red/**boldface**) to *TlmManager::post()* (see Appendix C) to prevent the non-reentrant *serviceDevice()* method from being called simultaneously by multiple tasks. Note that *serviceDevice()* is only called from *post()* when the packet queue is empty; otherwise, it is called from within the BEP interrupt handler when the previous DMA operation has completed. Since interrupts are disabled within the handler, *serviceDevice()* cannot, in this instance, be called from multiple threads.

It has proven much harder to validate the patch than to verify it. While the latter requires only a short run on the ACIS Engineering Unit to verify that the *tlmbusy* patch does not interfere with event processing, validation requires that anomalies must be reproduced using the standard patch load, after which the test must be rerun with the additional *tlmbusy* patch to verify that the anomaly no longer recurs. Both tests—without and with the new patch—must last for sufficient lengths of time to demonstrate that (a) the number of anomalies without the patch and (b) the absence of any anomaly with the patch, are consistent with the hypothesis that the patch prevents the anomaly, to a given level of confidence.

If the anomaly occurs with Poisson statistics at a mean rate of R per second, the probability that it will not recur within a further t seconds is given by $\exp(-Rt)$. If we were to emulate the on-orbit environment and wanted to be 99.9% certain that the patch cured the anomaly, we would be forced to test the engineering unit for $-5.03 \log_e(0.001)/14 = 2.48$ years, which is unacceptable. Instead, we ran tests on the engineering unit in a manner that maximized the rate of production of telemetry packets and the number of simultaneous tasks that wrote them. This was achieved by the following choices:

- Timed-exposure graded mode to minimize the science packet header lengths.
- 100-column sub-frame readout from a single (simulated) CCD with 0.3 second exposures to maximize the science packet rate with 4 “events” in the image loader.
- The maximum DEA housekeeping readout rate of one packet per second.
- The test was conducted by an *expect* script (see Fig. 3) that monitored the downlink data stream. Whenever an *exposureTeGraded* packet was received, the script issued a *readBep* command to report the contents of 4 bytes of BEP memory. This caused three BEP tasks to write telemetry packets: the command task to echo the command, the memory task to report the 4 bytes, and the software housekeeping task to write a “user pseudo-packet”.

With 5 tasks frequently calling the *TlmManager::post()* routine, the anomaly was found to occur 8 times in 622,101 seconds, *i.e.*, once per 77,800 seconds, compared with the on-orbit rate of once per 1.13×10^7 seconds. At this rate, the *tlmbusy* patch would be validated to a 99.9% confidence level provided no anomaly were found after 540,000 seconds of testing.

```

set timeout 300
expect -i $tlm_id {
    -re "exposureTeGraded.*exposureNumber=(\[0-9a-fx]+\).*\n" {
        if { [ expr $expect_out(1, string) ] < 1000000 } {
            send -i $cmd_id "read 0 0xa000e5e0 1\n"
            exp_continue
        }
    }
    timeout { fail "No exposure record" }
}

```

Figure 3: Extract from an *expect* script used to reproduce the telemetry anomaly on the ACIS engineering unit. The output from the *psci* procedure is inspected. When an *exposureTeGraded* packet is received, a *readBep* command is issued. The loop terminates either when the *exposureNumber* value reaches 1000000 or when no exposure packet is received within 300 seconds. In practice, the *expect* script is more complicated than this since the engineering unit cannot receive commands at this rate. Code was added to keep track of the number of *commandEcho* packets received, so that the commanding rate could be maximized by a simple feedback loop. The actual scripts are reproduced in Appendix D.

The *tlmbusy* patch was then installed and the test was repeated for 585991 secs; no anomaly was detected. Next, the patch was removed and the test was run for a further 623229 seconds, during which the anomaly occurred 5 times. Adding these 5 to the 8 anomalies seen in the previous test reduced the estimated anomaly rate, so the patch was reapplied and a final test was run for 624406 seconds. Seeing no anomalies, the probability that this was *not* due to the patch is 0.0000033, *i.e.*, the patch has been validated to a confidence level of 99.9997%.

8. Conclusions and Recommendations

The repeatability of the anomaly under predetermined test conditions—when multiple tasks make frequent calls to the *TlmManager::post()* method—and the disappearance of the anomalies when the *tlmbusy* patch is applied, provide very firm evidence that the cause of the in-flight anomalies is understood, and that the patch provides the means for preventing them.

A careful inspection of the non-reentrant code shows that the effect of the anomaly is to pass a bad input address and data length to the BEP DMA controller. The input address is restricted to the M-bus, *i.e.*, BEP bulk memory, and the length cannot exceed 16 kbytes. The output always goes to the BEP's serial digital interface port. The anomaly cannot result in spurious data being written to other BEP (or FEP) memory locations, or being read from anywhere other than bulk memory, and cannot therefore cause other BEP actions.

The question remains whether the patch should be applied to the flight instrument at this time. There is no indication that the anomalies have become more frequent since launch—the rate remains at one per ~110 days. In the worst case, a legitimate packet will be dropped and a spurious 16 kbytes will be output instead, but no packets output as a result of the daily load can be considered critical.¹ The only packets whose loss would affect CXC operations are output during real-time ACIS testing and patch loading, when the chance of the anomaly occurring is absolutely minimal.

The ACIS Operations Team discussed these recommendations on July 25th and concluded that there is no urgency in responding to this anomaly. The *tlmbusy* patch should be added to MIT's configuration control, and included in the next release, if one becomes necessary for other reasons. Meanwhile, the diagnostic logs from rou-

1. It is possible for corrupted packets to cause the ground processing system to halt, but it is a relatively simple matter to remove the offending telemetry frames and reprocess, as was done with OBSID 5645.

tine quick-look processing should be monitored for recurrences of the anomaly, and the situation should be reviewed if the anomaly rate appears to be increasing.

9. Applicable Documents

1. ACIS Software Problem Report M05040701, April 7, 2005, available online at <http://acis.mit.edu/axaf/spr/prob0138.html>.
2. ACIS Software IP&CL Structure Definitions, MIT 36-53204.0204 rev. N, March 15, 2001, online at <http://acis.mit.edu/acis/ipcl/>.
3. ACIS DPA Hardware Specification & System Description, MIT 36-02104, rev. C, October 5, 1995, online at <http://acis.mit.edu/axaf/dpa/>.
4. ACIS Software Problem Report M00062901, April 12, 2000, online at <http://acis.mit.edu/axaf/spr/prob0133.html>.
5. Object-Oriented Analysis and Design with Applications, by Grady Booch, 2nd Edition, Addison-Wesley, 1993 (ISBN: 0805353402).
6. ACIS Software Test Tools, MIT 36-55001, rev. 3.1, June 20, 1997, online at <http://acis.mit.edu/ttools/testtools.pdf>.

10. Abbreviations

ACIS	Advanced CCD Imaging Spectrometer
AXAF	Advanced X-ray Astronomy Facility (now Chandra)
BEP	(ACIS) Back-End Processor
CASE	Computer-Aided Software Engineering
CCD	Charge-Coupled Device
D-Cache	(R3000) Data Memory
DDTS	(Rational™) Distributed Defect Tracking System
DEA	Digital Electronics Assembly (a.k.a. ACIS video section)
DMA	(R3000) Direct Memory Access
DPA	(ACIS) Digital Processor Assembly (BEPs + FEPs)
DSN	Deep Space Network
FEP	(ACIS) Front-End Processor
FIFO	First-In-First-Out (stack)
FPGA	Field-Programmable Gate Array
GP	General Purpose (random access memory)
I-Cache	(R3000) Instruction Memory
MSB	Most-significant Bit
OBC	(Chandra) On-Board Computer
OBSID	(Chandra) Observation Identifier
PCB	Printed Circuit Board
R3000	(FEP and BEP) Radiation-Hardened Processor
RAM	Random-Access Memory
RCTU	Remote Control Telemetry Unit
SCET	Spacecraft Event Time (UTC)
SD	Serial Digital (output from the ACIS DPA)
SEU	Single-Event Upset
SOT	(Chandra) Science Operations Team
SPR	(ACIS) Software problem report
SSR	(On-board) Solid-State Recorder

Appendix A. The BEP-to-RCTU Interface

The following description is taken from §2.1.2.6.1 of Applicable Document 3.

The heart of the serial downlink interface is a DMA controller which transfers data from the M-bus bulk memory to a parallel to serial converter register which is memory mapped onto the M-bus. The RCTU clock shifts out at a bit rate of 128 Kbits/sec (7.8 microseconds/bit); see the RCTU Users' Guide for further details. The downlink data register is 32-bits wide, so that 4 bytes (one telemetry word = 8 bits) can be loaded into the register simultaneously, giving a minimum register turnaround time of 250 ns. (The turnaround time could be considerably longer since there are "dead" periods when the clock is absent, and the DPA is not feeding the link.)

A dedicated state machine resident in a FPGA will serve as the "downlink transfer controller" (DTC). The downlink transfer controller contains a 12 bit start_addr register, a 12-bit end_addr register, and a 6 bit page_addr register. An M-Bus address is generated by combining a 12 bit counter output with the page address. Each M-Bus read clocks the counter to the next longword address and simultaneously writes a longword to the DTC.

Upon detecting a DNLKTRENB (written to the BEP control register), the DTC loads an internal 12-bit counter with start_addr and fills two 32-bit buffers (two longwords indexed by the address counter). It then transmits serialized telemetry to the RCTU serial data I/F at a rate established by the RCTU, filling its 32-bit buffers on an alternating as-needed basis. An end-of-transfer signal and Downlink Interrupt is generated by the DTC when an end_addr has been detected. The hardware automatically disables the DTC (deasserts DnlkTransEnb) after fetching the word indexed by end_addr from main memory. The CPU can abort a DTC transfer by clearing the DNLKTRENB bit. In this case, the DTC will send out the pending 32-bit word and exit; no DNLKINT will be generated. (A watchdog or commanded reset will have a similar effect, as they both deassert DNLKTRENB.)

NOTE: the DTCENB control register bit is deasserted one clock cycle before the assertion of DNLKINT. Additionally, several clock cycles transpire before the CPU recognizes the interrupt. Therefore, at the trailing edge of a DTC packet transfer, polling the control register may indicate that the transfer has finished before the DNLKTRENB interrupt has gotten to the CPU. (There is a ~300 ns window where this could occur. This is only a conflict if the DTC is run simultaneously in polled and interrupt driven modes.)

The RCTU downlink I/F state machine receives data from the DTC, shifts out the 32-bit word (low byte first, MSB first within each byte). The RCTU downlink I/F state machine is responsible for detecting the Science Header Pulse, latching the timestamp counter, and shifting it out as the first four bytes following the pulse detection. When the DTC is disabled, the hardware shifts out the timestamp counter in the usual telemetry slot. A fill pattern (B7 hex) is inserted for science data when the DTC is disabled.

The following requirements are imposed by DPA dynamic operation and the flight software:

- DMA turnaround time of 0.5 ms. (The flight software must service a downlink interrupt, and set up for the next DTC transfer or it may miss sending (at the rate of one word per 0.25 ms) telemetry, in which case a fill-pattern will be inserted by the hardware.)
- DMA must not stall the bus (< than 5 ns per M-bus grant)
- Predefined fill pattern (B7 hex) to be supplied when the processor is not active, or is not driving the link
- Hardware must insert the running timestamp count into fixed positions in the telemetry stream (each four bytes following the detection of a Science Header Pulse).
- All DTC transfers must be longword aligned with respect to M-Bus memory. (Since the RCTU operates in byte transfer mode, the longword structure "floats" within the RCTU format, and will be reconstructed on the ground by the AXAF Science Center.)
- Startup and boundary characteristics shall be well defined: the falling edge of the next byte requested by the RCTU will activate the first "packet" transfer following the assertion of DNLKTRENB.

The maximum DTC packet size is 8192 bytes; the minimum packet size is 8 bytes.

Appendix B. Telemetry Software Classes

The BEP flight software used an object-oriented design called the Booch Method (see Applicable Document 5) and implemented in the Rational Rose™ CASE tool. In this approach, the relationships between classes is described by a Class Diagram, and particular scenarios by Object Diagrams (e.g., Fig. 6). Classes and their methods are enclosed in bubbles. Arrows point from subclass to superclass; lines terminating in dots and squares represent methods of one class using objects of the other, etc.

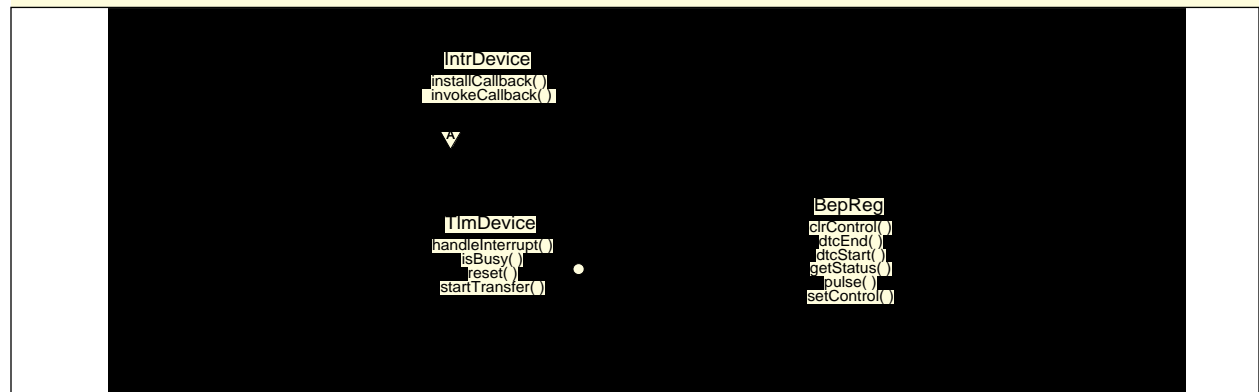
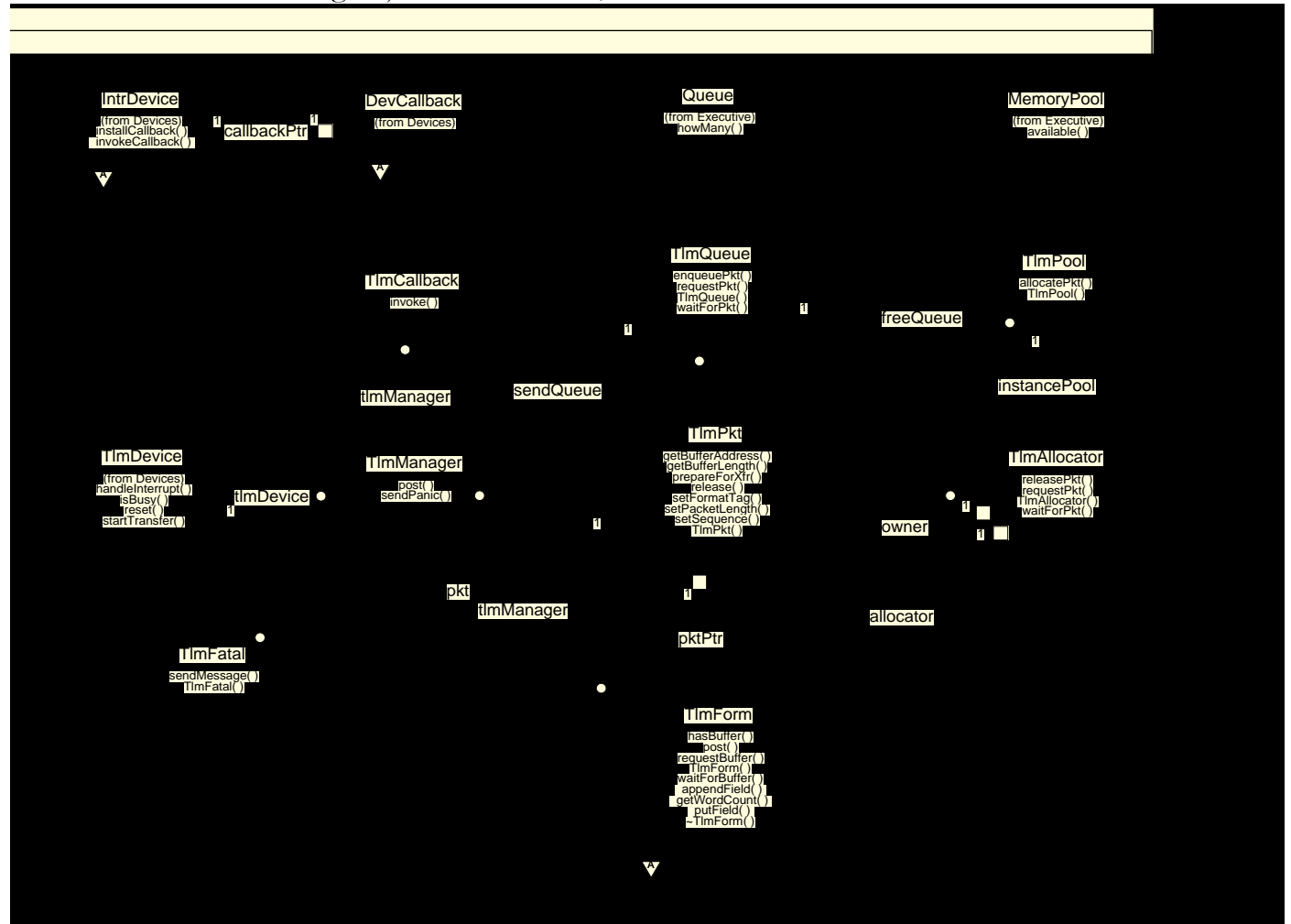


Figure 5: Class diagram for a Telemetry Device.

Appendix C. A Telemetry Scenario: Writing a Packet

Any routine in any BEP task can write a packet. The general scheme is shown in Fig. 6. The following steps are executed.

1. A client decides to create and send a telemetry packet. It declares *tlmForm*, whose constructor initializes the state of the instance, and zeros its packet instance pointer.
2. The client then waits for a telemetry packet buffer to become available using *tlmForm.waitForBuffer()*.
3. *tlmForm.waitForBuffer()* in-turn calls its allocator's member function, *tlmAllocator.waitForPkt()* to attempt to reserve an unused *TlmPkt* instance.
4. *tlmAllocator.waitForPkt()* then invokes *freeQueue.waitForPkt()*.
5. And finally, *freeQueue.waitForPkt()* invokes its protected member function, *Queue::waitForItem()* to block until a telemetry packet pointer becomes available. Once the allocator returns, *tlmForm* retains the acquired packet pointer until it is either posted, (see step 11), or until *tlmForm* is destroyed. If *tlmForm* is destroyed prior to the packet being posted, *tlmForm*'s destructor releases the packet back to its allocator. If the packet is posted to the *tlmManager*, *tlmForm* is no longer responsible for the packet, and it is *tlmManager*'s responsibility to ensure that the packet is released.
6. Once a packet has been obtained by *tlmForm*, it sets the packet's format tag using *tlmPkt.setFormatTag()*.
7. *tlmForm* then obtains and caches the packet's buffer address and length, for use later when the client writes fields into the packet, using *tlmPkt.getBufferAddress()* and *tlmPkt.getBufferLength()*.
8. The client then writes zero or more fields into the telemetry packet buffer, using member functions supplied by the format-specific *tlmForm* instance. The *tlmForm* instance uses the inherited *TlmForm::putField()* and *TlmForm::appendField()* functions to write the data into the packet's buffer.
9. Once the client has completed the packet, it tells the *tlmForm* to transfer the packet out of the instrument using *tlmForm.post()*.
10. *tlmForm.post()* then computes the number of words written into the packet, using *tlmForm.getWordCount()* and passes the result the packet, using *tlmPkt.setPacketLength()*.
11. *tlmForm.post()* then invokes *tlmManager.post()*, passing the address of *tlmPkt*.
12. ***tlmManager.post()* sets the packet's sequence number using *tlmPkt.setSequence()* and then places the packet address on the end of its queue, using *sendQueue.enqueuePkt()*, which then uses *Queue::enqueue()*.**
13. Once *tlmManager.post()* returns, *tlmForm* zeros its local pointer to the packet. This ends *tlmForm*'s responsibility concerning *tlmPkt*, and the client can destroy (*~TlmForm*) *tlmForm*, without affecting the posted *tlmPkt*.
14. Once the telemetry device is ready, *tlmManager* gets the next telemetry packet to send from the queue using *sendQueue.requestPkt()*.
15. *tlmManager* then copies packet header information and obtains the packet's buffer address and length using *tlmPkt.prepareForXfr()*.
16. *tlmManager* then tells the telemetry device to transfer the packet, using *tlmDevice.startTransfer()*.
17. Once the transfer completes, *tlmDevice*'s interrupt handler invokes the installed telemetry callback, *tlmCallback.invoke()*.

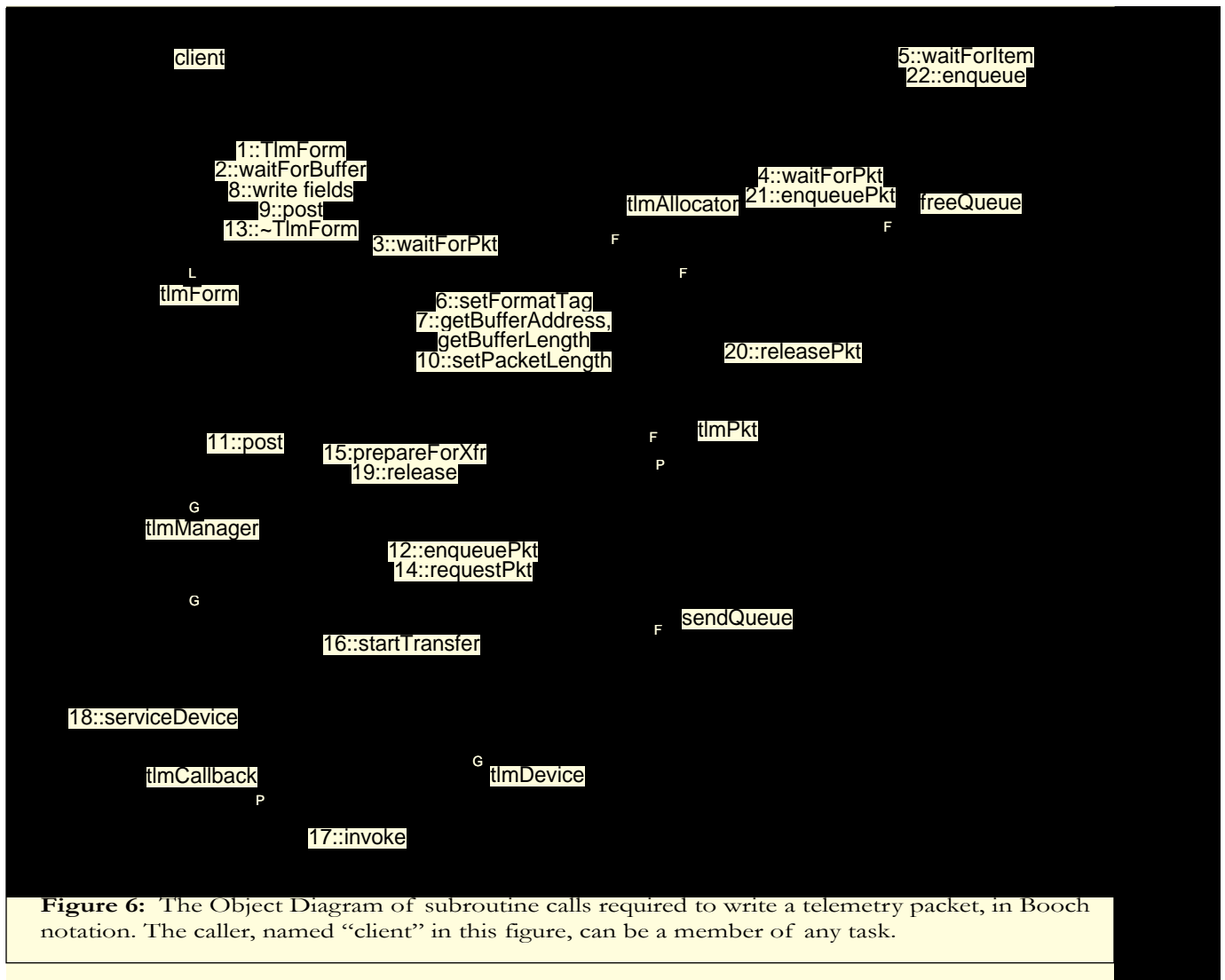


Figure 6: The Object Diagram of subroutine calls required to write a telemetry packet, in Booch notation. The caller, named “client” in this figure, can be a member of any task.

18. *tlmCallback.invoke()* then invokes the *tlmManager.serviceDevice()* to release the packet’s buffer and start the next transfer (if a packet is available).
19. *tlmManager.serviceDevice()* function then releases the completed packet using *tlmPkt.release()*.
20. *tlmPkt.release()* in-turn calls its owner’s *TlmAllocator::releasePkt()*.
21. *tlmAllocator.releasePkt()* then invokes *freeQueue.enqueuePkt()*.
22. Finally, *freeQueue.enqueuePkt()* uses the protected function, *Queue::enqueue()* to place the available packet onto the end of the queue.

The non-reentrant part occurs in Step 12 above, since *serviceDevice()* should only be called if there is no DMA activity under way, but this is indicated by giving *curPkt* a zero value after a call to *curPkt->release()* and *sendQueue.requestPkt()*. A second task calling *TlmManager.post()* and still finding zero *curPkt*, will go ahead and call *serviceDevice()* and trigger the anomaly.

```

void TlmForm::post () {
    // ---- Get and set length of packet ----
    unsigned count = getWordCount();
    pktPtr->setPacketLength (count);
    // ---- Tell telemetry manager to send packet ----
    tlmManager.post (pktPtr);
    // ---- No longer responsible for the packet ----
    pktPtr = 0;
}

void TlmManager::post (TlmPkt*pkt){
    // ---- Place packet onto queue ----
    sendQueue.enqueuePkt (pkt);
    // ---- If no transfers in progress, start one up ----
    if (curPkt == 0) {
        serviceDevice (0);
    }
}

void TlmManager::serviceDevice (IntrDevice*devptr) {
    // ---- If just finished transfer, release packet ----
    // NOTE: If we do not zero the packet pointer after the
    // release, we prevent post() from attempting to start a
    // new transfer during this block of code. This removes
    // the need for disabling interrupts here and in post().
    if (curPkt != 0) {
        curPkt->release (); // Release packet for re-use
    }
    // ---- Attempt to get next packet to send ----
    curPkt = sendQueue.requestPkt ();
    // ---- If non-zero, start a new transfer ----
    if (curPkt != 0) {
        // --- Set the packet sequence number ---
        curPkt->setSequence (curSequence);
        curSequence++; // Advance sequence number
        // --- Ready packet and get transfer address & count ----
        unsigned* xfraddr; // Transfer address
        unsigned xfrcnt; // Number of words to transfer
        curPkt->prepareForXfr (xfraddr, xfrcnt);
        // --- Start transferring the packet ---
        tlmDevice.startTransfer (xfraddr, xfrcnt);
    }
}

TlmPkt* TlmQueue::requestPkt() {
    TlmPkt* pkt = 0; // Packet pointer
    // ---- Attempt to dequeue, zero return value if it fails ----
    if (Queue::dequeue(&pkt) == BoolFalse) {
        pkt = 0; // None ready
    }
    // ---- Return dequeued packet, or 0 if none ready ---
    return pkt;
}

```

Appendix D. The expect script “runtest.tcl”

Two copies of the *runtest.tcl* script exist: one to run with the *standard.bcnd* patches, the other with *standard.bcnd* and with the new *tlmbusy.bcnd* patch (denoted **original** and **patched**). The script uses several tools from the ACIS patch test environment (see Applicable Document 6).

```

#!/bin/env expect
#
# Simulation of telemetry anomaly -- without tlmbusy patch
#

puts "Welcome to tlmbusy/testsuite/bugfix-hw/runtest.tcl"

# ---- Get runtime parameters ----
set basedir [lindex $argv 0] ; # patch base directory
set tools [lindex $argv 1] ; # tool directory
set patchdir [lindex $argv 2] ; # patches under test
set server $env(ACISSERVER) ; # lrtcu/ctue server
set port $env(PORT) ; # lrtcu/ctue server command port

# ---- Embed procedure library ----
source $basedir/$tools/lib/lib-exp/runtest_support.tcl

# ---- Start command pipe ----
spawn /bin/sh -c "bcnd | cclient $server $port"
set cmd_id $spawn_id ; # copy the process ID

# ---- Start telemetry pipe ----
spawn /bin/sh -c "filterClient -h $server | psci -m -u"
sleep 1 ; wait for client to start

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcnd\
                 $basedir/$tools/share/opt_printswhouse.bcnd\
                 $basedir/$tools/share/opt_dearepl.bcnd\
                 $basedir/$patchdir/standard.bcnd\
                 $basedir/$patchdir/tlmbusy.bcnd"
warm_boot

# ---- Power on FEP and CCD ----
power_on_boards "0 10 10 10 10 10"

# ---- While powering up, load the Bias Image ----
system "make bias"

# ---- Wait for FEP to finish powering ----
expect {
    -re ".*SWSTAT_FEP_EXECMEM: 0\[\r\n]" {}
    timeout {}
}

# ---- Start DEA housekeeping ----
send -s -i $cmd_id "load 0 dea 4 deablk.cmd\r"
command_echo 1 13 "load dea"
send -i $cmd_id "start 0 dea 4\r"
command_echo 1 18 "start dea housekeeping"

```

```

# ---- Load parameter block ----
send $cmd_id "load 0 te 4 teblk.cmd\r"
command_echo 1 9 "load te"

# ---- Start the Bias Run ----
send -i $cmd_id "start 0 te bias 4\r"
command_echo 1 15 "start bias run"
set timeout 600
wait_stop_science

# ---- Load the Event Image ----
system "make image"

# ---- Start the Science Run ----
send -i $cmd_id "start 0 te 4\r"
command_echo 1 14 "start science run"

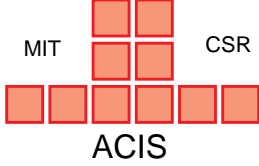
set ncmd          0 ; # command packet counter
set nread         0 ; # number of commands unacknowledged
set cmdmax        5 ; # maximum commands unacknowledged
set expo          0 ; # current exposure packet number
set nexpo        20000000 ; # limit to number of exposures

# ---- Examine the telemetry ----
expect {
  -re "commandEcho.*commandOpcode=3\[\r\n]" {
    if { $nread > 0 } {
      set nread [expr $nread - 1]
    }
    exp_continue
  }
  -re "exposureTeGraded.*exposureNumber=(\[0-9a-fx]+\).*\n" {
    set expo [ expr $expect_out(1,string) ]
    if { $expo < $nexpo } {
      if { $nread <= $cmdmax } {
        set ncmd [expr ($ncmd + 1) % 65536]
        incr nread
        send -i $cmd_id "read $ncmd 0xa000e5e0 1\r"
      }
      exp_continue
    }
  }
  # ---- fall through to stop the run ----
}
timeout {
  fail "No exposure record"
}
}

# ---- stop the science run ----
send -i $cmd_id "stop 0 science\r"
command_echo 1 19 "stop science"
set timeout 30
science_report 1 "science report"

# ---- Report end of run ----
pass "$expo exposures received"

```

		ENGINEERING CHANGE ORDER		<u>ECO No.</u> <u>36-1033</u>
CENTER FOR SPACE RESEARCH MASSACHUSETTS INSTITUTE OF TECHNOLOGY				
DWG. NO.	NEW REV.	DRAWING TITLE		
36-58030.29	A	Flight S/W patch to prevent BEP telemetry packet loss		
REASON FOR CHANGE: Telemetry packets are enqueued in the BEP via a call to the <code>TlmManager::post()</code> method. If this routine is called by one task while it is still processing a call from another task, the first packet will be truncated and a random block of data from within the telemetry buffer area will be telemetered in place of the second packet. The problem can be prevented by cancelling task switching while <code>TlmManager::post()</code> is running.				
DESCRIPTION OF CHANGE: Replace <code>TlmManager::post()</code> with a version that calls <code>taskManager.forbidPreempt()</code> before calling <code>serviceDevice()</code> and calls <code>taskmanager.permitPreempt()</code> afterwards.				
	SIGNATURE	DATE	REMARKS:	
ORIGINATOR	Peter Ford	08/09/07	Approved version	
MECHANICAL				
ELECTRICAL				
SOFTWARE				
STRUCTURE				
FABRICATION				
SCIENCE				
SYSTEMS ENG.				
QUALITY				
PROJ. ENGINEER				
DEPUTY PM				
PROJ. MANAGER				

1. REASONS FOR CHANGE

The CXCDS was unable to process OBSID 5645 at the first attempt on March 30, 2005, because of the presence of a single data packet of type *dataCcGraded*. Since the observation was being conducted in Very Faint Timed Exposure mode, this unexpected packet caused the processing program to halt. It was found that the *dataCcGraded* packet was followed by several kilobytes of “garbage”, consisting of partial telemetry buffers that could be recognized as already-telemetered packets.

Subsequent inspection of all ACIS telemetry since launch turned up 25 previous instances of this behavior, although the size and contents of the “garbage” differed in each case.

The problem was traced to a feature of the BEP telemetry manager. Telemetry packets are enqueued via a call to the `TlmManager::post()` method. If this routine is called by one task while it is still processing a call from another task, the first packet will be truncated and a random block of data from within the telemetry buffer area will be telemetered in place of the second packet.

The problem is prevented by cancelling task switching while `TlmManager::post()` is running, and has been implemented as the *tlmbusy* standard patch described in the current document. For details of the original analysis, see ACIS report SPR138-1.0, September 30, 2005.

2. PROPOSED CHANGE

Change the `TlmManager::post()` method:

```
void Test_TlmManager::post(TlmPkt*pkt)
{
    DebugProbe probe;
    // ---- Place packet onto queue ----
    sendQueue.enqueuePkt (pkt);
    // ---- If no transfers in progress, start one up ----
    if (curPkt == 0) {
        serviceDevice (0);
    }
}
```

by adding `taskManager` calls to suspend task switching within `serviceDevice()`:

```
void TlmManager::post(TlmPkt*pkt)
{
    DebugProbe probe;
    // ---- Place packet onto queue ----
    sendQueue.enqueuePkt (pkt);
    // ---- Prevent task preemption ----
    taskManager.forbidPreempt();
    // ---- If no transfers in progress, start one up ----
    if (curPkt == 0) {
        serviceDevice (0);
    }
    // ---- Allow task preemption again ----
    taskManager.permitPreempt();
}
```


Since these calls cannot be added “inline”, we must replace the entire method, which we do by defining a new public subclass of `TlmManager`.

```
class Test_TlmManager : public TlmManager
{
public:
    ~Test_TlmManager() {} ;
    void post(TlmPkt*pkt);
};
```

and we instruct the patch generator to replace the address of `TlmManager::post()` with that of `Test_TlmManager::post()` in the C++ jump table.

3. CONTROLLED SOURCES

tlmbusy	
<i>SPR138-1.0.pdf</i>	Report of the original anomaly and its subsequent investigation
<i>eco-1033.doc</i>	Engineering change order describing the <i>tlmbusy</i> patch.
<i>spr138.pdf</i>	Originating software problem report
<i>standalone.mak</i>	Makefile script to generate test patch
<i>tlmbusy.C</i>	Source code for Test_TlmManager class
<i>tlmbusy.mak</i>	Makefile script to generate standard patch
<i>tlmbusy.pkg</i>	Script to generate patch release
tlmbusy/testsuite	
<i>makebias</i>	Generate a TE bias image and copy it to the image loader.
<i>makeimage</i>	Generate a TE image with events and copy it to the image loader.
tlmbusy/testsuite/bug-hw	
<i>Makefile</i>	Run tests of the <i>fepignorebreak</i> commands.
<i>standard.bcnd</i>	Standard patch load for testing
<i>runtest.tcl</i>	Test script for 600 kilosecond run
tlmbusy/testsuite/fix-hw	
<i>Makefile</i>	Run tests of the <i>fepignorebreak</i> commands.
<i>standard.bcnd</i>	Standard patch load for testing
<i>tlmbusy.bcnd</i>	New patch for testing
<i>runtest.tcl</i>	Test script for 600 kilosecond run
tlmbusy/testsuite/smoke	
<i>Makefile</i>	Run tests of the <i>fepignorebreak</i> commands.
<i>standard.bcnd</i>	Standard patch load for testing
<i>tlmbusy.bcnd</i>	New patch for testing
<i>runtest.tcl</i>	Test script for short run

4. TESTING

Thus far, all ACIS flight software patches have been tested by explicit demonstration, *i.e.*, a science run is started on the engineering unit and stopped when the particular anomaly has been detected. The new patch is then applied and a second run is started. The job is stopped when it is clear that the anomaly has not recurred.

In the current instance, it is impractical to test in this manner since, as described in the report (SPR138-1.0), it takes many hundreds of hours of run time before the anomaly shows up, and even longer to be sure that the behavior has been prevented by the patch. We shall therefore use the report as a validation of the patch, and merely test it for a few minutes to be quite sure that it isn't affected by, or itself affects, the other patches.

The test is performed on the ACIS Engineering Unit using 6 FEPs, an image loader, and an L-RCTU interface. After setting up a *shim* process to handle I/O between UNIX and the L-RCTU, the tests are controlled by scripts written in the *expect* dialect of TCL.

4.1. Standard Test

An *expect* procedure, “*smoke/runtest.tcl*”, performs a timed-exposure science run with the *standard* and *tlmbusy* patches. The following steps are performed:

1. A command pipe is spawned down which ACIS commands will be written.
2. A telemetry pipe is spawned, terminating in the “*psci -m -u*” packet monitoring function with *expect* examining the standard output.
3. ACIS is cold-booted.
4. Software housekeeping, DEA replacement, and standard flight patches, including *tlmbusy*, are applied.
5. ACIS is warm-booted.
6. All FEPs are powered up.
7. A bias map containing the same value in each pixel of a given quadrant is written to the image loader.
8. A DEA housekeeping parameter block is sent to ACIS, calling for one readout per second. Housekeeping is restarted.
9. A *te_3x3* parameter block is sent to ACIS. It calls for one FEP to be run in faint graded mode, calling for a 100-row subarray and 0.3 second exposures.
10. A science run is started.
11. Whenever an exposure packet is generated, the *expect* script issues a *readBep* command. The aim is to stress the BEP, causing its task manager to switch between threads (Science, DEA housekeeping, Software housekeeping, Command management, Memory Management), and hence make it most likely that the anomaly will be triggered.
12. If the anomaly isn't triggered after 5 minutes of event data, a *stopScience* command will be sent.
13. If a *scienceReport* packet is received with `terminationCode = 1`, the test is determined to have passed.

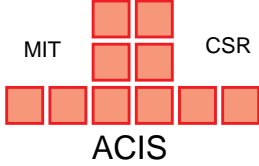
4.2. Test to Reproduce the Anomaly

This test used the same methodology as in Section 4.1, above, except that the *tlmbusy* patch was omitted from Step 4 and the *expect* script was run for an extended period. As described in Section 7 of the SPR-138 Report,¹ the test was run twice, the first time for 622,101 seconds, the second for 623,229 seconds, during which a total of 13 anomalies were recorded.

4.3. Test of the Patch

This test used the same methodology as in Section 4.1, above, except that the *expect* script was run for an extended period. As described in Section 7 of the SPR-138 Report,¹ the test was run twice, the first time for 585,991 seconds, the second for 624,406 seconds, during which no anomalies were recorded.^β

¹ This document may be downloaded from "<ftp://acis.mit.edu/pub/SPR138-1.0.pdf>".

		ENGINEERING CHANGE ORDER		<u>ECO No.</u> <u>36-1034</u>
CENTER FOR SPACE RESEARCH MASSACHUSETTS INSTITUTE OF TECHNOLOGY				
DWG. NO.	NEW REV.	DRAWING TITLE		
36-58030.30	A	Flight S/W patch to prevent BEP bus crash on FEP powerdown		
REASON FOR CHANGE: If ACIS is computing bias maps when commanded to power down its front-end processors (FEPs), it is likely to crash the back-end processor (BEP) interface bus, causing the BEP to reboot without flight software patches. Normal operations must be restored via ground command. The cause of the problem has been traced to a design flaw in the BEP flight software and this ECO describes a small patch that will fix it.				
DESCRIPTION OF CHANGE: Replace the <code>FepManager::loadBadPixel()</code> routine, which merely calls the appropriate <code>FepIo::writeBiasValue()</code> method, with one that first calls <code>FepManager::ieEnabled()</code> to check whether the FEP is powered up, and only calls <code>FepIo::writeBiasValue()</code> if it is.				
	SIGNATURE	DATE	REMARKS:	
ORIGINATOR	Peter Ford	08/09/07	Accepted version	
MECHANICAL				
ELECTRICAL				
SOFTWARE				
STRUCTURE				
FABRICATION				
SCIENCE				
SYSTEMS ENG.				
QUALITY				
PROJ. ENGINEER				
DEPUTY PM				
PROJ. MANAGER				

1. REASONS FOR CHANGE

If ACIS is computing bias maps when commanded to power down its front-end processors (FEPs), it is likely to crash the back-end processor (BEP) interface bus, causing the BEP to reboot without flight software patches. Normal operations must be restored via ground command. This situation has arisen 3 times since launch. The cause of the problem has been traced to a design flaw in the BEP flight software and this ECO describes a small patch that will fix it.

During execution of SCS107, typically due to high background radiation, ACIS is powered down. Science telemetry reports that the flight s/w version number is 11, whereas typical values (depending in the patch combination) are 30 or higher, indicating that the BEP rebooted itself. Subsequent inspection of the recorded telemetry shows no *scienceReport* packet from the last science run, but a *bepStartupMessage* packet with `lastFatalCode=7` and `watchdogFlag=1`.

Since the observatory is usually in safe mode for several hours following the SCS107, there is generally sufficient time to establish a realtime contact, set the BEP's warm-boot flag, and restart it. However, this takes time and manpower.

The bus crash has been traced to a flaw in the `FepManager::loadBadPixel()` method. This routine is executed after the FEP bias maps have been created and before they are (optionally) reported in telemetry. It uses the memory-mapped interface between BEP and FEP to change those locations in the FEP bias maps that correspond to "bad" pixels or whole columns. However, unlike all other `FepManager` operations, `loadBadPixel()` does not confirm that a FEP is powered up before it writes to its map. This causes the bus crash.

2. PROPOSED CHANGE

Replace the `FepManager::loadBadPixel()` method:

```
void FepManager::loadBadPixel(FepId fepid, unsigned row,
                             unsigned col)
{
    DebugProbe probe;
    fepIo[fepid]->writeBiasValue(row, col, PIXEL_BAD);
}
```

with the following:

```
class Test_FepManager
{
public:
    virtual void loadBadPixel(FepId fepid, unsigned row, unsigned col);
    friend class Test2_FepManager;
};

void Test_FepManager::loadBadPixel(FepId fepid, unsigned row,
                                   unsigned col)
{
    DebugProbe probe;
    if (fepManager.isEnabled(fepid) == BoolTrue) {
        fepIo[fepid]->writeBiasValue(row, col, PIXEL_BAD);
    }
}
```

It is necessary to define a new class because the call to `isEnabled()` takes up too many machine instructions to be performed in an "inline" patch. The choice of `Test_FepManager` for the new class is dictated by the existing header file, *fepmanager.H*, which already includes the statement

```
friend class Test_FepManager;
```

in its definition of the `FepManager` class. Since the only existing BEP patch that defines this new class is the optional *hybrid* patch, these two patches cannot be linked together without renaming one of the `Test_FepManager` declarations. Since *buscrash* will be a standard patch, and *hybrid* will be optional, it is the latter that must change to `Test2_FepManager`, and we make provision for this in *buscrash* with the declaration

```
friend class Test2_FepManager;
```

3. CONTROLLED SOURCES

buscrash	
<i>buscrash.C</i>	Source code for the <code>Test_FepManager</code> class
<i>buscrash.mak</i>	Makefile script to generate test patch
<i>buscrash.pkg</i>	Script to generate patch release
<i>eco-1034.doc</i>	Engineering change order describing the <i>buscrash</i> patch
<i>spr140.pdf</i>	Originating software problem report
fepignorebreak/testsuite	
<i>makebias</i>	Generate a bias image and copy it to the image loader
fepignorebreak/testsuite/bug-hw	
<i>Makefile</i>	Run a test without the <i>buscrash</i> patch
<i>runtest.tcl</i>	<i>expect</i> script to demonstrate a BEP bus crash
fepignorebreak/testsuite/fix-hw	
<i>Makefile</i>	Run a test with the <i>buscrash</i> patch
<i>standard.bcnd</i>	Standard patches, including <i>buscrash</i>
<i>runtest.tcl</i>	<i>expect</i> script to demonstrate prevention of BEP bus crash

4. TESTING

All tests are performed on the ACIS Engineering Unit using one FEP, an image loader, and an L-RCTU interface. After setting up a *shim* process to handle I/O between UNIX and the L-RCTU, the tests were controlled by scripts written in the *expect* dialect of TCL.

4.1. Reproduce Test

An *expect* procedure, “*bug-hw/runtest.tcl*”, performs a timed-exposure science run with the *standard* and *opt_dearepl* patches. The following steps are performed:

1. A command pipe is spawned down which ACIS commands will be written.
2. A telemetry pipe is spawned, terminating in the “*psci -m -u*” packet monitoring function with *expect* examining the standard output.
3. ACIS is cold-booted.
4. Software housekeeping, DEA replacement, and standard flight patches are applied.
5. ACIS is warm-booted.
6. FEP_0 is powered up.
7. A bias map containing the same value in each pixel of a given quadrant is written to the image loader.
8. A *te_3x3* parameter block is sent to ACIS. It calls for FEP_0 to be run in faint mode, calling for 3.3 second full-frame exposures.
9. A science run is started. Its telemetry is monitored by the *expect* script.
10. Once a *SWSTAT_FEP_STARTBIAS* user pseudopacket is received, three commands are sent to ACIS at 2-second intervals: two *stopScience* commands, followed by a command to power down FEP_0.
11. The script waits until one of three events occurs: (1) a *bepStartupMessage* packet is received, indicating that the BEP has crashed; (2) a *scienceReport* packet is received, indicating that the run ended normally without a crash; (3) neither packet has been received after 6 minutes.
12. The test is passed if case (1) occurs; otherwise, the test fails.

4.2. Fix Test

This test, controlled by the *expect* procedure “*fix-hw/runtestcc.tcl*”. is identical to the Reproduce Test except in two respects:

1. In step 4, *buscrash.bcnd* is added to the standard patch load.
2. In step 12, the test passes if case (2) occurs; otherwise it fails.

TITLE: ACIS Flight Software Standard Patch Component Release Notes

DOCUMENT NUMBER: 36-58010 REVISION: C

ORIGINATOR: Peter G. Ford <pgf@space.mit.edu>

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
01	36-984	Initial numeric release	jimf	10/27/1998
A	36-1006	Bug fixes, incorporate tests	RFG	05/11/1999
B	36-1019	Add new patches, retest	RFG	12/16/1999
C	36-1035	Add new patches, retest	RFG	08/09/2007
C	36-1035	Add tlmbust and buscrash		

=====
Title: ACIS Patch Release Notes for Version C

Software Change Order: 36-1035

Build Date: Tue Aug 14 17:47:07 EDT 2007
Part Number: 36-58010
Version: C
CVS Tag: release-C

IPCL Number: 36-53204.0204
IPCL Version: N
IPCL CVS Tag: release-N

Load Size: 2188 bytes

Description:

This is the second letter release of the standard patch set for the ACIS Flight Software.

The purpose of this release is to add the reportgradel optional patch.

This patch release uses new tools which combine FEP "C" code patches into a single FEP image, which is then patched into spare space in the BEP's FEP load image. It also includes a mechanism to run "release-level" tests of the patches, in addition to the regression tests provided by the individual patches.

This release consists of the following bug fix/system modification patches, where * indicates the new or modified patches since the previous release:

biastiming	- Fixes SPR 117
corruptblock	- Fixes SPR 113
digestbiaserror	- Fixes SPR 116
histogramvar	- Fixes SPR 115
rquad	- Fixes SPR 121
histogrammean	- Fixes SPR 123
zaplexpo	- Addresses SPR 122
condock	- Addresses SPR 127
fepbiasparity2	- Addresses SPR 130
cornermean	- Fixes SPR 128
* tlmbusy	- Fixes SPR 138
* buscrash	- Fixes SPR 140

For archival purposes, this document contains two attachments. The first contains ASCII command inputs to the ACIS command generator, "bcmd", used to generate the binary patch commands corresponding to this release. The second attachment contains the linker map listing for the ACIS Flight Software, and the patches built by this release.

The following documentation identifies these patches, provides a brief justification for each patch, and briefly describes the contents of these patches and their command, telemetry and science impacts.

Addressed Problem Reports:

SPR-128
SPR-123
SPR-127

SPR-130
SPR-138
SPR-122
SPR-115
SPR-113
SPR-140
SPR-117
SPR-116
SPR-121

Included Patches:

tlmbusy
fepbiasparity2
biastiming
histogramvar
zaplexpo
digestbiaserror
corruptblock
cornermean
buscrash
rquad
condock
histogrammean

Additional Release Level Tests:

=====
Patch Name: tlmbusy

Part Number: 36-58030.29

Version: A

SCO:

Description:

This standard patch prevents the BEP from writing anomalous telemetry output when the TlmManager::post() method is called from one task while it is still enqueueing a packet from another task.

The BEP will not drop the occasional packet (usually a housekeeping packet), and will be prevented from writing garbage in its stead. This will prevent the ground system from mis-processing science runs in which the garbage consists of correctly formatted, but unexpected, packets.

Applicable Reports/Requests:

SPR-138
SER-None

Test Results:

smoke --> PASS

Replaced Functions:

TlmManager::post

Command Impact:

None.

Telemetry Impact:

The occasional packet drop-out or garbling will no longer occur, so the impact should be wholly favorable.

Science Impact:

None.

=====
Patch Name: fepbiasparity2

Part Number: 36-58030.19
Version: A
SCO: 36-1015

Description:

In TE mode, this patch causes FEP_0 to bypass the upper half of each image map (rows 512 through 1023) if the bias parity errors in any one frame reported by the firmware exceed a threshold value (10). In addition, the 10 bias values, and their corresponding pixel values, are copied to a static location from which they can be dumped at a later time. In CC mode, the patch copies the lower half of the FEP_0 bias map into the upper half whenever 10 or more bias errors have been detected.

The patch has no effect on other FEPs.

Applicable Reports/Requests:
SPR-130

Test Results:

bugTe --> PASS
bugCc --> PASS
fixTe --> PASS
patchCc --> PASS

Replaced Functions:

Command Impact:

Once the patch is installed and FEP_0 powered up and running, it is advisable to clear its static save area via the following command:

```
write 'c' fep 0 0x80000210 {  
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
}
```

Then, either on a regular basis, or when it is noticed that 10 parity errors have been reported from a single FEP_0 exposure frame, the following command should be executed to dump the contents of the static save area:

```
read 'c' fep 0 0x80000210 20
```

Telemetry Impact:

If 10 or more bias parity errors are detected in FEP_0 during a timed-exposure science run, fepbiasparity2 will prevent more from being reported in telemetry. Once the threshold is reached, no further events will be reported from rows 512-1023. In 5x5 mode, a few additional parity errors may be reported from row 512.

In continuous clocking mode, when 10 or more bias parity errors are detected in FEP_0, fepbiasparity2 will copy the entire contents of the lower half of the bias map, i.e., 512 rows x 1024 pixels, to the upper half, thereby (hopefully) restoring the original contents. Occasional

parity errors will be corrected in the usual manner, i.e., by searching through the bias map, starting at row 0, for a pair of undamaged values.

Science Impact:

When this patch is triggered in timed-exposure modes, no further parity errors will be reported from rows 513-1023 of the CCD attached to FEP_0. In 3x3 mode, no events will be reported from rows 511-1023; in 5x5 mode, none will be reported from 510-1023. Ground software must be prepared to sense this condition, e.g., by examining the biasParityErrors fields in exposure packets, or by recognizing the absence of events above row 512, and updating the exposure maps accordingly.

The patch should have less impact in continuous clocking mode. When the 10-error threshold is triggered, FEP_0 may skip an exposure frame while replacing the upper half of its bias map, but otherwise, event processing will continue, taking advantage of the full area of the CCD.

=====
Patch Name: biastiming

Part Number: 36-58030.04
Version: A
SCO: 36-993

Description:

Reason:

This patch fixes a software problem which was first encountered during AXAF thermal vacuum testing at TRW.

Symptom:

At TRW thermal vacuum testing, someone observed that the instrument sent a science report in the middle of trickled bias map data. Bev has subsequently observed one case where the instrument started sending science data while trickling the bias maps.

Symptom Impact:

This symptom opens the possibility that the FEP threshold plane will lock up during a science run if the event rate is high enough (on the order of 5K events/sec/CCD).

Symptom Cause:

When the science manager tells the bias thief to start, by calling biasReady(), it set the thief's busy flag prior to signaling the task to start. If the task monitor sneaks in, the bias thief's main loop, goTaskEntry() ends up re-clearing the busyFlag, but then later picks up the start event and starts trickling the bias map. Since the busyFlag is clear at this point, the science manager assumes that the bias has been sent, and proceeds on to the data processing portion of the run (or if it's a bias only run or the run has been told to stop, the terminate the run).

Fix Description:

This patch replaces the BiasThief::biasReady() function with one that re-orders the setting of the busyFlag. In the patched version, the busyFlag is set AFTER the notification to the thief to start sending the bias. If the task monitor sneaks in, the thief will clear the flag, but once we return to the biasReady() function, the flag will be correctly asserted.

Applicable Reports/Requests:

SPR-117

Test Results:

unit --> PASS
fix --> PASS

Replaced Functions:

BiasThief::biasReady

Command Impact:

None

Telemetry Impact:

When this patch is not installed, it is possible, but rare, for bias maps to be telemetered while data processing is running and telemetering event data and exposure records, and even for a science report to be issued while the bias maps continue to be telemetered.

Once the patch is installed, the instrument will reliably wait until all of the bias maps have been telemetered before proceeding with the data processing portion of the run.

Science Impact:

Without this patch, it is possible, but extremely unlikely, that the FEP hardware threshold plane may lockup. This results in unreasonably low energy events being reported in the same set of positions, where ever there was a threshold crossing at the point where the threshold hardware locked up. This occurrence has only been seen with high event rates, on the order of 3000-5000 per exposure.

With this patch, this situation will not occur.

=====
Patch Name: histogramvar

Part Number: 36-58030.03
Version: A
SCO: 36-999

Description:

This patch fixes a software problem, SPR-115.

Symptom:

The Raw Histogram Mode occasionally produces anomalously large values for the low word of the overclock variances.

Symptom Impact:

This slightly degrades the science analysis of histogram mode data by very occasionally providing bad variance values for the overlocks.

Symptom Cause:

The error is cause by an unsigned integer divide which should have been a signed integer divide. If the low order word ends up negative this produces an incorrectly high value for the variance.

Fix Description:

This inline patch modifies the FEP to use a signed divide instead of unsigned divide.

Applicable Reports/Requests:

SPR-115

Test Results:

reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:

None

Telemetry Impact:

None

Science Impact:

This patch affects Histogram Mode Only.
Without this patch, the overclock variances in histogram mode may occasionally be incorrect. Once this patch is installed, the Flight Software correctly computes overclock variances.

=====
Patch Name: zaplexpo

Part Number: 36-58030.16

Version: A

SCO: 36-997

Description:

Reason:

In event-finding mode, the FEP thresholds are adjusted using delta-overclock values, which are calculated from difference between the average overclock values from the preceding frame and the average overclock values from the initial bias frame. The delta-overclocks for the initial data frame are set to zero, i.e., it is assumed that the mean bias levels haven't drifted since the first exposure frame used to compute the bias map. This is often a poor assumption, and can lead to a very large number of events being reported within the first exposure.

Fix Description:

Inhibit the FEP from finding any threshold crossings within the first examined exposure frame. This is performed at science run initialization time within the "fepSciTimed.c":FEPsciTimedInit function (TE mode) and the "fepSciCclk.c":FEPsciCclkInit function (CC mode) by storing 4095 in the FEP threshold registers. Thus,

```

186:fepSciTimed.c ****   for (iquad = 0; iquad < 4; iquad++) {
925 0290 21200000           move    $4,$0
926 0294 0000053C           la      $5,stageThresh
926      0000A524
187:fepSciTimed.c ****           fp->ex.bias0[iquad] = fp->br.bias0[iquad];
929 029c 40100400           sll    $2,$4,1
930                               $L90:
931 02a0 21105000           addu   $2,$2,$16
932 02a4 A0024394           lhu    $3,672($2)
933 02a8 00000000
934 02ac 100043A4           sh     $3,16($2)
188:fepSciTimed.c ****           fp->ex.dOclk[iquad] = 0;
937 02b0 180040A4           sh     $0,24($2)
189:fepSciTimed.c ****           FIOsetThresholdRegister(iquad, (short)(fp->tp.thresh[iqu
ad]));
944 02b4 80180400           sll    $3,$4,2
945 02b8 21107000           addu   $2,$3,$16
948 02bc 21186500           addu   $3,$3,$5
949 02c0 4C004284           lh     $2,76($2)
950 02c4 00000000
951 02c8 000062AC           sw     $2,0($3)
958 02cc 01008424           addu   $4,$4,1
959 02d0 0400822C           sltu   $2,$4,4
960                               .set   noreorder
961                               .set   nomacro
962 02d4 F2FF4014           bne    $2,$0,$L90
963 02d8 40100400           sll    $2,$4,1
964                               .set   macro
965                               .set   reorder
190:fepSciTimed.c ****   }
```

becomes

```

186:fepSciTimed.c ****   for (iquad = 0; iquad < 4; iquad++) {
925 0290 21200000           move    $4,$0
926 0294 0000053C           la      $5,stageThresh
926      0000A524
```

```

187:fepSciTimed.c **** fp->ex.bias0[iquad] = fp->br.bias0[iquad];
929 029c 40100400      sll      $2,$4,1
930                    $L90:
931 02a0 21105000      addu     $2,$2,$16
932 02a4 A0024394      lhu     $3,672($2)
933 02a8 00000000
934 02ac 100043A4      sh      $3,16($2)
188:fepSciTimed.c **** fp->ex.dOclk[iquad] = 0xffff;
937 02b0 FF0F0324      li      $3,0x00000fff
944 02b4 180043A4      sh      $3,24($2)
189:fepSciTimed.c **** FIOsetThresholdRegister(iquad, 0xffff);
945 02b8 80180400      sll     $3,$4,2
948 02bc 21186500      addu    $3,$3,$5
949 02c0 FF0F0224      li      $2,0x00000fff
950 02c4 00000000
951 02c8 000062AC      sw      $2,0($3)
958 02cc 01008424      addu    $4,$4,1
959 02d0 0400822C      sltu    $2,$4,4
960                    .set    noreorder
961                    .set    nomacro
962 02d4 F2FF4014      bne     $2,$0,$L90
963 02d8 40100400      sll     $2,$4,1
964                    .set    macro
965                    .set    reorder
190:fepSciTimed.c **** }

```

and

```

174:fepSciCCLK.c **** for (iquad = 0; iquad < 4; iquad++) {
774 01fc 21200000      move    $4,$0
775 0200 0000053C      la      $5,stageThresh
775      0000A524
175:fepSciCCLK.c **** fp->ex.bias0[iquad] = fp->br.bias0[iquad];
778 0208 40100400      sll     $2,$4,1
779                    $L83:
780 020c 21105000      addu    $2,$2,$16
781 0210 A0024394      lhu     $3,672($2)
782 0214 00000000
783 0218 100043A4      sh      $3,16($2)
176:fepSciCCLK.c **** fp->ex.dOclk[iquad] = 0;
786 021c 180040A4      sh      $0,24($2)
177:fepSciCCLK.c **** FIOsetThresholdRegister(iquad, (short)(fp->tp.thresh[iqu
ad]));
793 0220 80180400      sll     $3,$4,2
794 0224 21107000      addu    $2,$3,$16
797 0228 21186500      addu    $3,$3,$5
798 022c 4C004284      lh      $2,76($2)
799 0230 00000000
800 0234 000062AC      sw      $2,0($3)
807 0238 01008424      addu    $4,$4,1
808 023c 0400822C      sltu    $2,$4,4
809                    .set    noreorder
810                    .set    nomacro
811 0240 F2FF4014      bne     $2,$0,$L83
812 0244 40100400      sll     $2,$4,1
813                    .set    macro
814                    .set    reorder
178:fepSciCCLK.c **** }

```

becomes

```

174:fepSciCCLK.c **** for (iquad = 0; iquad < 4; iquad++) {
774 01fc 21200000      move    $4,$0
775 0200 0000053C      la      $5,stageThresh

```

```
775          0000A524
175:fepSciCclk.c ****      fp->ex.bias0[iquad] = fp->br.bias0[iquad];
778 0208 40100400          sll      $2,$4,1
779                                $L83:
780 020c 21105000          addu    $2,$2,$16
781 0210 A0024394          lhu     $3,672($2)
782 0214 00000000
783 0218 100043A4          sh      $3,16($2)
176:fepSciCclk.c ****      fp->ex.dOclk[iquad] = 0xffff;
786 021c FF0F0324          li      $3,0x00000fff
787 0220 180043A4          sh      $3,24($2)
177:fepSciCclk.c ****      FIOsetThresholdRegister(iquad, 0xffff);
793 0224 80180400          sll     $3,$4,2
797 0228 21186500          addu    $3,$3,$5
798 022c FF0F0224          li      $2,0x00000fff
799 0230 00000000
800 0234 000062AC          sw      $2,0($3)
807 0238 01008424          addu    $4,$4,1
808 023c 0400822C          sltu    $2,$4,4
809                                .set    noreorder
810                                .set    nomacro
811 0240 F2FF4014          bne     $2,$0,$L83
812 0244 40100400          sll     $2,$4,1
813                                .set    macro
814                                .set    reorder
178:fepSciCclk.c ****      }
```

Applicable Reports/Requests:
SPR-122

Test Results:
reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:
None

Telemetry Impact:
No events will be generated for the first examined exposure, i.e., the frame with exposureNumber == 2 (unless the teignore or ccignore patches are loaded, in which case it will be the frame with exposureNumber == ignoreInitialFrames).

To determine whether this patch was in effect during a particular science run, telemetry processing software should examine the 4 values in the deltaOverclocks array in exposure packets with exposureNumber == 2 (or with exposureNumber == ignoreInitialFrames if the relevant teignore or ccignore patch is installed). If they are all equal to 4095, the patch was installed and this exposure frame should not be included in the good time interval (GTI); if they are all zero, the patch was omitted.

Science Impact:
With this patch installed, the frame with exposureNumber == 2 (or with exposureNumber == ignoreInitialFrames if the relevant teignore or

ccignore patch is installed) should not be included in the GTI maps.

=====
Patch Name: digestbiaserror

Part Number: 36-58030.02
Version: A
SCO: 36-995

Description:

This patch fixes software problem SPR-116.

Symptom:

When a parity error is detected, the FEP produces a pair of bias values with a flag indicating if one or both are corrupt. The BEP mishandles this when telemetering the error. If the error occurs at an odd column position, the BEP reports the wrong column position of the error.

Symptom Impact:

This has the potential to degrade the science analysis by providing ambiguous knowledge of which bias map values have been corrupted.

Symptom Cause:

In PmEvent::digestBiasError, it assumes that only one of pair of bias values is corrupt and that the FEP reported column indicates which of the two is corrupt. This is WRONG.

Fix Description:

This inline patch provides a new representation of the bias error event and modifies the telemetry format tag to indicate the new format. Rather than telemeter the corrupt value (which is fairly useless), the 12-bit value field is as follows, where bit 0 is the least-significant bit:

- Bits 0 - 3: The top 4 bits of the bias value at the column position
- Bits 4 - 7: The top 4 bits of the bias value at column + 1
- Bits 8 - 11: Unused

These bits contain the results of the hardware parity check of the corresponding pixel bias value.

The format of these 4 bits are as follows:

- Bit 0 (H/W bit 12) - Always zero
- Bit 1 (H/W bit 13) - H/W computed parity of bias map value
- Bit 2 (H/W bit 14) - Parity bit stored in parity plane
- Bit 3 (H/W bit 15) - Parity error bit (0 - no parity error, 1 - parity error)

The bit definition information is derived from the "DPA Hardware Specification and System Description", MIT 36-02104 Rev. C., Section 2.2.2.5.5 "Bias Map Parity Detection".

Applicable Reports/Requests:

SPR-116

Test Results:

reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:
None

Telemetry Impact:

This patch affects the telemetry Pixel Bias Map Error records. Without this patch, the error records will be incorrect if the error occurs on an odd column. With this patch installed, the instrument will telemetry bias errors using a new telemetry format, TTAG_SCI_PATCHED_BIAS_ERROR, defined by the "Patch Data Bias Error" format in the IP&CL Software Structures Definitions, MIT 36-53204.0204 Rev. L.

Science Impact:

Without the patch installed, there is an ambiguity whether a bias error is in the reported pixel, or in the adjacent, odd column. Once the patch is installed, the ground can determine exactly which pixel was upset.

=====
Patch Name: corruptblock

Part Number: 36-58030.01
Version: A
SCO: 36-994

Description:

Reason:
This patch fixes software problem report SPR-113.

Symptom:
If a parameter block is corrupt, the flight software may use nonsense parameters, if just powered on, or run the previous run mode's parameter block.

Symptom Impact:
If the original parameter block was corrupt and if this was the first run since the instrument was powered, the nonsense parameters may cause the instrument to crash and reset, preventing any science activity during that observation's time period. The system will recover, although without patches, at the onset of the next observation. If there was an earlier run of the same type, Timed Exposure or Continuous Clocking, the previous run's parameter will be used, which may or may not be ideal.

Symptom Cause:
The flight software start run routine, ChStartSciRun::processCmd(), declares an "alternate" parameter block variable, which is filled in by the science mode's checkBlock() routine if the original parameter block is corrupt. processCmd() then erroneously passes this "alternate", and a reference to the "alternate" back to checkBlock() to verify that the alternate is not also corrupt. The called checkBlock() initializes the 2nd reference to INVALID, which ends up overwriting the desired alternate block id. This propagates through to the run, preventing the mode from loading the parameter block, and using, instead, what it had already staged from an earlier run.

Fix Description:
This inline patch modifies 2nd parameter to refer to a dummy variable when checking the default backup block. This prevents the id from being overridden and provides the proper default parameter block selection behavior when the selected block has been corrupted.

```
The original line from chstartscirun.C is:
    if (mode.checkBlock (blockid, alternate) == BoolTrue)
    {
        result = CMDRESULT_OK;
    }
<<< else if (mode.checkBlock (alternate, alternate) == BoolTrue)
    {
        blockid = alternate;
        usedAlternate = BoolTrue;
    }
    else
    {
        return CMDRESULT_CORRUPT_IDLE;
    }
```

The effect of the patch changes this to:

```
if (mode.checkBlock (blockid, alternate) == BoolTrue)
{
    result = CMDRESULT_OK;
}
>>> else if (mode.checkBlock (alternate, dummy) == BoolTrue)
{
    blockid = alternate;
    usedAlternate = BoolTrue;
}
else
{
    return CMDRESULT_CORRUPT_IDLE;
}
```

The stack frame of the modified patch will appear as follows, where the offsets in the left-hand column are relative to the stack pointer at the time the jump is made to the called subroutine mode.checkBlock(), the symbols in the center column indicate the "conventional" locations for various registers, and the right column indicates if the assembler actually put anything into that stack slot. If "unassigned" then the assembler didn't explicitly store anything into that stack slot. If blank, then the "convention"
(NOTE: In the MIPS processors, calls don't explicitly push anything on the stack. The return address is maintained in "ra" at the time of the call and the caller is then required to save it if needed):

```
*
* ChStartSciRun::processCmd() - Stack Frame
* Convention described in Section 2.3 of
* MIPS programmers handbook, by Farquahar and Bunce
*
* 60  pad      unassigned
* 56  ra      ra ($31)
* 52  s3      s3 ($19)
* 48  s2      s2 ($18)
* 44  s1      s1 ($17)
* 40  s0      s0 ($16)
* 36  f23     unassigned      (patch uses as local "dummy")
* 32  f22     alternate      (local variable)
* 28  f21     unassigned
* 24  f20     unassigned
* 20  pad     unassigned
* 16  arg     biasonly argument (arg4) to scienceManager.startRun()
* 12  a3      unassigned
* 8   a2      unassigned
* 4   a1      unassigned
* 0   a0      unassigned
```

Applicable Reports/Requests:
SPR-113

Test Results:
reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:
Without this patch, corruptions (if any are actually ever encountered) may cause an previous parameter block to be used for an observation, or

at worst, a reset of the instrument.

When the patch is installed, the instrument will use the appropriate default parameter block (slot 0 or slot 1) instead of the corrupted parameter block, or will skip the observation if the defaults are also corrupt.

Telemetry Impact:

None.

Although, without this patch, the instrument may select an inappropriate parameter block, the parameter blocks dumped to telemetry at the start of a science run will always be the the ones actually used for the run.

Science Impact:

None

=====
Patch Name: cornermean

Part Number: 36-58030.21
Version: A
SCO: 36-1017

Description:

Reason:
This patch fixes software problem report SPR-128.

Symptom:
In Timed Exposure Graded Telemetry mode, when some of the corner pixels have a small negative corrected pulse height, the system reports an incorrect, extremely large negative value for the mean corrected pulse height of the corner pixels. Additionally, the algorithm rounds incorrectly when the mean pulse height is negative (not mentioned in the SPR).

Symptom Impact:
Barring corrective ground analysis and action, the incorrectly reported corner mean value may confuse the science analysis process, and at worst, lead to incorrect conclusions about the science, or the state of the instrument data processing.

Symptom Cause:
The flight software routine, Pixel3x3:computePhGrade() divides a signed integer value, cornersum, with an unsigned integer value, sumcount (see filesscience/pixel3x3.H). In "C" and "C++", this division is performed as an unsigned divide, preventing any sign extension, hence the "signedness" of the cornersum is lost. The result is stored into a signed value, cornermean, which is later converted to a signed 13-bit value for telemetry. When the ground software extracts the 13-bit signed value, it will sign-extend the value. The effect of losing the sign in the divide, sometimes yields incorrect results, some of which appear as large negative values when processed by the ground.

The rounding problem is due to incorrect coding of the integer rounding for negative values:

```
mean = (sum + (count/2))/count  
should be:  
mean = (sum + (sign(sum) * int(count)/2))/int(count)
```

Fix Description:
This patch implements the fix to the loss of "signedness" problem and the rounding using an inline assembler patch.

To fix the loss of "signedness" problem the patch replaces the existing unsigned divide instruction (divu) with a signed divide (div).

In order to fix the rounding problem, more work was needed.

The coded formula is:
mean = (sum + (count/2))/count

In practice, the MIPS assembler implements divides as an embedded assembler macro which performs a divide by zero check. In the case of Pixel3x3 it is as follows:

```
0370 2000638E   lw      $3,32($19)
0374 00000000
0378 42100300   srl     $2,$3,1
037c 2400648E   lw      $4,36($19)
0380 00000000

---- Code we're going to muck with ----
0384 21104400   addu   $2,$2,$4
0388 1B004300   divu   $2,$2,$3
      02006014
      00000000
      0D000700
---- End of code we're going to muck with ----
0398 12100000
039c 00000000
      00000000
03a4 280062AE   sw     $2,40($19)

...

```

Since the C++ code already has an earlier zero check on the denominator, the patch re-codes this portion function as follows:

```
0370 2000638E   lw      $3,32($19)
0374 00000000
0378 42100300   srl     $2,$3,1
037c 2400648E   lw      $4,36($19)
0380 00000000

---- Start of change ----
0384           bgez   $4,positive
0388           add    $2,$2,$4
038c           sub    $2,$2,$3
positive:
0390           div    $0,$2,$3
0394           nop
---- End of change ----

0398 12100000
039c 00000000
      00000000
03a4 280062AE   sw     $2,40($19)

```

Applicable Reports/Requests:
SPR-128

Test Results:
reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:
None.

Telemetry Impact:

None.

Science Impact:

Without this patch, the corner mean values in Graded Telemetry mode may occasionally be invalid. There is a deterministic ground algorithm which can detect and and correct for this effect, but without the flight patch or the ground algorithm, the corner mean values may be grossly incorrect in some cases.

Once the patch is in place, the corner mean values should be within 1/2 an ADU of the true mean, regardless if sign, without further action needed by the ground science software.

=====
Patch Name: buscrash

Part Number: 36-58030.30

Version: A

SCO:

Description:

Reason:

If ACIS is computing bias maps when commanded to power down its front-end processors (FEPs), it is likely to crash the back-end processor (BEP) interface bus, causing the BEP to reboot without flight software patches. Normal operations must be restored via ground command. The cause of the problem has been traced to a design flaw in the BEP flight software and this ECO describes a small patch that will fix it.

Symptom:

During execution of SCS107, typically due to high background radiation, ACIS is powered down. Science telemetry reports that the flight s/w version number is 11, whereas typical values (depending in the patch combination) are 30 or higher, indicating that the BEP rebooted itself. Subsequent inspection of the recorded telemetry shows no scienceReport packet from the last science run, but a bepStartupMessage packet with lastFatalCode=7 and watchdogFlag=1.

Symptom Impact:

Since the observatory is usually in safe mode for several hours following the SCS107, there is generally sufficient time to establish a realtime contact, set the BEP's warm-boot flag, and restart it. However, this takes time and manpower.

Symptom Cause:

The bus crash has been traced to a flaw in the FepManager::loadBadPixel() method. This routine is executed after the FEP bias maps have been created and before they are (optionally) reported in telemetry. It uses the memory-mapped interface between BEP and FEP to change those locations in the FEP bias maps that correspond to "bad" pixels or whole columns. However, unlike all other FepManager operations, loadBadPixel() does not confirm that a FEP is powered up before it writes to its map. This causes the bus crash.

Fix Description:

Call the FepManager::isEnabled() method to check if the FEP is powered up before writing to a FEP's bias memory (and parity plane).

Applicable Reports/Requests:

SPR-140

Test Results:

reproduce --> PASS

fix --> PASS

Replaced Functions:

FepManager::loadBadPixel

Command Impact:

None.

Telemetry Impact:

None.

Science Impact:

None.

=====
Patch Name: rquad

Part Number: 36-58030.14

Version: A

SCO: 36-1000

Description:

Reason:

This patch fixes software problem report SPR-121.

Symptom:

If the center pixel of a 3x3 event is in the last column of any but the right-most quadrant (i.e. in FULL mode, quadrants A, B or C, but not D), the flight software will inappropriately use the delta overclock and split threshold for the center pixel's quadrant on the pixels on the right edge of the event. The instrument is supposed to use the delta overclock and split thresholds for the next quadrant on these pixels.

Symptom Impact:

This may lead to an incorrect estimate of the event's total pulse height and grade, possibly leading to inappropriate pulse height and grade filtering of these events, or, when using Graded Event formats, incorrect pulse height and grade code values.

Symptom Cause:

The flight software is fetching the quadrant identifier for the wrong column position for the right edge pixels:

```
quad = exposure->getQuadrant (col);  
doclk[1] = exposure->getOverclockDelta (quad);  
split[1] = exposure->getSplitThreshold (quad);
```

```
WRONG---> quad = exposure->getQuadrant (col);  
doclk[2] = exposure->getOverclockDelta (quad);  
split[2] = exposure->getSplitThreshold (quad);
```

```
computePhGrade (doclk, split);
```

This should be:

```
quad = exposure->getQuadrant (col);  
doclk[1] = exposure->getOverclockDelta (quad);  
split[1] = exposure->getSplitThreshold (quad);
```

```
CORRECT---> quad = exposure->getQuadrant (col+1);  
doclk[2] = exposure->getOverclockDelta (quad);  
split[2] = exposure->getSplitThreshold (quad);
```

```
computePhGrade (doclk, split);
```

Fix Description:

The patch increments the column register variable using an "nop" slot of an earlier instruction following the previous call to exposure->getQuadrant() and prior to the last call to exposure->getQuadrant().

This is the last time the register is used in the function, so it won't corrupt subsequent code, and the "nop" was inserted by the compiler after a "lw", which allows for increments of registers unrelated to the "lw".

```

                                05cc 2C00A2AF          sw      $2,44($sp)
                                $LM84:
                                210:../filesscience/pixel3x3.C ****
                                211:../filesscience/pixel3x3.C ****      quad = exposure->getQ
uadrant (col);
                                05d0 5400028E          lw      $2,84($16)
"addu $18,$18,1" --->> 05d4 00000000
                                05d8 0800428C          lw      $2,8($2)
                                00000000
                                05e0 21200002          move    $4,$16
                                .set    noreorder
                                .set    nomacro
"col" is passed in      05e4 09F84000          jal    $31,$2
a delay slot          --->>05e8 21284002          move    $5,$17
                                .set    macro
                                .set    reorder

                                05ec 21884000          move    $17,$2
                                $LM85:
                                ../filesscience/pixel3x3.C ****      doclk[2] = exposure->getO
verclockDelta (quad);
                                05f0 5400028E          lw      $2,84($16)
                                05f4 00000000
                                05f8 0400428C          lw      $2,4($2)
                                00000000
                                0600 21200002          move    $4,$16
                                .set    noreorder
                                .set    nomacro
                                0604 09F84000          jal    $31,$2
                                0608 21282002          move    $5,$17
                                .set    macro
                                .set    reorder

                                060c 2000A2AF          sw      $2,32($sp)
                                $LM86:
                                ../filesscience/pixel3x3.C ****      split[2] = exposure->getS
plitThreshold (quad);
                                .stabn 68,0,213,$LM86
                                0610 5400028E          lw      $2,84($16)
                                0614 00000000
                                0618 0C00428C          lw      $2,12($2)
                                00000000
                                0620 21200002          move    $4,$16
                                .set    noreorder
                                .set    nomacro
                                0624 09F84000          jal    $31,$2
                                0628 21282002          move    $5,$17
                                .set    macro
                                .set    reorder

                                062c 3000A2AF          sw      $2,48($sp)
                                $LM87:
                                ../filesscience/pixel3x3.C ****
                                ../filesscience/pixel3x3.C ****      computePhGrade (doclk, sp
lit);
                                .stabn 68,0,215,$LM87
                                0630 1000828E          lw      $2,16($20)
                                0634 00000000
                                0638 1C00428C          lw      $2,28($2)

```



```
00000000
0640 21208002          move    $4,$20
0644 1800A527          addu   $5,$sp,24
                                .set   noreorder
                                .set   nomacro
0648 09F84000          jal   $31,$2
064c 2800A627          addu   $6,$sp,40
                                .set   macro
                                .set   reorder

                                $LBB29:
                                $LM88:
                                $LBB30:
                                $LBE30:
                                $LM89:
                                $LBE29:
                                $LM90:
../filesscience/pixel3x3.C ****
../filesscience/pixel3x3.C **** //
../filesscience/pixel3x3.C **** }
                                $LBE26:
0650 4C00BF8F          lw     $31,76($sp)
                                00000000
0658 4800B48F          lw     $20,72($sp)
                                00000000
0660 4400B38F          lw     $19,68($sp)
                                00000000
0668 4000B28F          lw     $18,64($sp)
                                00000000
0670 3C00B18F          lw     $17,60($sp)
                                00000000
0678 3800B08F          lw     $16,56($sp)
                                00000000
0680 5000BD27          addu   $sp,$sp,80
0684 0800E003          j      $31
                                00000000

                                .end   Pixel3x3::attachData(FEEven
tRec3x3 const *, EventExposure *)

                                $LM91:
```

Applicable Reports/Requests:
SPR-121

Test Results:
reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:
None

Telemetry Impact:
See SCIENCE IMPACT.

Science Impact:
Without this patch, all Timed Exposure and CC3x3 events on the left edge of a quadrant boundary may have incorrect pulse heights and

grades, and events which impact at these positions may be inappropriately filter out or telemetered if pulse height and grade filters are used.

=====
Patch Name: condock

Part Number: 36-58030.17

Version: A

SCO: 36-1012

Description:

Reason:

The first timed exposure frames received during OAC (e.g., SOP_61052_DARK_CUR) showed sporadic increases in the overclock averages, and anomalous dark patches within bias maps. Once raw frames were examined (in SOP_61054_RAW_DATA and SAP_61079_RAW_BIAS), the effect was seen to be caused by charged particle background "leaking" into the overclocks.

Fix Description:

Patch the FEP overclock processing function, fepOclkProc in fep/fepCtl.c, to "condition" the overclock sum on a row-by-row basis. The patch, which will not apply to OC_RAW or OC_HIST modes, will ignore the overclock sum of particular row and node if it exceeds the previous sum by some suitable threshold. This entails replacing the following fepOclkProc() code:

```
for (ioclck = 0; ioclck < fp->tp.noclk; ioclck++) {
    unsigned p0 = *fp->oc.optr++;
    unsigned p1 = *fp->oc.optr++;
    switch (fp->tp.quadcode) {
    case FEP_QUAD_AC:
        fp->oc.osum[0] += PIXEL0(p0) & PIXEL_MASK;
        fp->oc.osum[1] += PIXEL0(p1) & PIXEL_MASK;
        break;
    case FEP_QUAD_BD:
        fp->oc.osum[0] += PIXEL1(p0) & PIXEL_MASK;
        fp->oc.osum[1] += PIXEL1(p1) & PIXEL_MASK;
        break;
    default:
        fp->oc.osum[0] += PIXEL0(p0) & PIXEL_MASK;
        fp->oc.osum[1] += PIXEL1(p0) & PIXEL_MASK;
        fp->oc.osum[2] += PIXEL0(p1) & PIXEL_MASK;
        fp->oc.osum[3] += PIXEL1(p1) & PIXEL_MASK;
        break;
    } /* end switch */
} /* end for ioclck */
```

with an inline patch that saves R9-R12:

```
condockCtl(fp);

    subu    $sp,$sp,16
    sw     $9,0($sp)
    sw     $10,4($sp)
    sw     $11,8($sp)
    sw     $12,12($sp)
    jal    condockCtl
    move   $4,$16
    lw     $9,0($sp)
    lw     $10,4($sp)
    lw     $11,8($sp)
    lw     $12,12($sp)
    j      fepCtl+0x0f74
    addu   $sp,$sp,16
```

and adding the condockCtl function:

```
void condockCtl(FEPparm *fp)
{
    unsigned dsum = OCLK_COND * fp->tp.noclk;
    unsigned ioclk, iquad;

    /* clear local accumulator */
    for (iquad = 0; iquad < 4; iquad++) {
        fp->oc.ossql[iquad] = 0;
        /* clear saved row sum at start of frame */
        if (fp->oc.osum[iquad] == 0) {
            fp->oc.ossqh[iquad] = 0;
        }
    } /* end for iquad */

    /* accumulate the overclock sums */
    for (ioclk = 0; ioclk < fp->tp.noclk; ioclk++) {
        unsigned p0 = *fp->oc.optr++;
        unsigned p1 = *fp->oc.optr++;
        switch (fp->tp.quadcode) {
            case FEP_QUAD_AC:
                fp->oc.ossql[0] += PIXEL0(p0) & PIXEL_MASK;
                fp->oc.ossql[1] += PIXEL0(p1) & PIXEL_MASK;
                break;
            case FEP_QUAD_BD:
                fp->oc.ossql[0] += PIXEL1(p0) & PIXEL_MASK;
                fp->oc.ossql[1] += PIXEL1(p1) & PIXEL_MASK;
                break;
            default:
                fp->oc.ossql[0] += PIXEL0(p0) & PIXEL_MASK;
                fp->oc.ossql[1] += PIXEL1(p0) & PIXEL_MASK;
                fp->oc.ossql[2] += PIXEL0(p1) & PIXEL_MASK;
                fp->oc.ossql[3] += PIXEL1(p1) & PIXEL_MASK;
                break;
        } /* end switch */
    } /* end for ioclk */

    /* condition the sums */
    for (iquad = 0; iquad < 4; iquad++) {
        if (fp->oc.ossqh[iquad] == 0) {
            /* always save first row sum */
            fp->oc.ossqh[iquad] = fp->oc.ossql[iquad];
        } else if (fp->oc.osum[iquad] == fp->oc.ossqh[iquad] &&
            fp->oc.ossqh[iquad] > fp->oc.ossql[iquad] + dsum) {
            /* if second row sum much less than first, replace the
            total sum by twice the second sum */
            fp->oc.osum[iquad] = fp->oc.ossqh[iquad] = fp->oc.ossql[iquad];
        } else if (fp->oc.ossql[iquad] <= fp->oc.ossqh[iquad] + dsum) {
            /* save row sum if not much greater than the saved sum */
            fp->oc.ossqh[iquad] = fp->oc.ossql[iquad];
        }
        /* increment overclock accumulator */
        fp->oc.osum[iquad] += fp->oc.ossqh[iquad];
    } /* end for iquad */
}
```

The algorithm uses the oc.ossql[4] and oc.ossqh[4] fields which would not otherwise participate in OC_SUM mode, and whose prior contents may be safely overwritten. The oc.ossql fields are used to accumulate the overlocks of the current row, and the current "best" value of this

sum is saved from row to row in oc.ossqh. If the current row sum exceeds the current best sum by a constant OCLK_COND times the number of overclocks in the row, the current best sum will be used in its place; otherwise, the sum of the current row will replace the current best. The first two rows of each frame receive special treatment: the first row sum is used to initialize oc.ossqh -- the "best" sum -- and, if the sum of the second row is anomalously LOWER than this, the best row sum and the running total sum are corrected.

Applicable Reports/Requests:
SPR-127

Test Results:
reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:
None

Telemetry Impact:
None

Science Impact:
With this patch installed, the effect of background events on overclock averages will be greatly reduced, directly reducing systematic errors within bias maps and increasing the accuracy of photon energy determination.

=====
Patch Name: histogrammean

Part Number: 36-58030.15
Version: A
SCO: 36-996

Description:

Reason:

In raw TE histogram mode, the FEPs report the mean of each CCD quadrant's overclocks. This is done in two steps: first, the overclocks of each quadrant of each frame are summed into fields "oc.osum" in the FEpparm structure, and these are then averaged over the separate "histogramCount" frames and reported to the BEP in "omean" fields in FEPEventRecHist structures. The error is caused by using the 16-bit "omean" fields as accumulators, as well as final values, since, if the mean overclock value multiplied by "histogramCount" exceeds 65535, overflow will occur.

Fix Description:

The patch adds 8 32-bit integer fields to the end of the D-cache stack employed by the fepCtl function. Within FEPsciTimedHist, machine instructions are altered to initialize these fields to zero, to use them to accumulate the intermediate sums, and hence to form the means which are stored into "omean".

(a) increase fepCtl stack length by an extra 32 bytes

```

                .globl fepCtl_lst_0000_0000
                .ent   fepCtl_lst_0000_0000
fepCtl_lst_0000_0000:
0000 88FABD27          subu    $sp,$sp,1368+32
0004 5405BFAF
                .end   fepCtl_lst_0000_0000

```

(b) decrease fepCtl stack length by an extra 32 bytes

```

                .globl fepCtl_lst_012c_012c
                .ent   fepCtl_lst_012c_012c
fepCtl_lst_012c_012c:
0128 00000000
012c 7805BD27          addu   $sp,$sp,1368+32
0130 0800E003
                .end   fepCtl_lst_012c_012c

```

(c) set mean and variance sums to zero

```

                .globl fepSciTimed_lst_1858_1864
                .ent   fepSciTimed_lst_1858_1864
fepSciTimed_lst_1858_1864:
1854 80180B00          addu   $3,$3,$16
1858 21187000          sw     $0,1368-16($3)
185c 480560AC          sw     $0,1368($3)
1860 580560AC          sh     $0,20($2)
1864 140040A4
1868 0C0044A4
                .end   fepSciTimed_lst_1858_1864

```

(d) increment mean sum

```
                .globl  fepSciTimed_lst_lacc_ladc
                .ent    fepSciTimed_lst_lacc_ladc
fepSciTimed_lst_lacc_ladc:
lab0 1B006A00
      02004015
      00000000
      0D000700
      12180000
lacc 34050925      addu   $9,$8,1368-36
lad0 4805028D      lw     $2,1368-16($8)
lad4 00000000      nop
lad8 21104300      addu   $2,$2,$3
ladc 480502AD      sw     $2,1368-16($8)
lae0 1B00AA01
lae4 02004015
lae8 00000000
laec 0D000700
laf0 12200000
                .end    fepSciTimed_lst_lacc_ladc
```

(e) save stack pointer in R9

```
                .globl  fepSciTimed_lst_lc38_lc38
                .ent    fepSciTimed_lst_lc38_lc38
fepSciTimed_lst_lc38_lc38:
lc34 1403028E
lc38 4805092E      addu   $9,$16,1368-16
lcec 22004010
                .end    fepSciTimed_lst_lc38_lc38
```

(f) load overclock mean sum

```
                .globl  fepSciTimed_lst_lc50_lc50
                .ent    fepSciTimed_lst_lc50_lc50
fepSciTimed_lst_lc50_lc50:
lc4c 21187200
lc50 0000228D      lw     $2,0($9)
lc54 00000000
                .end    fepSciTimed_lst_lc50_lc50
```

(g) load overclock variance sum

```
                .globl  fepSciTimed_lst_lc84_lc84
                .ent    fepSciTimed_lst_lc84_lc84
fepSciTimed_lst_lc84_lc84:
lc80 21187200
lc84 1000228D      lw     $2,16($9)
lc88 00000000
                .end    fepSciTimed_lst_lc84_lc84
```

(h) increment R9

```
                .globl  fepSciTimed_lst_lcb8_lcb8
                .ent    fepSciTimed_lst_lcb8_lcb8
fepSciTimed_lst_lcb8_lcb8:
lcb4 1403028E
lcb8 04002925      addu   $9,$9,4
lcbc 2B106201
                .end    fepSciTimed_lst_lcb8_lcb8
```

SPR-123

Test Results:

```
reproduce --> PASS
fix --> PASS
```

Replaced Functions:

Command Impact:

None

Telemetry Impact:

None. It should be pointed out that an alternative approach to fixing this problem is to add the following code to the downlink raw histogram software, although this algorithm may fail for very large values of "histogramCount".

```
if (fs->meanOverclock[node] < fs->minimumOverclock[node] ||
    fs->meanOverclock[node] > fs->maximumOverclock[node]) {
    unsigned hh = loadTeBlock_histogramCount(param);
    double dmlim = 8192.0*hh*loadTeBlock_overclockPairsPerNode(param);
    unsigned mmlim, mlim = (dmlim < 0x7fffffff) ? dmlim : 0x7fffffff;
    for (mm = 0; mm < mlim; mm += 65536) {
        unsigned nn = fs->meanOverclock[node]+(mm+hh/2)/hh;
        if (nn >= fs->minimumOverclock[node] &&
            nn <= fs->maximumOverclock[node]) {
            fs->meanOverclock[node] = nn;
            break;
        }
    }
}
```

Science Impact:

None -- raw histogram mode is not necessary for science processing.

TITLE: ACIS Flight Software Optional Patch Component Release Notes

DOCUMENT NUMBER: 36-58020 REVISION: C

ORIGINATOR: Peter G. Ford <pgf@space.mit.edu>

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
01	36-987	Initial numeric release	jimf	11/12/1998
A	36-1007	Bug fixes, incorporate tests	RFG	05/12/1999
B	36-1019	Add new patches, retest	RFG	12/16/1999
C	36-1022	Add new patches, retest	RFG	03/21/2003
C	36-1035	Recompile after change in standa		

=====
Title: ACIS Optional Patch Release Notes for Version C

Software Change Order: 36-1035

Build Date: Wed Aug 15 00:24:26 EDT 2007
Part Number: 36-58020
Version: C
CVS Tag: release-C-opt-C

Std Number: 36-58010
Std Version: C
Std Tag: release-C
Std SCO: 36-1035

IPCL Number: 36-53204.0204
IPCL Version: N
IPCL CVS Tag: release-N

Description:

This is the fourth letter release of the optional patch set for the ACIS Flight Software. The purpose of this release is to add new patches to the standard set, resulting in a Rev. C Standard Patch release, and to retest all optional patches against this release.

Although the patches listed in this release have been tested in combination with the standard patch release, they have NOT been tested in various combinations with each other as part of this release. Each needed combination will be provided a distinct part number, and will be released individually, based on the patches provided in this release.

This release relies on the patches produced by Revision C of the standard patch release. Refer to MIT 36-58010, Rev. C.

This release consists of the following optional flight patches, where * indicates additional or modified patches since the previous release:

- cc3x3 - Continuous Clocking 3x3 Event Mode
- ccignore - Ignore Continuous Clocking data frames
- compressall - Fixes SPR 134
- ctireport1 - Reports precursor charge
- ctireport2 - Reports precursor charge
- eventhist - Timed Exposure Event Histogram Mode
- reportgradel - Addresses SPR 132
- smtimedlookup - Supports eventhist and ctireport*
- teignore - Ignore Timed Exposure data frames
- untricklebias - Fixes SPR 133

This release also contains a set of informally controlled engineering patches, used for ground testing, debugging and experimentation:

- hybrid - Prototype of a hybrid clocking mode
 - squeegy - Prototype of a squeegee clocking mode
 - fepbiasparity1 - Prototype of the fepbiasparity2 patch
 - forcebiastrickle - Patch to set trickleBias flag
 - tlmio - Telemetry Standard I/O Utility Routines
 - printswhouse - Print S/W Housekeeping reports in realtime
 - deaeng - Detect/configure for DEA Engineering video boards
 - dearepl - Stubs for use when a DEA is not attache
-

Addressed Problem Reports:

SPR-134
SPR-126
SPR-132
SPR-133
SPR-120
SPR-124

Included Patches:

cc3x3 (4636 bytes)
ccignore (36 bytes)
compressall (2368 bytes)
ctireport1 (5452 bytes, depends on smtimedlookup)
ctireport2 (2784 bytes, depends on smtimedlookup)
deaeng (2604 bytes, depends on tlmio, conflicts with dearepl)
dearepl (556 bytes, conflicts with deaeng)
eventhist (5908 bytes, depends on smtimedlookup)
printswhouse (7224 bytes, depends on tlmio)
reportgradel (816 bytes)
smtimedlookup (3712 bytes)
teignore (36 bytes)
tlmio (10312 bytes)
untricklebias (1612 bytes)

=====
Patch Name: reportgradel

Part Number: 36-58030.22
Version: A
SCO: 36-1021
Environment: flight

Conflicts:
Depends On:
Size: 816 bytes

Bcmd File: opt_reportgradel.bcnd
Pkts File: opt_reportgradel.pkts

Description:

This patch reports per-FEP event filtering statistics via software housekeeping. The SwHousekeeper constructor is patched in order to add an extra 54 housekeeping codes, 9 per FEP, as follows:

```
SW_FILT_NONE,      /* events unfiltered */
SW_FILT_ENERGY,    /* events filtered by energy */
SW_FILT_GRADE1,    /* events filtered by SW_GRADE_CODE1 */
SW_FILT_GRADE2,    /* events filtered by SW_GRADE_CODE2 */
SW_FILT_GRADE3,    /* events filtered by SW_GRADE_CODE3 */
SW_FILT_GRADE4,    /* events filtered by SW_GRADE_CODE4 */
SW_FILT_GRADE5,    /* events filtered by SW_GRADE_CODE5 */
SW_FILT_OTHER,     /* events filtered by other grade */
SW_FILT_WIN,       /* events filtered by window */
```

These SwStatistic codes begin at a value of SWSTAT_FILTER_BASE. They are defined in "acis_h/interface.h", along with the 5 special grade codes:

```
SW_GRADE_CODE1 = 24,
SW_GRADE_CODE2 = 66,
SW_GRADE_CODE3 = 107,
SW_GRADE_CODE4 = 214,
SW_GRADE_CODE5 = 255
```

Thus, the number of grade 214 events rejected by FEP_3 during the current housekeeping interval will be reported in swHousekeeping packets with a "statistics[].swStatisticId" value of SWSTAT_FILTER_BASE+SW_FILT_GRADE4+(9*FEP_3). The corresponding "statistics[].count" field will contain the number of events in this particular class from this particular FEP during the current ~64 sec housekeeping interval. As an aide to synchronizing housekeeping data and event packets, the "statistics[].value" field will contain the most recent exposure number read from this FEP during this interval.

Applicable Reports/Requests:
SPR-132

Test Results:
testTe --> PASS
testCc --> PASS

Replaced Functions:

PmEvent::filterEvent

Command Impact:

None.

Telemetry Impact:

No reduction of telemetry throughput is anticipated. To identify the new housekeeping fields, ground software must recognize the new SwStatistic codes. Refer to the ACIS Software IP&CL Release Notes, Rev. L or later, for details

Science Impact:

None.

=====
Patch Name: untricklebias

Part Number: 36-58030.28
Version: A
SCO: 36-1028
Environment: flight

Conflicts:
Depends On:
Size: 1612 bytes

Bcmd File: opt_untricklebias.bcnd
Pkts File: opt_untricklebias.pkts

Description:

For reasons unknown, the BEP has occasionally run the science and bias thief tasks simultaneously. This causes the FEPs to start searching for x-ray events while the BEP is copying their bias maps to telemetry. If the threshold crossing frequency is sufficiently high, this can trigger an error in the FEP firmware leading to a "T-plane latchup" condition.

The untricklebias patch prevents this behavior by ensuring that the FEP bias maps are never accessed by the BiasThief task. Instead, the science task is given these functions.

The main routine of the bias thief task is repaced by Test_BiasThief::goTaskEntry, which does nothing beyond waking up whenever the task monitor tells it to, but goes back to sleep again immediately.

Where necessary, the remaining BiasThief methods that are called from the science task are replaced by methods that do not notify the bias thief task that a change has been made. The trickleTeBias and trickleCcBias do not need to be patched, but the checkMonitor method must be replaced with a version that is appropriate for being called from the science task. Note that it tests the EV_SM_BIAS_ABORT_RUN in the event mask: this is the value appropriate for a science task abort.

Applicable Reports/Requests:
SPR-133

Test Results:

patchTe --> PASS
patchAll --> PASS
patchCc --> PASS

Replaced Functions:

BiasThief::abort
ScienceMode::waitForBiasTrickle
BiasThief::goTaskEntry
BiasThief::biasReady
BiasThief::checkMonitor

08/15/07
00:24:27

Flight S/W Patches, Revision C-C-D
../dist/options-release-C-opt-C.notes

7

Command Impact:
None.

Telemetry Impact:
None.

Science Impact:
None.

=====
Patch Name: deaeng

Part Number: 36-58030.11
Version: 02
SCO: 36-1010
Environment: engineering

Conflicts: dearepl
Depends On: tlmio
Size: 2604 bytes

Bcmd File: opt_deaeng.bcnd
Pkts File: opt_deaeng.pkts

Description:

This patch provides the basic capability to detect and communicate with the engineering version of the DEA CCD controller boards. For historical reasons, these boards have a different interface than the flight CCD controllers.

This patch relies on printf() being installed (see tlmio).

Applicable Reports/Requests:
TOOL-PENDING

Test Results:
No Tests Specified

Replaced Functions:
DeaCcdController::updateRegister
DeaCcdController::powerOn
DeaCcdController::writeData

Command Impact:
This patch will determine the type of video boards installed in the system. Due to the interface differences between boards, high-speed tap commands will not work on engineering video boards, but will continue to work on "flight-like" video boards.

Telemetry Impact:
Since this patch calls printf(), it will result in TTAG_USER telemetry packets.

Science Impact:
N/A

=====
Patch Name: cc3x3

Part Number: 36-58030.06
Version: B
SCO: 36-1018
Environment: flight

Conflicts:
Depends On:
Size: 4636 bytes

Bcmd File: opt_cc3x3.bcmd
Pkts File: opt_cc3x3.pkts

Description:

This patch implements the Continuous Clocking 3x3 Event Mode. In this mode, the instrument performs the standard continuous clocking manipulation of the CCDs, but rather than accept and telemetry 1x3 events, the mode processes 3x3 event islands, improving the spectral performance of the mode and reducing the problems associated with vertically split events.

Because the Continuous Clocking parameter block only provides 4 bits for defining the grade selection for the mode (in 1x3, only 4 bits were necessary), this patch provides table which maps the 4-bit code into a set of pre-built 256-bit grade selection masks. In this release, the grade selection map is populated with masks provided by Fred Baganoff. Refer to grade_table.html for a description of the grade families. The following table summarizes the selections:

- Code 0 - Reject all grades
- Code 1 - Reject ASCA grades 1,2,3,4,5,6,7
- Code 2 - Reject ASCA grades 1,5,6,7
- Code 3 - Reject ASCA grades 1,5,7
- Code 4 - Undefined (currently rejects all grades)
- Code 5 - Undefined (currently rejects all grades)
- Code 6 - Undefined (currently rejects all grades)
- Code 7 - Reject ACIS flight grades 24,66,107,127,214,223,248,251,254,255
- Code 8 - Reject ACIS flight grades 24,107,127,214,223,248,251,254,255
- Code 9 - Reject ACIS flight grades 24,66,107,214,248,255
- Code 10 - Reject ACIS flight grades 24,66,107,214,255
- Code 11 - Reject ACIS flight grades 24,107,214,248,255
- Code 12 - Reject ACIS flight grades 24,107,214,255
- Code 13 - Reject ASCA grade 7
- Code 14 - Reject ACIS flight grade 255
- Code 15 - Accept all grades

NOTE: CC3x3 Codes 0 and 15 have the same effect as their numerical equivalents in CC1x3, where 0 will reject all events, and 15 will accept events with any grade code.

Applicable Reports/Requests:
SPR-126
SPR-120
SPR-124

Test Results:

unit --> PASS
smoke --> PASS

Replaced Functions:

SmContClocking::setupFepBlock
SmContClocking::setupProcess
SmContClocking::terminate

Command Impact:

This version of CC3x3 uses different grade sets than the previous version. This may have an impact on the grade selection field of CC Parameter Block command packets already built for CC3x3 observations.

This mode is invoked by using the FEP_CC_MODE_EV3x3 (2) in the fepMode field of the Continuous Clocking Parameter block, in conjunction with any of the BEP_CC event processing modes for the bepPackingMode field. This restricts the use of this mode to CC Faint and CC Graded modes. This patch does NOT support other Timed Exposure derived modes, such as Faint with Bias, 5x5, nor any of the existing nor patched histogram modes.

At the onset of a CC3x3 science run, the run will force two resets and reloads of the FEP software, the first to ensure that the boot-strap code is in the FEPs, and the second to load the patch code into the FEPs. This will always add up to 14 seconds per FEP to the start-up time of the run, compared to runs where the FEPs were already loaded and running.

To ensure that the patch is not present at the start of the next run, which may or may not be a CC3x3 run, a CC3x3 science run will always force the FEPs into a reset state at the end of the run. This will add another 7 seconds per FEP to the start up time of the run following a CC3x3 run, relative to the normal start up time, where the FEPs were already loaded and running.

These resets will also impact the power consumption of ACIS, where the system will draw up to 16 watts less than normal (with all 6 on and running) while the FEPs are held a reset state.

Refer to the ACIS Software IP&CL Structure Definitions, Rev. L or later for details.

Telemetry Impact:

This mode defines 4 new telemetry packet types.

When configured for FEP_CC_MODE_EV3x3 and BEP_CC_MODE_FAINT, the patch produces TTAG_SCI_CC_REC_FAINT3x3 exposure records and TAG_SCI_CC_DAT_FAINT3x3 event data packets.

When configured for FEP_CC_MODE_EV3x3 and BEP_CC_MODE_GRADED, it produces TTAG_SCI_CC_REC_GRADED3x3 exposure records and TTAG_SCI_CC_DAT_GRADED3x3 event data packets.

The size of and overhead of these packets are the same as their Timed Exposure counterparts, TTAG_SCI_TE_REC_FAINT3x3, TTAG_SCI_TE_DAT_FAINT3x3, TTAG_SCI_TE_REC_GRADED3x3 and TTAG_SCI_TE_DAT_GRADED3x3.

When used, a CC3x3 science run will produce additional Software Housekeeping counts to the FEP write and execute statistics, reflecting the additional resets and reloads of the FEPs. Runs immediately following a CC3x3 run will also produce additional FEP related counts, as they load and run the reset FEPs.

Refer to the ACIS Software IP&CL Structure Definitions, Rev. L or later for details

Science Impact:

This version of CC3x3 uses different grade sets than the previous version. The ground data analysis software may have to be aware of which version of CC3x3 is installed for a given set of CC3x3 data. Please refer to the ACIS command generation system for the set of ACIS Software Version identifiers (telemetered in the BEP Startup Message and in each Software Housekeeping telemetry packet) corresponding to the different installed CC3x3 versions.

This mode produces a new type of data product, consisting of 3x3 islands around accepted events in Continuous Clocking mode. This is intended to provide better spectral resolution and event detection performance when in Continuous Clocking mode.

This mode will not report events on row 0 and row 511, leaving a 2-row timing gap with a period of 512 rows.

As in other Continuous Clocking modes, no bias errors will be reported when in this mode, since the bias map is extremely redundant (there's 512 copies of the bias value for any given column).

=====
Patch Name: tlmio

Part Number: 36-58030.07
Version: 02
SCO: 36-1010
Environment: flight

Conflicts:
Depends On:
Size: 10312 bytes

Bcmd File: opt_tlmio.bcmd
Pkts File: opt_tlmio.pkts

Description:

This patch provides basic standard I/O functions which emit TTAG_USER telemetry packets containing data written via calls to write().

This patch stubs the functions open(), close() and read(), and implements the function write(), used by higher level I/O library functions, such as printf().

The patch maintains a 1024 word telemetry buffer just at the end of bulk memory. write() appends data to this buffer until either the buffer fills, or until a newline is written. Once write() fills the buffer or a newline is encountered, the telemetry buffer is sent as follows:

1. Interrupts are disabled
2. The hardware is polled until the current packet is finished.
3. The packet buffer header is filled in, and the first data word is set to 0 (a hook used to support different subtypes of TTAG_USER).
4. Transfer the packet
5. Wait for the transfer to complete
6. If no transfer was in progress prior to the interrupt disable, clear the pending interrupt caused by the TTAG_USER packet transfer
7. Reset the the buffer contents
8. Reenable interrupts

Applicable Reports/Requests:
TOOL-PENDING

Test Results:
No Tests Specified

Replaced Functions:

Command Impact:
None

Telemetry Impact:
If this patch is used by client code (this patch itself doesn't

initiate any messages), it will emit telemetry packets consisting of the tag TTAG_USER. The format of these packets consist of the standard telemetry header, followed by 1 32-bit word containing a zero, followed by the number of data words indicated by the packet length. If the clients of the patch issue "printf" calls, the data will consist of a single null-terminated ascii string.

Word 0: SYNC (0x736f4166)
Word 1: [0..9] Length (3 + "n"/4)
Word 1: [10..31] TTAG_USER
Word 2: 0
Word 3..Length: Data

Science Impact:

Since this patch "plays" with the hardware and telemetry software, the use of this patch may interfere with the smooth operation of science runs.

=====
Patch Name: compressall

Part Number: 36-58030.27
Version: A
SCO: 36-1027
Environment: flight

Conflicts:
Depends On:
Size: 2368 bytes

Bcmd File: opt_compressall.bcnd
Pkts File: opt_compressall.pkts

Description:

This patch ensures that all raw mode packets are written to the telemetry stream without data loss. It eliminates the prior behavior in which, if a compressed pixel row was too long to fit into an output packet, the entire row was skipped and a zero-data-length was telemetered.

In the new version, rows that are too long when compressed are written uncompressed, with the telemetry packet header fields rewritten to indicate that that particular packet is uncompressed.

Applicable Reports/Requests:
 SPR-134

 SER-none

Test Results:
 reproduce --> PASS
 fix --> PASS

Replaced Functions:
 PmCcRaw::digestRawRecord
 PmTeRaw::digestRawRecord

Command Impact:
 None.

Telemetry Impact:
 Ground software must examine the compressionTableSlotIndex and compressionTableIdentifier fields of all dataCcRaw and dataTeRaw packets. If their values are 255 and 0, respectively, the pixel array should not be decompressed.

Science Impact:
 None. Raw mode is intended for diagnostic purposes only.

=====
Patch Name: ccignore

Part Number: 36-58030.10
Version: A
SCO: 36-1004
Environment: flight

Conflicts:
Depends On:
Size: 36 bytes

Bcmd File: opt_ccignore.bcnd
Pkts File: opt_ccignore.pkts

Description:
This patch causes the FEP to ignore "ignoreInitialFrames"
frames of data at the onset of Continuous Clocking data processing.

Applicable Reports/Requests:
SER-PENDING

Test Results:
smoke --> PASS

Replaced Functions:

Command Impact:
This patch will cause the start up time of a Continuous
Clocking run to increase by "ignoreInitialFrames" times
the frame rate configured for the run. If "ignoreInitialFrames"
is less than 2, the 2 frames will be skipped.

Telemetry Impact:
When "ignoreInitialFrames" is greater than 2,
the first telemetered Continuous Clocking exposure number
will be "ignoreInitialFrames", rather than "2".

Science Impact:
This may reduce the amount of noise in the early
telemetered frames of the Continuous Clocking run by
running the CCDs longer before processing and sending the data.

=====
Patch Name: eventhist

Part Number: 36-58030.05
Version: B
SCO: 36-1025
Environment: flight

Conflicts:
Depends On: smtimedlookup
Size: 5908 bytes

Bcmd File: opt_eventhist.bcnd
Pkts File: opt_eventhist.pkts

Description:

This patch implements the Event Histogram Mode. In this mode, the instrument performs the standard timed exposure clocking, and event detection and filtering, but rather than send the events to telemetry, the instrument builds CCD quadrant specific histograms of the summed corrected pulse heights of the accepted events. These histograms contain bins 0 through 4095. Events with a pulse height above 4095 are counted in bin 4095 and events with a negative value are counted in bin 0. All histogram bin values consist of a 26-bit count, followed by 5-bit of Hamming error detection/correction code, and 1 spare bit. The code is capable of detecting and correcting 1-bit errors in the count and hamming code bits.

Important: This version of the eventhist patch will only run correctly if the smtimedlookup patch is also loaded.

Applicable Reports/Requests:

Test Results:

smoke --> PASS
smoke2 --> PASS

Replaced Functions:

smTimedLookup3x3[3]
smTimedLookup5x5[3]

Command Impact:

As in normal Raw Histogram Mode, Event Histogram mode can only be used for Timed Exposure Science runs, and not in Continuous Clocking runs.

This mode is invoked by using the FEP_TE_MODE_EV3x3 or FEP_TE_MODE_EV5x5 for the fepMode field of the Timed Exposure Parameter Block, in conjunction with the new BEP_TE_MODE_EVHIST (3) for the bepPackingMode field.

Refer to the ACIS Software IP&CL Structure Definitions, Rev. M for details.

Telemetry Impact:

This mode defines new telemetry formats, TTAG_SCI_TE_REC_EV_HIST for exposure records, and TTAG_SCI_TE_DAT_EV_HIST for histogram data

packets. This new mode now places the count of error corrections performed on the quadrant's histogram bins within the previously unused "Variance Overclock High" of the exposure record, TTAG_SCI_TE_REC_EV_HIST. The Rev. M version of IP&CL renames this field accordingly.

The size of these packets are the same as those for TTAG_SCI_TE_REC_HIST and TTAG_SCI_TE_DAT_HIST respectively.

This mode always requires 10 telemetry buffers for each quadrant it accumulates (9 data buffers + 1 exposure record buffer per histogram). When accumulating histograms from all 4 quadrants on all 6 CCDs, the system requires 216 data buffers, and once the histograms are complete, it requires an additional 24 exposure record buffers. ACIS is configured for 400 science telemetry buffers, and as such, has enough buffering to accumulate only 1 complete set of histograms at a time. This will cause time gaps between sets of histograms when no events are accumulated. These gaps will consist of complete exposures, so partial exposures will not be accumulated in the histograms. As the previous buffers are telemetered and released back to the telemetry pool, eventually enough buffers (to be exact, 56) will be available to hold the 2nd set of histograms. At 24Kbps (format 2), this results in a time gap on the order of half a minute to a minute, and, at 500bps (format 1), a gap on the order of a half an hour to 45 minutes.

The total transmission time for a set of histograms at 24Kbps is about 3 minutes, whereas at 500bps, it starts approaching 2 hours.

If only 5 CCDs are used, ACIS can double-buffer the histograms, eliminating this gap, assuming that the histogram count times the frame time (exposure time + overhead) is large enough to accommodate the transmission time of the histograms. The total transmission time for 5 CCDs at 24Kbps is about 2 minutes, and at 500bps, the transmission time approaches 1.5 hours.

Details of these formats are described in the ACIS Software IP&CL Structure Definitions, Rev. M.

Science Impact:

This mode produces a new type of data product, histograms of the corrected and summed pulse heights from filtered events.

=====
Patch Name: printswhouse

Part Number: 36-58030.08
Version: 01
SCO: 36-986
Environment: flight

Conflicts:
Depends On: tlmio
Size: 7224 bytes

Bcmd File: opt_printswhouse.bcnd
Pkts File: opt_printswhouse.pkts

Description:
This patch provides a diagnostic which prints software housekeeping reports to telemetry in real-time, using the tlmio package.

Applicable Reports/Requests:
TOOL-PENDING

Test Results:
No Tests Specified

Replaced Functions:
SwHousekeeper::report

Command Impact:
None

Telemetry Impact:
This patch will cause the system to emit TTAG_USER packets containing a null terminated string, which describes the software housekeeping element currently being reported. See a description of the tlmio patch, MIT 36-58030.07.

Science Impact:
See the tlmio patch, 36-58030.07

=====
Patch Name: dearepl

Part Number: 36-58030.12
Version: 02
SCO: 36-1010
Environment: engineering

Conflicts: deaeng
Depends On:
Size: 556 bytes

Bcmd File: opt_dearepl.bcnd
Pkts File: opt_dearepl.pkts

Description:

This patch provides the basic capability to fake the existence of a DEA. This patch is used when no DEA box is available, or one wants to test without actually talking to the DEA.

Applicable Reports/Requests:
TOOL-PENDING

Test Results:
No Tests Specified

Replaced Functions:

DeaDevice::sendCmd
DeaManager::writeData
DeaManager::checkLoads
DeaDevice::isReplyReady
DeaCcdController::updateRegister
DeaDevice::readReply
DeaDevice::isCmdPortReady

Command Impact:

This "fakes" the existence of the DEAs. Commands which read and write PRAM, SRAM or DEA hardware will not crash, but won't work either.

Telemetry Impact:

This will produce true fiction from the DEAs.

Science Impact:

Can't do any, since the patch replaces the interface to the real DEAs.

=====
Patch Name: teignore

Part Number: 36-58030.09
Version: A
SCO: 36-1003
Environment: flight

Conflicts:
Depends On:
Size: 36 bytes

Bcmd File: opt_teignore.bcnd
Pkts File: opt_teignore.pkts

Description:
This patch causes the FEP to ignore "ignoreInitialFrames"
frames of data at the onset of Timed Exposure data processing.

Applicable Reports/Requests:
SER-PENDING

Test Results:
smoke --> PASS

Replaced Functions:

Command Impact:
This patch will cause the start up time of a Timed Exposure
run to increase by "ignoreInitialFrames" times the frame
rate configured for the run. If "ignoreInitialFrames"
is less than 2, the 2 frames will be skipped.

Telemetry Impact:
When "ignoreInitialFrames" is greater than 2,
the first telemetered exposure number will be
"ignoreInitialFrames", rather than "2".

Science Impact:
This may reduce the amount of noise in the early
telemetered frames of the Timed Exposure run by running
the CCDs longer before processing and sending the data.

=====
Patch Name: smt timedlookup

Part Number: 36-58030.24
Version: A
SCO: 36-1025
Environment: flight

Conflicts:
Depends On:
Size: 3712 bytes

Bcmd File: opt_smtimedlookup.bcnd
Pkts File: opt_smtimedlookup.pkts

Description:

This patch replaces several "switch" statements in SmTimedExposure class methods with a set of lookup tables indexed by the value of the BepMode and FepMode fields from the current TE parameter block. If a table slot is empty, the corresponding mode will be treated as unimplemented. With this patch, it is therefore possible to add more than one new TE mode via optional patches without the need to deliver a version of each patch for every possible combination of the other patches. The following methods, tables, and indices are used:

Method	lookup table	index
SmTimedExposure::setupProcess	smTimedLookupMode smTimedLookup3x3 smTimedLookup5x5	FepMode BepPackingMode BepPackingMode
SmTimedExposure::setupFepBlock	smTimedSetupFep	FepMode
SmTimedExposure::terminate	smTimedTerminate	FepMode

These tables may be patched by an extension of the "func" directive in the *.pkg file used to describe an ACIS patch. Hence, the line

```
func smTimedLookupMode[4] Test2_SmTimedExposure::setupCtil
```

instructs the linker to insert the address of the setupCtil() method of the Test2_SmTimedExposure class into slot 4 of the smTimedLookupMode table, so that setupCtil() will be called when FepMode == 4.

Applicable Reports/Requests:

Test Results:
smoke --> PASS

Replaced Functions:
SmTimedExposure::terminate
SmTimedExposure::setupProcess
SmTimedExposure::setupFepBlock

Command Impact:

None.

Telemetry Impact:
None.

Science Impact:
None.

=====
Patch Name: ctireport1

Part Number: 36-58030.25
Version: A
SCO: 36-1026
Environment: flight

Conflicts:
Depends On: smtimedlookup
Size: 5452 bytes

Bcmd File: opt_ctireport1.bcnd
Pkts File: opt_ctireport1.pkts

Description:

This patch implements a variant of timed-exposure 3x3 faint event mode in which the presence of precursor charge in each of the three columns that can contribute to each event is encoded in the 16 "outlying" pixels of Te5x5 mode.

FEP patches are loaded after the default code by two additional calls to `fehManager.loadRunProgram` from `Test2_SmTimedExposure::setupCtilFep`. Once loaded, the FEPs are marked as having been reset, thereby causing the following run to reload their default code.

Within the FEP, additional stack space is reserved for the `ctilstk` structure that holds the row indices and bias-subtracted pixel values of the most recently located precursor charge in each CCD column.

The new `FEPtestCtil` routine is called from an inline patch within `FEPsciTimedEvent` in advance of the `FEPtestOddPixel` or `FEPtestEvenPixel` routines. When a threshold crossing is detected, `FEPtestCtil` clears the `ctilstk` array (if this is a new frame), calls `FEPtestOddPixel` or `FEPtestEvenPixel`, and then pushes the pixel value and row index onto `ctilstk`. If `ctilstk` is full, the most distant (by row) value is dropped.

`FEPappendCtil` is called by the patched FEP code in place of the original `FEPappend5x5` routine. It determines the maximum bias-subtracted pixel value in each column, then inspects the `ctilstk` stacks for those columns, and packs up to 15 precursor charge values (adu and row) into elements 1 through 15 of the `pe[]` array:

```
pe[i] = STORE_PIX(pixel - bias - delta_overclock, row_index)
```

`pe[0]` contains three 4-bit fields, the number of successive `pe[]` precursor values corresponding to `col-1`, `col`, and `col+1` of the event.

Applicable Reports/Requests:

Test Results:
smoke --> PASS

Replaced Functions:
smTimedLookupMode[4]

```
smTimedTerminate[4]
smTimedSetupFep[4]
```

Command Impact:

This patch requires that the `smtimedlookup` patch must also be loaded. Once loaded, it is invoked by setting `fepMode = FEP_TE_MODE_CTII1` in a `loadTeBlock` packet, writing that packet to a parameter block slot, and then starting a timed-exposure science run from that slot. The uplink format is defined in the ACIS IP&CL document 36-53204.0204 Rev. N.

Telemetry Impact:

The downlinked exposure and event data packets are identical in format to `exposureTeFaint` and `dataTeVaryFaint` except that their `formatTag` fields contain `TTAG_SCI_TE_REC_CTII1` and `TTAG_SCI_TE_DAT_CTII1`, respectively. When a `TTAG_SCI_TE_DAT_CTII1` is received, precursor charge data will be located in the `dataTeVaryFaint.pulseHeights` array, as follows:

```
pulseHeights[0]           - three 4-bit counters
pulseHeights[1..5,9,10,14,15,19..24] - precursor ADU and row
```

The sub-fields of `pulseHeights[0]` determine the contents of the other 15 fields:

```
ncol[0] = (pulseHeights[0] >> 8) & 15 -
ncol[1] = (pulseHeights[0] >> 4) & 15 -
ncol[2] = pulseHeights & 15           -
```

The fields from `icol-1`, if any, are written starting at `pulseHeights[1]`, followed by those from `icol`, and finally those from `icol+1`. The ADU values are stored in the 7 most significant bits of `pulseHeights[]` and the row indices in the least significant 5 bits, and should be extracted as follows:

```
adu = pulseHeights[i] & 0xfe0;
row = (pulseHeights[i] & 0x01f) << 5;
```

Unused `pulseHeights[]` will be filled with zeroes.

Science Impact:

This patch is intended for on-orbit diagnostic use only.

=====
Patch Name: ctireport2

Part Number: 36-58030.26
Version: A
SCO: 36-1026
Environment: flight

Conflicts:
Depends On: smtimedlookup
Size: 2784 bytes

Bcmd File: opt_ctireport2.bcnd
Pkts File: opt_ctireport2.pkts

Description:

This patch implements a variant of timed-exposure 3x3 faint event mode in which the presence of precursor charge in each of the three columns that can contribute to each event is encoded in the low-order bits of three of the corner pixels.

FEP patches are loaded after the default code by two additional calls to `fehManager.loadRunProgram` from `Test3_SmTimedExposure::setupCtilFep`. Once loaded, the FEPs are marked as having been reset, thereby causing the following run to reload their default code.

Within the FEP, additional stack space is reserved for the `cti2stk` structure that holds the row indices of the most recently located precursor charge in each CCD column.

The new `FEPtestCti2` routine is called from an inline patch within `FEPsciTimedEvent` in advance of the `FEPtestOddPixel` or `FEPtestEvenPixel` routines. When a threshold crossing is detected, `FEPtestCti2` clears the `cti2stk` array (if this is a new frame), calls `FEPtestOddPixel` or `FEPtestEvenPixel`, and then updates `cti2stk` to indicate that this column contains charge.

`FEPappendCti2` is called by the patched FEP code instead of the original `FEPappend5x5`. It finds the maximum of the 4 corner pixels of the event that is being reported. Then it determines whether any of the three contributing columns contained precursor charge. Finally, it encodes this information in the low order bytes of the three smallest corner pixels. (Since the low-order bit of each corner pixel may be replaced, only the 11 high-order bits are compared when determining the maximum value).

Applicable Reports/Requests:

Test Results:
smoke --> PASS

Replaced Functions:
smTimedLookupMode[5]
smTimedTerminate[5]
smTimedSetupFep[5]

Command Impact:

The uplink format is defined in the ACIS IP&CL document 36-53204.0204 Rev. N. The `fepMode` field in the `loadTeBlock` command packet must be set equal to `FEP_TE_MODE_CTI2`. Unless the `smtimedlookup` patch has also been loaded, this value will cause a subsequent `startScience` command that references this parameter block to fail.

Telemetry Impact:

The downlinked exposure and event data packets are identical in format to `exposureTeFaint` and `dataTeFaint`. To process the precursor charge information, ground software must first inspect the `loadTeBlock` reported in the `dumpedTeBlock` packet that started the run. If the `fepMode` field is equal to `FEP_TE_MODE_CTI2`, subsequent `dataTeFaint` packets should be inspected. The following code fills `ee[i]` with one (zero) according to whether column (`ccdColumn+i-1`) did (did not) contain precursor charge:

```
unsigned nn, mm, ii, ee[3];

for (mm = 0, nn = 2; nn < 9; nn++) {
    if ((nn & 1) == 0 && nn != 4) {
        if ((pulseHeights[nn] & 0xffe) > (pulseHeights[mm] & 0xffe))
            mm = nn;
    }
}
for (nn = ii = 0; nn < 9; nn++) {
    if ((nn & 1) == 0 && nn != 4 && nn != mm) {
        ee[ii++] = pulseHeights[nn] & 1;
    }
}
```

Science Impact:

This patch is intended for on-orbit diagnostic use only.

```
/* =====  
*  
* $$Source: /nfs/acis/h3/acisfs/confignt1/patches/buscrash/buscrash.C,v $$  
*  
* Patch Name: Bus Crash Prevention  
*  
* Description:  
* This defines a C++ replacement function to FepManager::loadBadPixel()  
*  
* References:  
* Refer to the 1.5 release of filesprotocols/fepmanager.C  
*  
* $$Log: buscrash.C,v $  
* $Revision 1.4 2007/08/14 16:09:36 pgf  
* $Add friend statement  
* $  
* $Revision 1.3 2007/07/14 22:48:29 pgf  
* $Change method from static to virtual  
* $  
* $Revision 1.2 2007/04/18 21:10:57 pgf  
* $Call fepManager.isEnabled to prevent bus crash.  
* $  
* $Revision 1.1 2007/04/17 18:52:35 pgf  
* $Initial version.  
* $$  
*  
* ===== */
```

```
#include <stdio.h>  
#include "acis_h/interface.h"  
#include "filesprotocols/fepmanager.H"  
#include "filesswhouse/swhousekeeper.H"
```

```
class Test_FepManager  
{  
public:  
    virtual void loadBadPixel(FepId fepid, unsigned row, unsigned col);  
    friend class Test2_FepManager;  
};  
  
void Test_FepManager::loadBadPixel(FepId fepid, unsigned row, unsigned col)  
{  
    DebugProbe probe;  
  
    if (fepManager.isEnabled(fepid) == BoolTrue) {  
        fepIo[fepid]->writeBiasValue(row, col, PIXEL_BAD);  
    }  
}
```

```
#!/bin/env expect

puts "Welcome to buscrash/testsuite/bug-hw/runtest.tcl"

# ---- Split off the command arguments ----
set basedir [lindex $argv 0]
set tools [lindex $argv 1]
set patchdir [lindex $argv 2]

# ---- Launch the command and telemetry server processes ----
set first_fep 0 ; # first FEP under test
set last_fep 0 ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD" ; # desired outputRegisterMode
set ccd_list "0 10 10 10 10 10" ; # desired fepCcdSelect

# ---- Embed procedure library ----
source $basedir/$tools/lib/lib-exp/runtest_support.tcl

# ---- Sleep while reporting packets ----
proc gotosleep { secs } {
    set timeout $secs
    expect { timeout { } }
}

# ---- Start command pipe ----
spawn $basedir/$tools/bin/cmdclient $env(ACISSERVER)
set cmd_id $spawn_id

# ---- Start telemetry pipe ----
spawn $basedir/$tools/bin/tlmclient $env(ACISSERVER)
gotosleep 1

# ---- Select Input from Image Loader ----
system make loaderselect

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcmod\
                 $basedir/$tools/share/opt_printswhouse.bcmod\
                 $basedir/$tools/share/opt_dearepl.bcmod\
                 ../standard.bcmod"

warm_boot

# ---- Power on FEPs and CCDs ----
power_on_boards "$ccd_list"

# ---- Wait for FEPs to finish powering ----
expect {
    -re ".*SWSTAT_FEPMAN_ENDLOAD: $last_fep\[\\r\\n]*" { }
    timeout { fail "Power-up Failure" }
}

# ---- Load Pblock for Faint Timed-Exposure Mode ----
send -i $cmd_id "load 0 te 4 {
    parameterBlockId      = 0x00000014
    fepCcdSelect          = $ccd_list
    fepMode                = 2 # FEP_TE_MODE_EV3x3
    bepPackingMode        = 2 # BEP_TE_MODE_GRADED
    onChip2x2Summing      = 0
    ignoreBadPixelMap     = 0
    ignoreBadColumnMap    = 0
    recomputeBias         = 1
    trickleBias           = 1
    subarrayStartRow      = 0
}
```

```
subarrayRowCount          = 1023
overclockPairsPerNode    = 8
outputRegisterMode       = $quad_mode
ccdVideoResponse         = 0 0 0 0 0 0
primaryExposure          = 33
secondaryExposure        = 0
dutyCycle                = 0
fep0EventThreshold       = 100 100 100 100
fep1EventThreshold       = 100 100 100 100
fep2EventThreshold       = 100 100 100 100
fep3EventThreshold       = 100 100 100 100
fep4EventThreshold       = 100 100 100 100
fep5EventThreshold       = 100 100 100 100
fep0SplitThreshold       = 50 50 50 50
fep1SplitThreshold       = 50 50 50 50
fep2SplitThreshold       = 50 50 50 50
fep3SplitThreshold       = 50 50 50 50
fep5SplitThreshold       = 50 50 50 50
fep4SplitThreshold       = 50 50 50 50
fep5SplitThreshold       = 50 50 50 50
lowerEventAmplitude      = 0
eventAmplitudeRange      = 65535
gradeSelections          = 0xffffffff 0xffffffff 0xffffffff 0xffffffff
                          0xffffffff 0xffffffff 0xffffffff 0xffffffff
windowSlotIndex          = 65535
histogramCount           = 0
biasCompressionSlotIndex = 3 3 1 1 1 1
rawCompressionSlotIndex = 0
ignoreInitialFrames      = 2
biasAlgorithmId          = 1 1 1 1 1 1
biasArg0                 = 9 9 9 9 9 1
biasArg1                 = 25 25 25 25 25 25
biasArg2                 = 20 20 20 20 20 20
biasArg3                 = 26 26 50 50 50 50
biasArg4                 = 20 20 20 20 20 20
fep0VideoOffset          = 65 65 65 65
fep1VideoOffset          = 65 65 65 65
fep2VideoOffset          = 65 65 65 65
fep3VideoOffset          = 65 65 65 65
fep4VideoOffset          = 65 65 65 65
fep5VideoOffset          = 65 65 65 65
deaLoadOverride          = 0
fepLoadOverride          = 0
}
"
command_echo 1 9 "load te"
system make bias

puts ""
puts "# Starting test 1"
puts ""
send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 "start science run"
set timeout 360
expect {
    -re "SWSTAT_FEP_STARTBIAS.*[\r\n]*" { }
    timeout { fail "Bias Failure" }
}
gotosleep 10

puts "# stopScience"
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"
gotosleep 2
```

```
puts "# stopScience"
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"
gotosleep 2

puts "# powering boards off"
power_off_boards
set timeout 360
expect {
  -re "bepStartupMessage.*[\r\n]*" {
    pass "Bus crash reproduced"
  }
  -re "scienceReport.*[\r\n]*" {
    fail "Science run ends without bus crash"
  }
  timeout {
    fail "No crash or stopScience"
  }
}

puts "Done"
```

```
#! /bin/env expect

puts "Welcome to buscrash/testsuite/fix-hw/runtest.tcl"

# ---- Split off the command arguments ----
set basedir [lindex $argv 0]
set tools [lindex $argv 1]
set patchdir [lindex $argv 2]

# ---- Launch the command and telemetry server processes ----
set first_fep 3 ; # first FEP under test
set last_fep 3 ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD" ; # desired outputRegisterMode
set ccd_list "10 10 10 1 10 10" ; # desired fepCcdSelect

# ---- Embed procedure library ----
source $basedir/$tools/lib/lib-exp/runtest_support.tcl

# ---- Sleep while reporting packets ----
proc gotosleep { secs } {
    set timeout $secs
    expect { timeout { } }
}

# ---- Start command pipe ----
spawn $basedir/$tools/bin/cmdclient $env(ACISSERVER)
set cmd_id $spawn_id

# ---- Start telemetry pipe ----
spawn $basedir/$tools/bin/tlmclient $env(ACISSERVER)
gotosleep 1

# ---- Select Input from Image Loader ----
system make loaderselect

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcmod\
                 $basedir/$tools/share/opt_printshouse.bcmod\
                 $basedir/$tools/share/opt_dearepl.bcmod\
                 $basedir/$patchdir/standard.bcmod"
warm_boot

# ---- Power on FEPs and CCDs ----
power_on_boards "$ccd_list"

# ---- Wait for FEPs to finish powering ----
expect {
    -re ".*SWSTAT_FEPMAN_ENDLOAD: $last_fep\[\\r\\n]*" { }
    timeout { fail "Power-up Failure" }
}

# ---- Load Pblock for Faint Timed-Exposure Mode ----
send -i $cmd_id "load 0 te 4 {
    parameterBlockId = 0x00000014
    fepCcdSelect = $ccd_list
    fepMode = 2 # FEP_TE_MODE_EV3x3
    bepPackingMode = 2 # BEP_TE_MODE_GRADED
    onChip2x2Summing = 0
    ignoreBadPixelMap = 0
    ignoreBadColumnMap = 0
    recomputeBias = 1
    trickleBias = 1
    subarrayStartRow = 0
}
```

```
subarrayRowCount          = 1023
overclockPairsPerNode    = 8
outputRegisterMode       = $quad_mode
ccdVideoResponse         = 0 0 0 0 0 0
primaryExposure          = 33
secondaryExposure        = 0
dutyCycle                = 0
fep0EventThreshold       = 100 100 100 100
fep1EventThreshold       = 100 100 100 100
fep2EventThreshold       = 100 100 100 100
fep3EventThreshold       = 100 100 100 100
fep4EventThreshold       = 100 100 100 100
fep5EventThreshold       = 100 100 100 100
fep0SplitThreshold       = 50 50 50 50
fep1SplitThreshold       = 50 50 50 50
fep2SplitThreshold       = 50 50 50 50
fep3SplitThreshold       = 50 50 50 50
fep5SplitThreshold       = 50 50 50 50
fep4SplitThreshold       = 50 50 50 50
fep5SplitThreshold       = 50 50 50 50
lowerEventAmplitude      = 0
eventAmplitudeRange      = 65535
gradeSelections          = 0xffffffff 0xffffffff 0xffffffff 0xffffffff
                          0xffffffff 0xffffffff 0xffffffff 0xffffffff
windowSlotIndex          = 65535
histogramCount           = 0
biasCompressionSlotIndex = 3 3 1 1 1 1
rawCompressionSlotIndex = 0
ignoreInitialFrames      = 2
biasAlgorithmId          = 1 1 1 1 1 1
biasArg0                  = 9 9 9 9 9 1
biasArg1                  = 25 25 25 25 25 25
biasArg2                  = 20 20 20 20 20 20
biasArg3                  = 26 26 50 50 50 50
biasArg4                  = 20 20 20 20 20 20
fep0VideoOffset          = 65 65 65 65
fep1VideoOffset          = 65 65 65 65
fep2VideoOffset          = 65 65 65 65
fep3VideoOffset          = 65 65 65 65
fep4VideoOffset          = 65 65 65 65
fep5VideoOffset          = 65 65 65 65
deaLoadOverride          = 0
fepLoadOverride          = 0
}
"
command_echo 1 9 "load te"
system make bias

puts ""
puts "# Starting test 1"
puts ""
send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 "start science run"
set timeout 360
expect {
    -re "SWSTAT_FEP_STARTBIAS.*[\r\n]*" { }
    timeout { fail "Bias Failure" }
}
gotosleep 10

puts "# stopScience"
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"
gotosleep 2
```



```
puts "# stopScience"
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"
gotosleep 2

puts "# powering boards off"
power_off_boards
set timeout 360
expect {
  -re "bepStartupMessage.*[\r\n]*" {
    fail "Bus crash"
  }
  -re "scienceReport.*[\r\n]*" {
    pass "Science run ends without bus crash"
  }
  timeout {
    fail "No crash or stopScience"
  }
}

puts "Done"
```

```
/* =====
*
* $$Source: /nfs/acis/h3/acisfs/confignt1/patches/tlmbusy/tlmbusy.C,v $$
*
* Patch Name: Make TlmManager::post() reentrant to concurrent tasks
*
* Description:
* This implements the standard tlmbusy patch, which prevents the
* BEP from attempting to start two telemetry packets simultaneously
* from separate tasks. It does this by replacing the post() method
* of TlmManager with a function that calls the forbidPreempt() and
* then permitPreempt() methods of TaskManager.
*
* This patch consists of the following files:
*   tlmbusy.C - The replaced TlmManager::post() method.
*
* The tlmbusy.C file defines and implements the following:
*
*   Test_TlmManager - This is a "friendly" subclass of TlmManager
*                     that provides a replacement function.
*   Test_TlmManager - This constructor is provided to avoid
*                     compiler/linker complaints. It is never invoked.
*   ~Test_TlmManager - This destructor is provided to avoid
*                     compiler/linker errors. It is never invoked.
*   post             - This function replaces TlmManager::post and
*                     prevents task switching while it is invoked,
*                     i.e., it prevents the routine from being
*                     called in a reentrant manner from multiple
*                     tasks.
*
* References:
*   Refer to the 1.5 release of filesprotocols/tlmmanager.C
*
* $$Log: tlmbusy.C,v $
* $Revision 1.1 2005/06/08 20:02:11  pgf
* $Initial version of patch.
* $$
* ===== */

#include "ipcl/interface.h"
#include "filesexecutive/task.H"
#include "filesprotocols/tlmmanager.H"

class Test_TlmManager : public TlmManager
{
public:
    ~Test_TlmManager();
    void post(TlmPkt*pkt);
};

Test_TlmManager::~Test_TlmManager() {};

// -----
void Test_TlmManager::post(TlmPkt*pkt)
// -----
{
    DebugProbe probe;

    // ---- Place packet onto queue ----
    sendQueue.enqueuePkt (pkt);

    // ---- Prevent task preemption ----
    taskManager.forbidPreempt();
}
```

```
// ---- If no transfers in progress, start one up ----  
if (curPkt == 0)  
{  
    serviceDevice (0);  
}  
  
// ---- Allow task preemption again ----  
taskManager.permitPreempt();  
};
```

```
#!/bin/env expect
#
# $Source: /nfs/acis/h3/acisfs/confignt1/patches/tlmbusy/testsuite/bug-hw/runtest.tcl,v $
#
# Test telemetry posting by concurrent tasks -- without tlmbusy patch
#

puts "Welcome to tlmbusy/testsuite/bug-hw/runtest.tcl"

# ---- Launch the command and telemetry server processes ----
set basedir [lindex $argv 0] ; # patch base directory
set tools [lindex $argv 1] ; # tool directory
set patchdir [lindex $argv 2] ; # patches under test
set quad_mode "0 \# QUAD_ABCD" ; # desired outputRegisterMode
set ccd_list "0 10 10 10 10 10" ; # desired fepCcdSelect
set last_fep 0 ; # last FEP read out
set nexpo 2000000 ; # limit to number of exposures
set server $env(ACISSEVER) ; # lrtcu/ctue server
set port $env(PORT) ; # lrtcu/ctue server command port

# ---- Embed procedure library ----
source $basedir/$tools/lib/lib-exp/runtest_support.tcl

# ---- Start command pipe ----
spawn /bin/sh -c "( bcmd | cclient $server $port ) > $server.err 2>&1"
set cmd_id $spawn_id
set send_slow {10 0.001}

# ---- Start telemetry pipe ----
spawn /bin/sh -c "filterClient -h $server | psci -m -u"
sleep 1

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcmd\
                 $basedir/$tools/share/opt_printswhouse.bcmd\
                 $basedir/$tools/share/opt_dearepl.bcmd\
                 $basedir/$patchdir/standard.bcmd"
warm_boot

# ---- Power on FEPs and CCDs ----
power_on_boards "$ccd_list"

# ---- While powering up, load the Bias Image ----
system "make bias"

# ---- Wait for FEPs to finish powering ----
expect {
    -re ".*SWSTAT_FEP_EXECMEM: $last_fep\[\r\n]" {}
    timeout {}
}

# ---- Start DEA housekeeping ----
send -s -i $cmd_id "load 0 dea 4 {
    deaBlockId          = 0x00001014
    sampleRate          = 1
    queries = {
        ccdId           = 10
        queryId         = 0
    }
}
"
command_echo 1 13 "load dea"
send -i $cmd_id "start 0 dea 4\r"
```

```
command_echo 1 18 "start dea housekeeping"

# ---- Write long string line by line ----
proc long_send {id str} {
    set cnt 0
    foreach s [split $str "\n"] {
        send -s -i $id "$s\r"
        incr cnt
    }
    puts "$cnt command lines sent"
}

# ---- Load parameter block ----
long_send $cmd_id "load 0 te 4 {
    parameterBlockId          = 0x00057014
    fepCcdSelect               = $ccd_list
    fepMode                    = 2 # FEP_TE_MODE_EV3x3
    bepPackingMode             = 2 # BEP_TE_MODE_GRADED
    onChip2x2Summing           = 0
    ignoreBadPixelMap          = 1
    ignoreBadColumnMap         = 1
    recomputeBias              = 0
    trickleBias                 = 0
    subarrayStartRow           = 0
    subarrayRowCount           = 99
    overclockPairsPerNode      = 2
    outputRegisterMode         = $quad_mode
    ccdVideoResponse           = 0 0 0 0 0 0
    primaryExposure             = 3
    secondaryExposure          = 0
    dutyCycle                   = 0
    fep0EventThreshold         = 100 100 100 100
    fep1EventThreshold         = 100 100 100 100
    fep2EventThreshold         = 100 100 100 100
    fep3EventThreshold         = 100 100 100 100
    fep4EventThreshold         = 100 100 100 100
    fep5EventThreshold         = 100 100 100 100
    fep0SplitThreshold         = 50 50 50 50
    fep1SplitThreshold         = 50 50 50 50
    fep2SplitThreshold         = 50 50 50 50
    fep3SplitThreshold         = 50 50 50 50
    fep5SplitThreshold         = 50 50 50 50
    fep4SplitThreshold         = 50 50 50 50
    fep5SplitThreshold         = 50 50 50 50
    lowerEventAmplitude        = 0
    eventAmplitudeRange        = 65535
    gradeSelections            = 0xffffffff 0xffffffff 0xffffffff 0xffffffff\
                                0xffffffff 0xffffffff 0xffffffff 0xffffffff
    windowSlotIndex           = 65535
    histogramCount              = 0
    biasCompressionSlotIndex   = 3 3 1 1 1 1
    rawCompressionSlotIndex    = 0
    ignoreInitialFrames        = 2
    biasAlgorithmId            = 1 1 1 1 1 1
    biasArg0                    = 1 1 1 1 1 1
    biasArg1                    = 0 0 0 0 0 0
    biasArg2                    = 0 0 0 0 0 0
    biasArg3                    = 26 26 50 50 50 50
    biasArg4                    = 20 20 20 20 20 20
    fep0VideoOffset            = 65 65 65 65
    fep1VideoOffset            = 65 65 65 65
    fep2VideoOffset            = 65 65 65 65
    fep3VideoOffset            = 65 65 65 65
    fep4VideoOffset            = 65 65 65 65
}
```

```
fep5VideoOffset          = 65 65 65 65
deaLoadOverride          = 0
fepLoadOverride          = 0
}
"
command_echo 1 9 "load te"

# ---- Start the Bias Run ----
send -i $cmd_id "start 0 te bias 4\r"
command_echo 1 15 "start bias run"
set timeout 600
wait_stop_science

# ---- Load the Event Image ----
system "make image"

# ---- Start the Science Run ----
send -i $cmd_id "start 0 te 4\r"
command_echo 1 14 "start science run"

set ncmd 0
set nloop 0
set nread 0
set expo 0
set expolast 0

# ---- Wait for Exposure Records ----
expect {
    -re "commandEcho.*commandOpcode=3\[\\r\\n]" {
        if {$nread > 0} {set nread [expr $nread - 1]}
        set expo $nread
        exp_continue
    }
    -re "exposureTeGraded.*fepId=(\[0-5\]).*exposureNumber=(\[0-9a-fx\]+).*\n" {
        set fep $expect_out(1,string)
        set expo [expr $expect_out(2,string)]
        if { $expo < $nexpo || $fep != $last_fep } {
            if {$nread <= 5} {
                set ncmd [expr ($ncmd + 1) % 65536]
                incr nread
                send -i $cmd_id "read $ncmd 0xa000e5e0 1\r"
            } elseif { $expo > $expolast + 2000 } {
                if {$nloop < 5} {
                    set nread 0
                    set expolast $expo
                    incr nloop
                } else {
                    fail "nexpo=$expo: no commandEcho since nexpo=$expolast"
                }
            }
        }
        exp_continue
    }
}
# ---- fall through to wait for scienceReport packet ----
}
timeout {
    fail "No exposure record"
}
}

send -i $cmd_id "stop 0 science\r"
command_echo 1 19 "stop science"

set timeout 600
science_report 1 "science report"
```

06/08/05
16:02:18

Flight S/W Patches, Revision C-C-D
../../tlmbusy/testsuite/bug-hw/runtest.tcl

4

```
# ---- Report fate ----  
pass "$expo exposures received, $nloop command resets"
```

```
#!/bin/env expect
#
# $Source: /nfs/acis/h3/acisfs/confignt1/patches/tlmbusy/testsuite/fix-hw/runtest.tcl,v $
#
# Test telemetry posting by concurrent tasks -- with tlmbusy patch
#

puts "Welcome to tlmbusy/testsuite/fix-hw/runtest.tcl"

# ---- Launch the command and telemetry server processes ----
set basedir [lindex $argv 0] ; # patch base directory
set tools [lindex $argv 1] ; # tool directory
set patchdir [lindex $argv 2] ; # patches under test
set quad_mode "0 \# QUAD_ABCD" ; # desired outputRegisterMode
set ccd_list "0 10 10 10 10 10" ; # desired fepCcdSelect
set last_fep 0 ; # last FEP read out
set nexpo 2000000 ; # limit to number of exposures
set server $env(ACISSERVER) ; # lrtcu/ctue server
set port $env(PORT) ; # lrtcu/ctue server command port

# ---- Embed procedure library ----
source $basedir/$tools/lib/lib-exp/runtest_support.tcl

# ---- Start command pipe ----
spawn /bin/sh -c "( bcmd | cclient $server $port ) > $server.err 2>&1"
set cmd_id $spawn_id
set send_slow {10 0.001}

# ---- Start telemetry pipe ----
spawn /bin/sh -c "filterClient -h $server | psci -m -u"
sleep 1

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcmd\
                 $basedir/$tools/share/opt_printswhouse.bcmd\
                 $basedir/$tools/share/opt_dearepl.bcmd\
                 $basedir/$patchdir/standard.bcmd\
                 $basedir/$patchdir/tlmbusy.bcmd"

warm_boot

# ---- Power on FEPs and CCDs ----
power_on_boards "$ccd_list"

# ---- While powering up, load the Bias Image ----
system "make bias"

# ---- Wait for FEPs to finish powering ----
expect {
    -re ".*SWSTAT_FEP_EXECMEM: $last_fep\[\\r\\n\]" {}
    timeout {}
}

# ---- Start DEA housekeeping ----
send -s -i $cmd_id "load 0 dea 4 {
    deaBlockId          = 0x00001014
    sampleRate          = 1
    queries = {
        ccdId           = 10
        queryId         = 0
    }
}
"
command_echo 1 13 "load dea"
```



```
send -i $cmd_id "start 0 dea 4\r"
command_echo 1 18 "start dea housekeeping"

# ---- Write long string line by line ----
proc long_send {id str} {
    set cnt 0
    foreach s [split $str "\n"] {
        send -s -i $id "$s\r"
        incr cnt
    }
    puts "$cnt command lines sent"
}

# ---- Load parameter block ----
long_send $cmd_id "load 0 te 4 {
    parameterBlockId          = 0x00057014
    fepCcdSelect               = $ccd_list
    fepMode                    = 2 # FEP_TE_MODE_EV3x3
    bepPackingMode             = 2 # BEP_TE_MODE_GRADED
    onChip2x2Summing           = 0
    ignoreBadPixelMap          = 1
    ignoreBadColumnMap         = 1
    recomputeBias              = 0
    trickleBias                 = 0
    subarrayStartRow           = 0
    subarrayRowCount            = 99
    overclockPairsPerNode      = 2
    outputRegisterMode         = $quad_mode
    ccdVideoResponse            = 0 0 0 0 0 0
    primaryExposure             = 3
    secondaryExposure           = 0
    dutyCycle                   = 0
    fep0EventThreshold          = 100 100 100 100
    fep1EventThreshold          = 100 100 100 100
    fep2EventThreshold          = 100 100 100 100
    fep3EventThreshold          = 100 100 100 100
    fep4EventThreshold          = 100 100 100 100
    fep5EventThreshold          = 100 100 100 100
    fep0SplitThreshold          = 50 50 50 50
    fep1SplitThreshold          = 50 50 50 50
    fep2SplitThreshold          = 50 50 50 50
    fep3SplitThreshold          = 50 50 50 50
    fep5SplitThreshold          = 50 50 50 50
    fep4SplitThreshold          = 50 50 50 50
    fep5SplitThreshold          = 50 50 50 50
    lowerEventAmplitude         = 0
    eventAmplitudeRange         = 65535
    gradeSelections             = 0xffffffff 0xffffffff 0xffffffff 0xffffffff\
                                0xffffffff 0xffffffff 0xffffffff 0xffffffff

    windowSlotIndex            = 65535
    histogramCount              = 0
    biasCompressionSlotIndex    = 3 3 1 1 1 1
    rawCompressionSlotIndex     = 0
    ignoreInitialFrames         = 2
    biasAlgorithmId             = 1 1 1 1 1 1
    biasArg0                    = 1 1 1 1 1 1
    biasArg1                    = 0 0 0 0 0 0
    biasArg2                    = 0 0 0 0 0 0
    biasArg3                    = 26 26 50 50 50 50
    biasArg4                    = 20 20 20 20 20 20
    fep0VideoOffset             = 65 65 65 65
    fep1VideoOffset             = 65 65 65 65
    fep2VideoOffset             = 65 65 65 65
    fep3VideoOffset             = 65 65 65 65
```

```
fep4VideoOffset      = 65 65 65 65
fep5VideoOffset      = 65 65 65 65
deaLoadOverride      = 0
fepLoadOverride      = 0
}
"
command_echo 1 9 "load te"

# ---- Start the Bias Run ----
send -i $cmd_id "start 0 te bias 4\r"
command_echo 1 15 "start bias run"
set timeout 600
wait_stop_science

# ---- Load the Event Image ----
system "make image"

# ---- Start the Science Run ----
send -i $cmd_id "start 0 te 4\r"
command_echo 1 14 "start science run"

set ncmd 0
set nloop 0
set nread 0
set expo 0
set expolast 0

# ---- Wait for Exposure Records ----
expect {
    -re "commandEcho.*commandOpcode=3\\[\r\n]" {
        if {$nread > 0} {set nread [expr $nread - 1]}
        set expolast $expo
        exp_continue
    }
    -re "exposureTeGraded.*fepId=(\[0-5\]).*exposureNumber=(\[0-9a-fx]+\).*\n" {
        set fep $expect_out(1,string)
        set expo [expr $expect_out(2,string)]
        if { $expo < $nexpo || $fep != $last_fep } {
            if {$nread <= 5} {
                set ncmd [expr ($ncmd + 1) % 65536]
                incr nread
                send -i $cmd_id "read $ncmd 0xa000e5e0 1\r"
            } elseif { $expo > $expolast + 2000 } {
                if {$nloop < 5} {
                    set nread 0
                    set expolast $expo
                    incr nloop
                } else {
                    fail "nexpo=$expo: no commandEcho since nexpo=$expolast"
                }
            }
        }
        exp_continue
    }
}
# ---- fall through to wait for scienceReport packet ----
}
timeout {
    fail "No exposure record"
}
}

send -i $cmd_id "stop 0 science\r"
command_echo 1 19 "stop science"

set timeout 600
```

```
science_report 1 "science report"
```

```
# ---- Report fate ----
```

```
pass "$expo exposures received, $nloop command resets"
```

```
#!/bin/env expect
#
# $Source: /nfs/acis/h3/acisfs/confignt1/patches/tlmbusy/testsuite/smoke/runtest.tcl,v $
#
# Simple 'smoke' test -- with tlmbusy patch
#

puts "Welcome to tlmbusy/testsuite/smoke/runtest.tcl"

# ---- Launch the command and telemetry server processes ----
set basedir [lindex $argv 0] ; # patch base directory
set tools [lindex $argv 1] ; # tool directory
set patchdir [lindex $argv 2] ; # patches under test
set quad_mode "0 \# QUAD_ABCD" ; # desired outputRegisterMode
set ccd_list "0 10 10 10 10 10" ; # desired fepCcdSelect
set last_fep 0 ; # last FEP read out
set nexpo 100 ; # limit to number of exposures

# ---- Embed procedure library ----
source $basedir/$tools/lib/lib-exp/runtest_support.tcl

# ---- Start command pipe ----
spawn $basedir/$tools/bin/cmdclient $env(ACISSERVER)
set cmd_id $spawn_id

# ---- Start telemetry pipe ----
spawn $basedir/$tools/bin/tlmclient $env(ACISSERVER)
sleep 1

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcml\
                 $basedir/$tools/share/opt_printshouse.bcml\
                 $basedir/$tools/share/opt_dearepl.bcml\
                 $basedir/$patchdir/standard.bcml"
warm_boot

# ---- Power on FEPs and CCDs ----
power_on_boards "$ccd_list"

# ---- While powering up, load the Bias Image ----
system "make bias"

# ---- Wait for FEPs to finish powering ----
expect {
    -re ".*SWSTAT_FEP_EXECMEM: $last_fep\[\r\n]" {}
    timeout {}
}

# ---- Start DEA housekeeping ----
send -i $cmd_id "load 0 dea 4 {
    deaBlockId          = 0x00001014
    sampleRate          = 1
    queries = {
        ccdId           = 10
        queryId         = 0
    }
}
"
command_echo 1 13 "load dea"
send -i $cmd_id "start 0 dea 4\r"
command_echo 1 18 "start dea housekeeping"

# ---- Load parameter block ----
```

```
send -i $cmd_id "load 0 te 4 {
  parameterBlockId      = 0x00057014
  fepCcdSelect          = $ccd_list
  fepMode               = 2 # FEP_TE_MODE_EV3x3
  bepPackingMode        = 2 # BEP_TE_MODE_GRADED
  onChip2x2Summing      = 0
  ignoreBadPixelMap     = 1
  ignoreBadColumnMap    = 1
  recomputeBias         = 0
  trickleBias           = 0
  subarrayStartRow      = 0
  subarrayRowCount      = 99
  overclockPairsPerNode = 2
  outputRegisterMode    = $quad_mode
  ccdVideoResponse      = 0 0 0 0 0 0
  primaryExposure        = 3
  secondaryExposure     = 0
  dutyCycle             = 0
  fep0EventThreshold    = 100 100 100 100
  fep1EventThreshold    = 100 100 100 100
  fep2EventThreshold    = 100 100 100 100
  fep3EventThreshold    = 100 100 100 100
  fep4EventThreshold    = 100 100 100 100
  fep5EventThreshold    = 100 100 100 100
  fep0SplitThreshold    = 50 50 50 50
  fep1SplitThreshold    = 50 50 50 50
  fep2SplitThreshold    = 50 50 50 50
  fep3SplitThreshold    = 50 50 50 50
  fep5SplitThreshold    = 50 50 50 50
  fep4SplitThreshold    = 50 50 50 50
  fep5SplitThreshold    = 50 50 50 50
  lowerEventAmplitude   = 0
  eventAmplitudeRange   = 65535
  gradeSelections       = 0xffffffff 0xffffffff 0xffffffff 0xffffffff\
                        0xffffffff 0xffffffff 0xffffffff 0xffffffff
  windowSlotIndex       = 65535
  histogramCount         = 0
  biasCompressionSlotIndex = 3 3 1 1 1 1
  rawCompressionSlotIndex = 0
  ignoreInitialFrames   = 2
  biasAlgorithmId        = 1 1 1 1 1 1
  biasArg0               = 1 1 1 1 1 1
  biasArg1               = 0 0 0 0 0 0
  biasArg2               = 0 0 0 0 0 0
  biasArg3               = 26 26 50 50 50 50
  biasArg4               = 20 20 20 20 20 20
  fep0VideoOffset        = 65 65 65 65
  fep1VideoOffset        = 65 65 65 65
  fep2VideoOffset        = 65 65 65 65
  fep3VideoOffset        = 65 65 65 65
  fep4VideoOffset        = 65 65 65 65
  fep5VideoOffset        = 65 65 65 65
  deaLoadOverride        = 0
  fepLoadOverride        = 0
}
"
command_echo 1 9 "load te"

# ---- Start the Bias Run ----
send -i $cmd_id "start 0 te bias 4\r"
command_echo 1 15 "start bias run"
set timeout 600
wait_stop_science
```

```
# ---- Load the Event Image ----
system "make image"

# ---- Start the Science Run ----
send -i $cmd_id "start 0 te 4\r"
command_echo 1 14 "start science run"

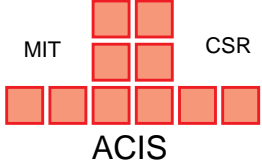
set ncmd 0
set nread 0
set expo 0

# ---- Wait for Exposure Records ----
expect {
    -re "commandEcho.*commandOpcode=3\[\r\n]" {
        if {$nread > 0} {set nread [expr $nread - 1]}
        exp_continue
    }
    -re "exposureTeGraded.*fepId=(\[0-5\]).*exposureNumber=(\[0-9a-fx]+\).*\n" {
        set fep $expect_out(1,string)
        set expo [ expr $expect_out(2,string) ]
        if { $expo < $nexpo || $fep != $last_fep } {
            if {$nread <= 5} {
                set ncmd [expr ($ncmd + 1) % 65536]
                incr nread
                send -i $cmd_id "read $ncmd 0xa000e5e0 1\r"
            }
            exp_continue
        }
    }
}
# ---- fall through to wait for scienceReport packet ----
}
timeout {
    fail "No exposure record"
}

send -i $cmd_id "stop 0 science\r"
command_echo 1 19 "stop science"

set timeout 600
science_report 1 "science report"

# ---- Report fate ----
pass "$expo exposures received"
```

		ENGINEERING CHANGE ORDER		<u>ECO No.</u> <u>36-1036</u>
CENTER FOR SPACE RESEARCH MASSACHUSETTS INSTITUTE OF TECHNOLOGY				
DWG. NO.	NEW REV.	DRAWING TITLE		
36-58021.03	D	Flight Software Patch Release C-C-D Certification		
REASON FOR CHANGE: Certification of standard patch release C, which includes the new <i>tlmbusy</i> and <i>buscrash</i> patches, along with the same optional patches that were certified in release B-C-C: <i>smtimedlookup</i> , <i>compressall</i> , <i>eventhist</i> , <i>cc3x3</i> , and <i>untricklebias</i> .				
DESCRIPTION OF CHANGE: Three optional patch combinations are certified as release C-C-D: (a) <i>cc3x3</i> , <i>eventhist</i> , and <i>smtimedlookup</i> . (b) <i>cc3x3</i> , <i>eventhist</i> , <i>compressall</i> , and <i>smtimedlookup</i> . (c) <i>cc3x3</i> , <i>eventhist</i> , <i>compressall</i> , <i>untricklebias</i> , and <i>smtimedlookup</i> . The certification tests are taken from the optional release C suite, the only difference being that the tests are conducted with release C of the standard patches.				
	SIGNATURE	DATE	REMARKS:	
ORIGINATOR	Peter Ford	08/09/07	Released version	
MECHANICAL				
ELECTRICAL				
SOFTWARE				
STRUCTURE				
FABRICATION				
SCIENCE				
SYSTEMS ENG.				
QUALITY				
PROJ. ENGINEER				
DEPUTY PM				
PROJ. MANAGER				

08/15/07
18:53:06

Flight S/W Patches, Revision C-C-D
../../certsrc/cc3x3+eventhist.notes

1

TITLE: ACIS eventhist, cc3x3, smtimedlookup Patch Certification Release Notes

DOCUMENT NUMBER: 36-58021.03 REVISION: D

ORIGINATOR: Peter G. Ford <pgf@space.mit.edu>

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
D	36-1036	Certify CC3x3/EventHist/smTimedL RFG		08/09/2007
D	36-1036	Rev. C Standard patches		

=====
Title: ACIS eventhist, cc3x3, smtimedlookup Patch Certification Release Notes for Version D

Software Change Order: 36-1036

Build Date: Wed Aug 15 18:53:05 EDT 2007
Part Number: 36-58021.03
Version: D
CVS Tag: cc3x3+eventhist-C-C-D

Std Number: 36-58010
Std Version: C
Std Tag: release-C
Std SCO: 36-1035

Opt Number: 36-58020
Opt Version: C
Opt Tag: release-C-opt-C
Opt SCO: 36-1035

IPCL Number: 36-53204.0204
IPCL Version: N
IPCL CVS Tag: release-N

Description:

This certification verifies the operation of Continuous Clocking 3x3 Patch in conjunction with the Event Histogram and smTimedLookup Patches.

The certification consists of three tests, copied from the original test runs during the Options Release. The tests have been modified to load all three optional patches, rather than just one of them, and to clean up some false failures due to timing/pattern matching issues in the tests.

The tests verify that the patch modes run as they did during the original test when they are both installed into the system.

The Continuous Clocking 3x3 (cc3x3) test consists of two parts. The first launches a CC3x3 run, whereas the second runs CClx3. This suite performs CClx3 tests to verify that the modifications to the existing BEP Continuous Clocking functions do not break the existing CClx3 functionality. Since the FEP software only contains CC3x3 code during CC3x3 runs (this is verified by the CClx3 run), and no BEP functions used by Timed Exposure are modified by the patch, the Timed Exposure modes do not need to be re-tested as part of this certification.

Each test sends a series of events on the right side of each quadrant (the original test was derived from the test for the rquad bug fix), and verifies that the mode runs nominally, and produces the expected event list. Since the "stop" critereon for the test is a little fuzzy, the runs tend to produce additional exposures that aren't in the file used to check the run's event output. "diff" used in the test produces mismatches on the additional exposures produced by the test run. Manual check of the run data shows that the event lists are replicated correctly by the run. Later, a "wrapping"

comparison may be developed to eliminate this manual step.

The Event Histogram test uses a similar strategy to the CC3x3 test. It starts an Event Histogram run, and sends in a series of standard events. It then compares the resulting quadrant histograms with an example file to verify the results.

One caveat that arose during the review of the Optional patches is that, when the standard patch "zaplexpo" is present, which it should always be, the first exposure of event histogram mode will not contain any events. This will cause the first histogram from each FEP quadrant to appear to have been integrated for 1 less frame time than subsequent quadrant histograms. This is different than Raw Histogram mode, which is not affected by the "zaplexpo" patch. The histogram example file used for this certification assumes that no events are sent during exposure 2 (the first "real" exposure of the run).

The smTimedExposure patch is tested by merely running a timed-exposure faint run, verifying that the bias and event detection phases have been invoked, and then stopping the run.

Included Patches:

eventhist
cc3x3
smtimedlookup

Test Support Patches:

printswhouse
dearepl
tlmio

Test Results:

smtimedlookup --> PASS
cc3x3 --> PASS
eventhist --> PASS
eventhist --> PASS

08/15/07
20:44:57

Flight S/W Patches, Revision C-C-D
../../certsrc/cc3x3+eventhist+compressall.notes

1

TITLE: ACIS eventhist, cc3x3, compressall, smtimedlookup Patch Certification Release Notes

DOCUMENT NUMBER: 36-58021.03 REVISION: D

ORIGINATOR: Peter G. Ford <pgf@space.mit.edu>

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
D	36-1036	Certify CC3x3/EventHist/smTimedL RFG		08/09/2007
D	36-1036	Rev. C standard patches		

=====
Title: ACIS eventhist, cc3x3, compressall, smt timedlookup Patch Certification Release Notes for Version D

Software Change Order: 36-1036

Build Date: Wed Aug 15 20:44:56 EDT 2007
Part Number: 36-58021.03
Version: D
CVS Tag: cc3x3+eventhist+compressall-C-C-D

Std Number: 36-58010
Std Version: C
Std Tag: release-C
Std SCO: 36-1035

Opt Number: 36-58020
Opt Version: C
Opt Tag: release-C-opt-C
Opt SCO: 36-1035

IPCL Number: 36-53204.0204
IPCL Version: N
IPCL CVS Tag: release-N

Description:

This certification verifies the operation of the Continuous Clocking 3x3, Event Histogram, Compress All, and Science Mode Timed Lookup Patches.

The certification consists of two tests, copied from the original test run during the Options Release. The tests have been modified to load all four optional patches, rather than just one at a time, and to clean up some false failures due to timing/pattern matching issues in the tests.

The tests verify that the patch modes run as they did during the original test when they are both installed into the system.

The Continuous Clocking 3x3 (cc3x3) test consists of two parts. The first launches a CC3x3 run, whereas the second runs CCLx3. This suite performs CCLx3 tests to verify that the modifications to the existing BEP Continuous Clocking functions do not break the existing CCLx3 functionality. Since the FEP software only contains CC3x3 code during CC3x3 runs (this is verified by the CCLx3 run), and no BEP functions used by Timed Exposure are modified by the patch, the Timed Exposure modes do not need to be re-tested as part of this certification.

Each test sends a series of events on the right side of each quadrant (the original test was derived from the test for the rquad bug fix), and verifies that the mode runs nominally, and produces the expected event list. Since the "stop" critereon for the test is a little fuzzy, the runs tend to produce additional exposures that aren't in the file used to check the run's event output. "diff" used in the test produces mismatches on the additional exposures produced by the test run. Manual check of the run data shows that the event lists are replicated correctly by the run. Later, a "wrapping"

comparison may be developed to eliminate this manual step.

The Event Histogram test uses a similar strategy to the CC3x3 test. It starts an Event Histogram run, and sends in a series of standard events. It then compares the resulting quadrant histograms with an example file to verify the results.

One caveat that arose during the review of the Optional patches is that, when the standard patch "zaplexpo" is present, which it should always be, the first exposure of event histogram mode will not contain any events. This will cause the first histogram from each FEP quadrant to appear to have been integrated for 1 less frame time than subsequent quadrant histograms. This is different than Raw Histogram mode, which is not affected by the "zaplexpo" patch. The histogram example file used for this certification assumes that no events are sent during exposure 2 (the first "real" exposure of the run).

The smTimedExposure patch is tested by merely running a timed-exposure faint run, verifying that the bias and event detection phases have been invoked, and then stopping the run.

The Compress All patch is tested by copying an image to the image loader that contains several very "noisy" rows that are known to be incompressible by the Huffman tables. A timed-exposure raw-mode run is executed and the pixelCount field of the dataTeRaw packets of a couple of raw frames is monitored. The test fails if pixelCount is ever zero.

Included Patches:
eventhist
cc3x3
compressall
smtimedlookup

Test Support Patches:
printswhouse
dearepl
tlmio

Test Results:
smtimedlookup --> PASS
cc3x3 --> PASS
eventhist --> PASS
eventhist --> PASS
compressall --> PASS

08/16/07
01:12:08

Flight S/W Patches, Revision C-C-D

1

.././certsrc/cc3x3+eventhist+compressall+untricklebias.notes

TITLE: ACIS untricklebias, eventhist, cc3x3, compressall, smtimedlookup Patch Certification Release Notes

DOCUMENT NUMBER: 36-58021.03 REVISION: D

ORIGINATOR: Peter G. Ford <pgf@space.mit.edu>

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
D	36-1036	Certify CC3x3/EventHist/smTimedL	RFG	08/09/2007
D	36-1036	Rev. C standard patches		

=====
Title: ACIS untricklebias, eventhist, cc3x3, compressall, smtmedlookup Patch Certification
Release Notes for Version D

Software Change Order: 36-1036

Build Date: Thu Aug 16 01:12:08 EDT 2007
Part Number: 36-58021.03
Version: D
CVS Tag: cc3x3+eventhist+compressall+untricklebias-C-C-D

Std Number: 36-58010
Std Version: C
Std Tag: release-C
Std SCO: 36-1035

Opt Number: 36-58020
Opt Version: C
Opt Tag: release-C-opt-C
Opt SCO: 36-1035

IPCL Number: 36-53204.0204
IPCL Version: N
IPCL CVS Tag: release-N

Description:

This certification verifies the operation of the Continuous Clocking 3x3, Event Histogram, Compress All, Untrickle Bias, and Science Mode Timed Lookup Patches.

The certification consists of two tests, copied from the original test run during the Options Release. The tests have been modified to load all four optional patches, rather than just one at a time, and to clean up some false failures due to timing/pattern matching issues in the tests.

The tests verify that the patch modes run as they did during the original test when they are both installed into the system.

The Continuous Clocking 3x3 (cc3x3) test consists of two parts. The first launches a CC3x3 run, whereas the second runs CClx3. This suite performs CClx3 tests to verify that the modifications to the existing BEP Continuous Clocking functions do not break the existing CClx3 functionality. Since the FEP software only contains CC3x3 code during CC3x3 runs (this is verified by the CClx3 run), and no BEP functions used by Timed Exposure are modified by the patch, the Timed Exposure modes do not need to be re-tested as part of this certification.

Each test sends a series of events on the right side of each quadrant (the original test was derived from the test for the rquad bug fix), and verifies that the mode runs nominally, and produces the expected event list. Since the "stop" critereon for the test is a little fuzzy, the runs tend to produce additional exposures that aren't in the file used to check the run's event output. "diff" used in the test produces mismatches on the additional exposures produced by the test run. Manual check of the run data shows that the event lists are replicated correctly by the run. Later, a "wrapping"

comparison may be developed to eliminate this manual step.

The Event Histogram test uses a similar strategy to the CC3x3 test. It starts an Event Histogram run, and sends in a series of standard events. It then compares the resulting quadrant histograms with an example file to verify the results.

One caveat that arose during the review of the Optional patches is that, when the standard patch "zaplexpo" is present, which it should always be, the first exposure of event histogram mode will not contain any events. This will cause the first histogram from each FEP quadrant to appear to have been integrated for 1 less frame time than subsequent quadrant histograms. This is different than Raw Histogram mode, which is not affected by the "zaplexpo" patch. The histogram example file used for this certification assumes that no events are sent during exposure 2 (the first "real" exposure of the run).

The smTimedExposure patch is tested by merely running a timed-exposure faint run, verifying that the bias and event detection phases have been invoked, and then stopping the run.

The Compress All patch is tested by copying an image to the image loader that contains several very "noisy" rows that are known to be incompressible by the Huffman tables. A timed-exposure raw-mode run is executed and the pixelCount field of the dataTeRaw packets of a couple of raw frames is monitored. The test fails if pixelCount is ever zero.

The Untrickle Bias patch is tested by a pair of expect scripts, each of which performs 12 tests, one in TE mode, the other in CC mode. Each test starts a science run and then terminates it in one of the possible ways, viz:

- 1: stopScience during bias map creation
- 2: double stopScience during bias map creation
- 3: startScience during bias map creation
- 4: assert/deassert RADMON during bias map creation
- 5: stopScience during bias map telemetering
- 6: double stopScience during bias map telemetering
- 7: startScience during bias map telemetering
- 8: assert/deassert RADMON during bias map telemetering
- 9: stopScience during event processing
- 10: double stopScience during event processing
- 11: startScience during event processing
- 12: assert/deassert RADMON during event processing

The tests fail unless all steps complete and return the anticipated scienceReport return codes.

Included Patches:

untricklebias
eventhist
cc3x3
compressall
smtimedlookup

Test Support Patches:

printswhouse
dearepl
tlmio

Test Results:

smtimedlookup --> PASS
cc3x3 --> PASS
eventhist --> PASS
eventhist --> PASS
compressall --> PASS
untricklebias --> PASS
untricklebias --> PASS