ENGINEERING CHANGE ORDER

ECO No. 36-1053

KAVLI INSTITUTE FOR ASTROPHYSICS AND SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DRAWING NO.	REVISION	DRAWING TITLE
36-58010	H	Flight Software Standard Patch Release G and Optional Release H

REASON FOR CHANGE:

The standard *buscrash* patch has been updated to force a science run to end when it is commanded to do so, *e.g.*, by an SCS107 from the OBC, while it is waiting for the *biasthief* task to complete. This fixes a bug that was introduced by revision C of the *buscrash2* patch.

The optional *deahktrip* monitors DPA component temperatures and, if an anomalous value occurs, optionally stops a science run, powers down FEP and video boards, & sets S/W bilevels.

DESCRIPTION OF CHANGE:

In the updated *buscrash*, code is added to the `FepManager::pollBiasComplete()` to check whether any FEPs are still being has been accessed while the *biasthief* task is commanded to abort. If not, the task monitor is told to halt the science run.

In *deahktrip*, code is added to `Tf_Dea_Housekeeping_Data::append_Entries()` to monitor DEA H/K channels for bad DPA component temperatures and optionally terminate a science manager task, power down boards, and/or alter S/W bilevels to ‘1110’ (14) to signal to the OBC.

	SIGNATURE	DATE	REMARKS
ORIGINATOR	PGF	06/29/18	Signature on file
MECHANICAL			
ELECTRICAL	DA	06/29/18	Signature on file
SOFTWARE	JEF	06/29/18	Signature on file
STRUCTURE			
FABRICATION			
SCIENCE			
SYSTEMS ENG.			
QUALITY	RB	06/29/18	Signature on file
PROJ. ENGINEER	RFG	06/29/18	Signature on file
DEPUTY PM			
PROJ. MANAGER			
APM RELEASE			

Existing ACIS Flight Software Patches

ID	Name	Rev	Size	Part	ECO	SPR
Standard Release G						
i	corruptblock	A	16	36–58030.01	994	113
ii	digestbiaserror	A	64	36–58030.02	995	116
iii	histogramvar	A	16	36–58030.03	999	115
iv	rquad	A	16	36–58030.14	1000	121
v	histogrammean	A	156	36–58030.15	996	123
vi	zap1expo	A	64	36–58030.16	997	122
vii	condoclk	A	640	36–58030.17	1012	127
viii	fepbiasparity2	A	504	36–58030.19	1015	130
ix	cornermean	A	32	36–58030.21	1017	128
x	tlmbusy	A	344	36–58030.29	1033	138
xi	buscrash	B	440	36–58030.30	1051	140,151
xii	badpix	A	60	36–58030.31	1037	141
xiii	buscrash2	C	1576	36–58030.32	1047	148,150
Optional Release H						
1	smtimedlookup	A	3712	36–58030.24	1025	N/A
2	eventhist	B	5908	36–58030.05	1025	N/A
3	cc3x3	B	4636	36–58030.06	1018	120,124,126
4	ctireport1	A	5452	36–58030.25	1026	N/A
5	ctireport2	A	2784	36–58030.26	1026	N/A
6	compressall	A	2368	36–58030.27	1027	134
7	reportgrade1	A	816	36–58030.22	1021	131,132
8	txings	A	3128	36–58030.33	1044	N/A
9	deahktrip	A	1940	36–58030.34	1052	N/A
leaf	teignore	A	36	36-58030.09	1003	N/A
leaf	ccignore	A	36	36-58030.10	1004	N/A
Under Development						
12	fepbiasparity1	2	N/A	36–58030.18	1014	N/A
13	hybrid	3	6104	36–58030.13	1010	N/A
14	squeegy	6	4412	36-58030.23	1023	N/A
15	forcebiastrickle	1	N/A	36-58030.29	1024	133
Engineering Unit Utility Patches						
10	tlmio	2	10312	36–58030.07	1010	N/A
11	printswhouse	1	7240	36–58030.08	986	N/A
leaf	deaeng	2	2604	36–58030.11	1010	N/A
leaf	dearepl	2	556	36–58030.12	1010	N/A

Name	Part Number	Description	Typos[a]	RIDs[b]	Status
buscrash	36–58030.30 (ECO 36–1051)	Prevent Trickle-Bias anomalies and BEP crashes	1	0	Passed RFG 06/29/2018
deahktrip	36–58030.34 (ECO 36–1052)	React to anomalous DPA component temperatures	6	0	Passed RFG 06/29/2018
S/W Review	36–58010 (ECO 36–1053)	Documentation accompanying the individual patch ECOs	0	0	Passed RFG 06/29/2018
Certification	36–58021.04 (ECO 36–1054)	Documentation describing the multi-patch certification tests	0	0	Passed RFG 06/29/2018

[a] typographical errors in the documentation

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

MIT CSR ACIS

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 154

MIT CSR ACIS

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	x3-6485	03/08/16		

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

SCS107 commands failed to terminate a science run during bias creation

On March 3rd 2016 the observatory entered Normal Sun Mode as a result of a pointing maneuver exceeding its expected limits. The on-board computer executed an SCS107 command sequence which sent the usual safing commands to ACIS, which was creating a bias map for CCD_S3 in OBSID 17719. During recovery, the ACIS team noticed that the `1STAT1ST` flag in the ACIS bilevel channel was zero, indicating that the BEP's science thread was still active.

The anomaly was quickly reproduced on the ACIS Engineering Unit and it was found that the simplest way of returning the flight instrument to `SCIENCE_IDLE` mode would be to warm-boot the BEP, which was done without further incident.

Corrective Action:

The anomaly was traced to the combined behavior of several BEP methods belonging to the `FepManager` classes. `pollOperationComplete`() first calls `isBusy()` to confirm that a FEP is powered up, and then `queryFepStatus()` to check whether its bias map is complete. If no FEPs are still busy creating bias maps, `pollOperationComplete()` returns `BoolTrue`. The code doesn't distinguish a scenario in which all maps are available from one in which all FEPs are powered down.

If all FEPs are powered off, the science thread receives `BoolTrue` from `ScienceMode::waitForBias`(), which it interprets as the go-ahead for bias trickling (if requested), so it calls `waitForBiasTrickle`() to wait for the bias thief thread to copy the bias maps to telemetry. However, when the *buscrash2* patch is loaded, its `Test_BiasThief::goTaskEntry`() method will loop indefinitely without setting `busyFlag` to `BoolFalse` to tell `waitForBiasTrickle`() that the bias maps have been trickled.

The *buscrash* patch has been updated to test that at least one FEP is powered up before trickling the bias. If not, it tells the task manager to invoke a `EV_SM_ABORT_RUN`, which ends science processing.

Problem closed on:	Date:	Refer to ECO #:	Refer to Patch ID:
		36-1051	buscrash

Problem ID: **M16030301**	Status: Open	Sheet: 151 of 154

MIT CSR ACIS	**ENGINEERING CHANGE ORDER**	**ECO No.** **36-1051**

KAVLI INSTITUTE FOR ASTROPHYSICS AND SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DRAWING NO.	REVISION	DRAWING TITLE
36-58030.30	B	*buscrash* patch to force `SCIENCE_IDLE` when all FEPs powered off

REASON FOR CHANGE:

During OBSID 17719 on March 3rd 2016 the Chandra OBC commanded ACIS to power down all FEPs during bias map creation. During recovery, the ACIS team noticed that ACIS `1STAT1ST` bilevel bit was zero, indicating that the BEP's science thread was still active. The anomaly was traced to the combined behavior of several BEP methods belonging to the `FepManager` class.

DESCRIPTION OF CHANGE:

In `ScienceMode::computeBias()`, `FepManager` method `pollOperationComplete()` calls `isBusy()` to confirm that a FEP is powered up, and then `queryFepStatus()` to check whether its bias map is complete. If no FEPs are still busy creating bias maps, `pollOperationComplete()` returns `BoolTrue`. The code doesn't distinguish a scenario in which all maps are available from one in which all FEPs are powered down. In both cases, the science thread receives `BoolTrue` from `waitForBias()`, so it calls `waitForBiasTrickle()` to wait for the bias thief task to copy bias maps to telemetry. However, when the *buscrash2* patch is loaded, its `Test_BiasThief::goTaskEntry()` method will loop indefinitely without setting `busyFlag` to `BoolFalse` to tell `waitForBiasTrickle()` that the bias maps have been trickled. The *buscrash* patch has been updated to test that at least one FEP is powered up before trickling the bias. If not, it tells the task manager to invoke a `EV_SM_ABORT_RUN`, which ends science processing.

	SIGNATURE	DATE	REMARKS
ORIGINATOR	PGF	06/29/18	Signature on file
MECHANICAL			
ELECTRICAL	DA	06/29/18	Signature on file
SOFTWARE	JEF	06/29/18	Signature on file
STRUCTURE			
FABRICATION			
SCIENCE			
SYSTEMS ENG.			
QUALITY	RB	06/29/18	Signature on file
PROJ. ENGINEER	RFG	06/29/18	Signature on file
DEPUTY PM			
PROJ. MANAGER			
APM RELEASE			

1. Reasons for the Patch

On March 3rd 2016 the observatory entered Normal Sun Mode as a result of a pointing maneuver exceeding its expected limits. The on-board computer executed an SCS107 command sequence which sent the usual safing commands to ACIS, which was creating a bias map for `CCD_S3` in OBSID 17719. During recovery, the ACIS team noticed that the `1STAT1ST` flag in the ACIS bilevel channel was zero, indicating that the BEP´ s science thread was still active.

The anomaly was quickly reproduced on the ACIS Engineering Unit and it was found that the simplest way of returning the flight instrument to `SCIENCE_IDLE` mode would be to warm-boot the BEP, which was done without further incident.

The anomaly was traced to the combined behavior of several BEP methods belonging to the `FepManager` classes. `pollOperationComplete`() first calls `isBusy`() to confirm that a FEP is powered up, and then `queryFepStatus`() to check whether its bias map is complete. If no FEPs are still busy creating bias maps, `pollOperationComplete`() returns `BoolTrue`. The code doesn't distinguish a scenario in which all maps are available from one in which all FEPs are powered down.

If all FEPs are powered off, the science thread receives `BoolTrue` from `ScienceMode`::`waitForBias`(), which it interprets as the go-ahead for bias trickling (if requested), so it calls `waitForBiasTrickle`() to wait for the bias thief thread to copy the bias maps to telemetry. However, when the *buscrash2* patch is loaded, its `Test_BiasThief`::`goTaskEntry`() method will loop indefinitely without setting `busyFlag` to `BoolFalse` to tell `waitForBiasTrickle`() that the bias maps have been trickled.

The *buscrash* patch has been updated to test that at least one FEP is powered up before trickling the bias. If not, it tells the task manager to invoke a `EV_SM_ABORT_RUN`, which ends science processing.

2. Description of the Original Patch

The purpose of the original patch was to prevent the BEP from crashing when one or more FEPs were powered down while bias maps were being created. While the existing flight code correctly determined when the maps were ready, it went on to call `FepManager`::`loadBadPixel`() to update the maps with the known locations of 'bad' pixels and columns. If that routine was called for a FEP that wasn't powered up, a bus error would result and the BEP would crash.

```
void FepManager::loadBadPixel(FepId fepid, unsigned row, unsigned col)
{
    DebugProbe probe;

    fepIo[fepid]->writeBiasValue(row, col, PIXEL_BAD);
}
```

The patch replaced loadBadPixel() with the following code that tests whether the FEP is powered up:

```
void Test_FepManager::loadBadPixel(FepId fepid, unsigned row, unsigned col)
{
    DebugProbe probe;

    if (fepManager.isEnabled(fepid) == BoolTrue) {
        fepIo[fepid]->writeBiasValue(row, col, PIXEL_BAD);
    }
}
```

3. Update to the *buscrash* Patch

The new *buscrash* patch replaces the `FepManager` class method `pollBiasComplete`() which was originally

```
Boolean FepManager::pollBiasComplete()
{
    DebugProbe probe;
    Boolean retval = BoolFalse;
    retval = pollOperationComplete();
    return retval;
}
```

with the following code:

```
Boolean FepManager::pollBiasComplete()
{
    DebugProbe probe;
    Boolean retval = BoolFalse;
    retval = pollOperationComplete();
    if (retval == BoolTrue && fepManager.anyEnabled() == BoolFalse) {
        Task * curTask = taskManager.queryCurrentTask();
        if (curTask != 0) {
            curTask->notify(ScienceMode::EV_SM_ABORT_RUN);
            retval = BoolFalse;
        }
    }
    return retval;
}
```

which waits until *either* the bias maps are ready *or* all FEPs are powered off, and, in the latter case, passes the `EV_SM_ABORT_RUN` signal to the task manager, which ends the science run, possibly truncating or eliminating bias maps. Since the trickle-bias task is never activated, no change is necessary in the *buscrash2* patch.

4. Controlled Sources

buscrash	
Makefile	Generate a stand-alone *buscrash.bcmd* file
buscrash.C	Source code for the `Test_BiasThief` class
buscrash.mak	Makefile script to generate flight patch
buscrash.pkg	Script to describe patch release
eco-1051.doc	Engineering change order describing the *buscrash2* patch
spr151.pdf	Originating software problem report
buscrash/testsuite	
makebias	Generate a timed exposure bias image
buscrash/testsuite/bug-hw	
Makefile	Run a test to demonstrate BEP bugs
runtest.tcl	*expect* script to demonstrate a BEP bus crash without the *buscrash* patch
runtest2.tcl	*expect* script to demonstrate anomaly with *buscrash2* loaded and FEP power-down
buscrash/testsuite/fix-hw	
Makefile	Run a test with the *buscrash* patch
buscrash.bcmd	Stand-alone *buscrash* patch with modifications to detect all FEPs powered down
runtest.tcl	*expect* script to demonstrate correct power-down behavior in timed exposure mode

5. Testing

All tests are performed on the ACIS Engineering Unit using one FEPs and the L-RCTU interface. All tests also use the image loader. 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.

5.1. Test to reproduce a BEP bus crash

An *expect* procedure, "*bug-hw/runtest.tcl*", performs a timed-exposure science run with the optional *tlmio*, *printswhouse*, and *dearepl* patches. The following steps are performed:

1. A command pipe is spawned, through which ACIS commands will be sent to the EU.
2. A telemetry pipe is spawned, terminating in the "*psci -m -u*" packet-monitoring filter, with *expect* examining the standard output.
3. ACIS is cold-booted.
4. Software housekeeping and DEA replacement 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 parameter block is sent to ACIS, calling for FEP_0 to be run in timed-exposure graded mode, with 3.3 second full-frame exposures.
9. A science run is started. Its telemetry is monitored by the *expect* script.
10. Once a `FEP_STARTBIAS` housekeeping pseudopacket is received, the script waits for 10 seconds and then simulates an SCS107 command sequence by executing two *stopScience* commands at 2-second intervals, followed by a command to power down all FEPs and DEAs.
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.

5.2. Test to reproduce failure to terminate science mode

An *expect* procedure, "*bug-hw/runtest2.tcl*", performs a timed-exposure science run with the *standard* patches from release F and the optional *tlmio*, *printswhouse*, and *dearepl* patches. The following steps are performed:

1. A command pipe is spawned, through which ACIS commands will be sent to the EU.
2. A telemetry pipe is spawned, terminating in the "*psci -m -u*" packet-monitoring filter, with *expect* examining the standard output.
3. ACIS is cold-booted.
4. Standard level F, software housekeeping and DEA replacement 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 parameter block is sent to ACIS, calling for FEP_0 to be run in timed-exposure graded mode, with 3.3 second full-frame exposures.
9. A science run is started. Its telemetry is monitored by the *expect* script.
10. Once a `FEP_STARTBIAS` housekeeping pseudopacket is received, the script waits for 10 seconds and then simulates an SCS107 command sequence by executing two *stopScience* commands at 2-second intervals, followed by a command to power down all FEPs and DEAs.

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 1 minute.
12. The test is passed if case (3) occurs; otherwise, the test fails.

5.3. Test to verify correct behavior of patch

An *expect* procedure, "fix-*hw/runtest2.tcl*", performs a timed-exposure science run with the standalone *buscrash* patch and the optional *tlmio*, *printswhouse*, and *dearepl* patches. The following steps are performed:

1. A command pipe is spawned, through which ACIS commands will be sent to the EU.
2. A telemetry pipe is spawned, terminating in the "*psci -m -u*" packet-monitoring filter, with *expect* examining the standard output.
3. ACIS is cold-booted.
4. Standalone *buscrash*, software housekeeping and DEA replacement 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 parameter block is sent to ACIS, calling for FEP_0 to be run in timed-exposure graded mode, with 3.3 second full-frame exposures.
9. A science run is started. Its telemetry is monitored by the *expect* script.
10. Once a `FEP_STARTBIAS` housekeeping pseudopacket is received, the script waits for 10 seconds and then simulates an SCS107 command sequence by executing two *stopScience* commands at 2-second intervals, followed by a command to power down all FEPs and DEAs.
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 (2) occurs; otherwise, the test fails.

Appendices

A. Example of *expect* script

This example shows that part of the "*bug-hw/runtest.tcl*" script after patches have been loaded and the BEP warm-booted. The other scripts are identical except for the pass/fail criteria in the second `expect` statement.

```
# ---- Load Pblock for Faint Timed-Exposure Mode ----
send -i $cmd_id "load 0 te 4 {
. . .
}
”
command_echo 1 9 "load te”

# ---- Copy bias map into Image Loader ----
system make bias

# ---- Start the run ----
puts "\n# Starting test\n"
send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 "start science run"

# ---- Wait for bias calculation to start ----
expect {
    -timeout 360
    -re "SWSTAT_FEP_STARTBIAS.*\[\r\n]*" { }
    timeout { fail "Bias Failure" }
}
sleep 10

# ---- Mimic an SCS107: stop science ----
puts "# stopScience"
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"
sleep 2

# ---- Repeat the stop science ----
puts "# stopScience"
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"
sleep 2

# ---- Power off all FEPs and video boards ----
puts "# powering boards off"
power_off_boards

# ---- Inspect the result ----
expect {
    -timeout 360
    -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"
    }
}

# ---- Don’t come here ----
```

B. Glossary

BEP	ACIS Back-End Processor — a component of the DPA
BiasThief	BEP task (processing thread) to read FEP bias maps and write them to telemetry
bug-hw	Directory containing tests designed to reproduce an ACIS hardware error
CCD	Charge Coupled Device
DEA	ACIS Detector Electronics Assembly comprising analog and interface boards
dearepl	Patch to BEP software to initialize non-flight design DEA video boards
EU	ACIS Engineering Unit — hardware simulator of the DEA and DPA
expect	Interactive input/output scripting language based on TCL
FEP	ACIS Front-End Processor — a component of the DPA
`fepId`	BEP software variable denoting a FEP — 0 through 5
fix-hw	Directory containing tests designed to eliminate an ACIS hardware error
L-RCTU	Jim Littlefield's Remote Command and Telemetry Unit, interface to the DPA
OBSID	Chandra observation ID
SCS107	Stored command sequence to protect the Chandra payload when entering safe mode
TCL	Tool Command Language, a tiresome scripting language best avoided whenever possible

MIT CSR ACIS	**ENGINEERING CHANGE ORDER**	**ECO No.** **36-1052**

KAVLI INSTITUTE FOR ASTROPHYSICS AND SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DRAWING NO.	REVISION	DRAWING TITLE
36-58030.34	A	*deahktrip* patch to respond to anomalous DPA boar d temperatures

REASON FOR CHANGE:

Chandra's thermal protection is aging. When the Sun illuminates the spacecraft at unfavorable pitch angles, the temperatures of several critical components in the ACIS digital processor assembly (DPA) are approaching their yellow alarm limits. Reducing the permitted range of pitch angles would improve the situation, but at the expense of losing valuable science results. As an alternative, the *deahktrip* patch will monitor the 12 DPA component temperatures and, should any exceed its limits, optionally halt the observation, power down the video and front-end processor (FEP) boards, and/or set the ACIS bilevel values to alert the spacecraft's on-board computer (OBC) to place ACIS in a safe mode.

DESCRIPTION OF CHANGE:

The ACIS back-end processor's (BEP) flight software monitors DPA voltages and temperatures from its DEA housekeeping task, calling `Tf_Dea_Housekeeping_Data::append_Entries`() to pack the `ccdId`, `channelId` and data values into telemetry packets for downlink. The *deahktrip* patch replaces this routine with one that performs these same operations with additional code to compare the component temperatures against a table of temperature limits. If any exceeds its limit, a science run in progress can be halted and video and FEP boards powered down; also the 4-bit ACIS software bilevel channels can be set to `LED_BOOT_SPARE2` (*i.e.*, '1110'b or 14_{10}) to instruct the OBC to command ACIS to terminate the observation and to power down the boards.

	SIGNATURE	DATE	REMARKS
ORIGINATOR	PGF	06/29/18	Signature on file
MECHANICAL			
ELECTRICAL	DA	06/29/18	Signature on file
SOFTWARE	JEF	06/29/18	Signature on file
STRUCTURE			
FABRICATION			
SCIENCE			
SYSTEMS ENG.			
QUALITY	RB	06/29/18	Signature on file
PROJ. ENGINEER	RFG	06/29/18	Signature on file
DEPUTY PM			
PROJ. MANAGER			
APM RELEASE			

1. Reasons for the Patch

Chandra's thermal protection is aging. When the Sun illuminates the spacecraft at unfavorable pitch angles, the temperatures of several critical components in the ACIS digital processor assembly (DPA) are approaching their yellow alarm limits. Reducing the permitted range of pitch angles would improve the situation, but at the expense of losing valuable science results. As an alternative, the *deahktrip* patch will monitor the 12 DPA component temperatures and, should any exceed its limits, optionally halt the observation, power down the video and front-end processor (FEP) boards, and/or set the ACIS bilevel values to alert the spacecraft's on-board computer (OBC) to place ACIS in safe mode. Within the BEP, these temperatures are measured by thermistors in series with resistors, read by an A/D converter in unipolar mode. The relationship between raw 12-bit DN values and temperatures in centigrade is as follows:

$$\text{T (in C)} = (1.074\times10^{-7}Q^{3} + 2.372\times10^{-4}Q + 1.4733\times10^{-3})^{-1} - 273.16$$
$$\text{where } Q = \log_{e}(5230 \times R/(2048 - R)) \text{ and } R = 1.14 \times (DN - 2048)$$

All raw (DN) values below 2060 are considered "Hot" and those above 3880 "Cold". The following table lists the current yellow and red alert values for the 12 temperatures [see Appendix D, item 5].

	Channel Mnemonic	Channel Description	low alert limits		high alert limits	
			red	yellow	yellow	red
1	BEP_PCB	BEP-A PC Board	3313 (-10 C)	2953 (6.5 C)	2342 (44.0 C)	2297 (49.0 C)
2	BEP_OSC	BEP-A Oscillator	3313 (-10 C)	2953 (6.5 C)	2362 (42.0 C)	2314 (47.0 C)
3	FEP0_MONG	FEP_0 Mongoose	3313 (-10 C)	3098 (0.0 C)	2306 (48.0 C)	2266 (53.1 C)
4	FEP0_PCB	FEP_0 PC Board	3313 (-10 C)	3098 (0.0 C)	2332 (45.0 C)	2289 (50.0 C)
5	FEP0_ACTEL	FEP_0 ACTEL	3313 (-10 C)	3098 (0.0 C)	2314 (47.0 C)	2274 (52.0 C)
6	FEP0_RAM	FEP_0 RAM	3313 (-10 C)	3098 (0.0 C)	2323 (46.0 C)	2281 (51.1 C)
7	FEP0_FB	FEP_0 Frame Buffer	3313 (-10 C)	3098 (0.0 C)	2352 (43.0 C)	2306 (48.0 C)
8	FEP1_MONG	FEP_1 Mongoose	3313 (-10 C)	3098 (0.0 C)	2297 (49.0 C)	2259 (54.1 C)
9	FEP1_PCB	FEP_1 PC Board	3313 (-10 C)	3098 (0.0 C)	2323 (46.0 C)	2281 (51.1 C)
10	FEP1_ACTEL	FEP_1 ACTEL	3313 (-10 C)	3098 (0.0 C)	2306 (48.0 C)	2266 (53.1 C)
11	FEP1_RAM	FEP_1 RAM	3313 (-10 C)	3098 (0.0 C)	2306 (48.0 C)	2266 (53.1 C)
12	FEP1_FB	FEP_1 Frame Buffer	3313 (-10 C)	3098 (0.0 C)	2352 (43.0 C)	2306 (48.0 C)

2. Description of the Patch

The ACIS back-end processor's (BEP) DEA housekeeping task monitors DPA voltages and temperatures, calling `Tf_Dea_Housekeeping_Data::append_Entries()` to pack the `ccdId`, `channelId` and data values into telemetry packets for subsequent downlink. The *deahktrip* patch replaces this routine with one that performs the same operations with additional code to compare the component temperatures against a table of temperature limits.

The replacement `append_Entries`() routine is controlled by the static `ndhk` structure, which begins with the `state` word, whose 4 least-significant bits determine how the patch is to function. The following table describes these four flags. The rightmost column shows whether `append_Entries`() alters the flag bit during execution. If not, the flag, once initialized, will retain its value unless changed by a *writeBep* command.

All flags are initialized to zero, *i.e.*, when a temperature channel goes outside its limits, no science run will be halted, the boards won't be powered down, but the software bilevels will be set to 14 ('`1110b`') for a period of `ndhk.delay` seconds. Note that it is *always* necessary to set `NDHK_TEST=1` when testing *deahktrip* on the ACIS Engineering Unit but *never* on the flight unit.

Name	Bit	Value	Description of Bit Field	Altered?
`NDHK_TRIP`	0	0	Set when a channel value is found to be outside `ndhk` limits	Yes
		1	Cleared `ndhk.delay` seconds after NDHK_TRIP first set	
`NDHK_HALT`	1	0	Don't halt a run or power-down video or FEP boards	No
		1	Halt any science run in process and power down the boards	
`NDHK_NBLV`	2	0	Set software bilevels to 14 (1110b) while `NDHK_TRIP` set	No
		1	Do not alter the bilevel values when a channel value is exceeded	
`NDHK_TEST`	3	0	Normal operation on ACIS flight unit	Yes
		1	Used when testing on ACIS Engineering Unit	

The C++ patch code (see below) defines a new public sub-class of the `Tf_Dea_Housekeeping_Data` class to which the original method belongs. The `ndhk` structure defines the upper and lower limits, but the initial values (see the top of Page 4) effectively ignore the upper DN values (*i.e.*, the *lower* temperatures). All `ndhk` fields can be changed from the ground by means of *writeBep* commands.

After compilation and linking, the patch is converted to a series of *addPatch* commands for the *bcmd* program, with an additional *addPatch* to overwrite the start of the original `append_Entries`() routine with instructions to make an unconditional jump to the start of the replacement routine. When a temperature value goes out of limits, the `NDHK_HALT` and `NDHK_NBLV` flags determine the patch behavior. A *bepReadReply* packet containing the `ndhk` structure will be written to telemetry and unless `NDHK_NBLV=1`, software bilevel values will be set to `LED_BOOT_SPARE2` for a period of `ndhk.delay` seconds after *deahktrip* has triggered. When `NDHK_HALT=1`, `append_entries`() will also halt any science run and *biasThief* task already in process, and power down the video and FEP boards.

```
#define NDHKT      (12)     // maximum number of channels checked
#define NDHK_NERR  (17)     // science run error code
#define NDHK_TRIP  (1)      // =1 if channel limit exceeded
#define NDHK_HALT  (2)      // =1 to halt science run, power down boards
#define NDHK_NBLV  (4)      // =1 to suppress report via bilevels
#define NDHK_TEST  (8)      // =1 to force alarm (for EU testing)

typedef struct {
   unsigned low;            // low DN (high temperature) limit value
   unsigned high;           // high DN (low temperature) limit value
   unsigned count;          // count of consecutive trips
} NDHK_VAL;

typedef struct {            // static channel limit table
   unsigned state;          // =0 until tripped, then =1
   unsigned sample;         // conditioning sample size
   unsigned delay;          // seconds before resuming testing
   unsigned cmdid;          // commandId for bepReadReply packet
   unsigned size;           // number of channels used in lim array
   unsigned base;           // index of lowest channel id
   unsigned lowvalid;       // lowest valid DN value (lower values are “Hot”)
   unsigned highvalid;      // highest valid DN value (higher values are “Cold”)
   unsigned tick1;          // bepTickCounter of first tripped packet
   unsigned tick2;          // bepTickCounter of second tripped packet
   unsigned spare;          // for debugging purposes
   NDHK_VAL lim[NDHKT];     // lowest,highest,count channel limit values
} NHKD;

class Test_Tf_Dea_Housekeeping_Data : public Tf_Dea_Housekeeping_Data {
public:
  void append_Entries(unsigned Ccd_Id, unsigned Query_Id, unsigned Value);
};
```

```
// A single static instance of the NDHK structure with all high-temperatures
// set to 9/18/17 red limits and low-temperature limits disabled (DN=4096)
NHKD ndhk = { 0, 2, 3600, 1010, 12, 1, 2060, 4096, 0, 0, 0
  {
   { 2297, 4096, 0 },  /* BEP_PCB    */    { 2314, 4096, 0 },  /* BEP_OSC    */
   { 2266, 4096, 0 },  /* FEP0_MONG  */    { 2289, 4096, 0 },  /* FEP0_PCB   */
   { 2274, 4096, 0 },  /* FEP0_ACTEL */    { 2281, 4096, 0 },  /* FEP0_RAM   */
   { 2306, 4096, 0 },  /* FEP0_FB    */    { 2259, 4096, 0 },  /* FEP1_MONG  */
   { 2281, 4096, 0 },  /* FEP1_PCB   */    { 2266, 4096, 0 },  /* FEP1_ACTEL */
   { 2266, 4096, 0 },  /* FEP1_RAM   */    { 2306, 4096, 0 },  /* FEP1_FB    */
  }
};
void Test_Tf_Dea_Housekeeping_Data::append_Entries(unsigned Ccd_Id,
  unsigned Query_Id, unsigned Value)
{
  // Check that we’re not in a triggered state and the channel is Board 11/12
  if ((ndhk.state & NDHK_TRIP) == 0 && Ccd_Id == 10 && ndhk.size > 0) {
    int ii = QueryId - ndhk.base;
    // Execute if this is a desired channel
    if (ii >= 0 && ii < ndhk.size && ii < NDHKT) {
      // Check if the value violates a limit
      if (ndhk.state & NDHK_TEST) {
        ndhk.state |= NDHK_TRIP;
      } else if ((Value > ndhk.lowvalid && Value <= ndhk.lim[ii].low) ||
          (Value < ndhk.highvalid && Value >= ndhk.lim[ii].high)) {
        // Increment the counter and trip if over sample limit
        if (++ndhk.lim[ii].count >= ndhk.sample) {
          ndhk.state |= NDHK_TRIP;
        }
      } else {
        ndhk.lim[ii].count = 0;
      }
    }
  }
  // Check for triggered state
  if (ndhk.state & NDHK_TRIP) {
    unsigned tick = (getBufPtr())[4];     // get bepTickCounter from TlmForm
    if ((ndhk.state & NDHK_NBLV) == 0) {
      // Set the software bilevels
      bepReg.showLeds(LED_BOOT_SPARE2);
    }
    if (ndhk.tick1 == 0) {
      // Execute once in same housekeeping packet as trigger
      ndhk.tick1 = tick;
      ndhk.tick2 = 0;
      // Stop science mode and biasThief, if running
      if ((ndhk.state & NDHK_HALT) && scienceManager.currentMode != 0) {
        *(unsigned *)&scienceManager.currentMode->termReason = 17;
        scienceManager.notify(ScienceMode::EV_SM_ABORT_RUN);
      }
      if (ndhk.state & NDHK_TEST) {
        swHousekeeper.report(SWSTAT_CMDECHO_DROPPED, ndhk.cmdid);
        ndhk.state &= ~NDHK_TEST;
      }
    } else if (tick != ndhk.tick1 && ndhk.tick2 == 0) {
      // Excecute once in next housekeeping packet following trigger
      ndhk.tick2 = tick;
   if (ndhk.state & NDHK_HALT) {
    // power down all FEP and video boards
    sysConfigTable.changeEntry(SYSSET_DEA_POWER, 0);
    sysConfigTable.changeEntry(SYSSET_FEP_POWER, 0);
   }

   // write the contents of the ndhk structure to telemetry
   unsigned *a = (unsigned *)&ndhk;
   unsigned w = sizeof(ndhk)/sizeof(unsigned);
   CmdResult rc = memoryServer.readBep(ndhk.cmdid, a, w, TTAG_READ_BEP);
   if (rc != CMDRESULT_OK) {
    swHousekeeper.report(SWSTAT_CMDECHO_DROPPED, ndhk.cmdid);
   }
  } else if (tick != ndhk.tick1 && tick != ndhk.tick2 &&
   tick > ndhk.tick1 + ndhk.delay*Acis::TICKS_PER_SECOND) {
   // Execute once at least ndhk.delay seconds after trigger packet
   ndhk.state &= ~NDHK_TRIP;
   ndhk.tick1 = ndhk_tick2 = 0;
   // Clear the channel counters
   for (int ii = 0; ii < NDHKT; ii++) {
    ndhk.lim[ii].count = 0;
   }
  }
 }
 // ---- Continue with the code of the original append_Entries() method ----
}
```

The first block of new code, executed when `NDHK_TRIP` is deasserted, confirms the validity of the `ndhk` structure and checks the subroutine arguments (or `NDHK_TEST` when testing *deahktrip* on the ACIS EU) for an alert, setting `NDHK_TRIP` if found and persisting for `ndhk.sample` consecutive DEA housekeeping samples. The second block of code, executed while `NDHK_TRIP` remains asserted, sets the 4-bit software bilevel values to `LED_BOOT_SPARE2` (unless `NDHK_NBLV=1`) and then performs one of three actions depending on the values of `ndhk.tick1` and `ndhk.tick2`.

1. If `ndhk.tick1` is zero, *i.e.*, in the same call to `append_Entries`() as when the trip occurred, `NDHK_TEST` is cleared, the BEP interrupt timer value is saved in `ndhk.tick1`, and, if `NDHK_HALT` is asserted and a science run is in progress, the task manager is commanded to signal the *scienceManager* and *biasthief* tasks to terminate immediately. Within the patch, the "interrupt timer value" is its value at the time that the *deaHousekeepingData* packet was initialized, and is located in the 4th word of the packet. Retrieved via a call to `getBufPtr`(), the patch can determine whether it is executing while filling the same packet as the trip, or the next packet, or sometime later.

2. Otherwise, if `ndhk.tick1`, is non-zero but `ndhk.tick2` is zero, *i.e.*, `append_Entries`() is being called to update the next *deaHousekeepingData* packet after the packet in which the trip occurred, the patch sets `ndhk.trip2` to the interrupt timer value, tells the *memoryManager* task to write a copy of the `ndhk` structure to telemetry and, if `NDHK_HALT` is asserted, it instructs the *configurationManager* task to power down all video and FEP boards.

3. Otherwise, when at least `ndhk.delay` seconds have elapsed since `NDHK_TRIP` was first asserted, the patch deasserts it and clears `ndhk.tick1`, `ndhk.tick2`, and the channel counters.

This 3-step procedure, using `ndhk.tick1` and `ndhk.tick2` to ensure that Step 1 and Step 2 occur while constructing different *deaHousekeepingData* packets, forces a delay of several seconds between stopping the *scienceManager* and powering down the boards, thereby avoiding a potential FEP-BEP bus crash.

3. Controlled Sources

deahktrip	
deahktrip.C	Source code for the `Test_Tf_Dea_Housekeeping_Data` class
deahktrip.mak	Makefile script to generate *opt_deahktrip.bcmd* for a multi-patch release
deahktrip.pkg	Script to describe this patch and assist with release compilation and testing
eco-1052.doc	Engineering change order describing the *deahktrip* patch, *i.e.*, this document
standalone.mak	Generate a stand-alone *deahktrip.bcmd* file
deahktrip/testsuite	
makebias.pl	Perl script to define a bias map for the Image Loader
makeimage.pl	Perl script to define an eventful frame for the Image Loader
deahktrip/testsuite/smoke	
Makefile	Run a test with the *deahktrip* patch
aux.tcl	Additional *expect* procedures to define parameter blocks
opt_ deahktrip.bcmd	*deahktrip* patch in *bcmd* format linked with standard G optional H patches
opt_ deahktrip.map	Load map for the *deahktrip.bcmd* patch linked with standard G optional H patches
runtest.tcl	*expect* script to demonstrate correct *deahktrip* operation in telemetry format 2
runtest2.tcl	*expect* script to demonstrate correct *deahktrip* operation in telemetry format 1

4. Testing

All tests were performed on the ACIS Engineering Unit using the L-RCTU interface. After setting up a 2-way interface to the EU, the tests were controlled by a script written in the *expect* dialect of TCL. When the *dearepl* patch is applied, the BEP's requests for DPA component temperatures return DN values of zero, leaving nothing on which *deahktrip* can trigger. For this reason, the patch will be triggered by asserting the `NDHK_TEST` flag via a *writeBep* command.

4.1. Test to verify correct behavior of the patch in Format 2

The interface is established by typing "`make shim`" in the "*testsuite/smoke*" directory. Then typing "`make report`", an *expect* procedure, "*runtest.tcl*" (listed in Appendix A) performs a simple test of red-alert detection, as enumerated below. Finally, the interface is removed by typing "`make unshim`".

1. A command pipe is spawned, through which ACIS commands will be sent to the EU.
2. A telemetry pipe is spawned, terminating in the "*psci -m -u -EacisEUbilevels.ttm*" packet-monitoring filter, with *expect* examining the standard output. Note the "*–E*" option which, when combined with "*–m*", instructs *psci* to report ACIS bilevel values found in *pseudoEngineering* packets.
3. The EU BEP is cold-booted.
4. The *opt_deahktrip* flight patch is applied, along with *opt_dearepl* to configure the BEP to take pixel data from the Image Loader. No other patches are necessary when running a "stand alone" test.
5. The EU BEP is warm-booted.
6. A *writeBep* command is sent to the BEP to set the contents of the `ndhk` structure to those in the table at the top of page 4.
7. A *changeConfigSetting* command powers up 6 FEPs and 6 video boards. The script waits 60 seconds for this to complete.
8. A *loadDeaBlock* command is sent to the BEP calling for the 12 DPA component temperatures to be reported in *deaHousekeepingData* packets, with a 10 second delay following each set of 12 readouts.
9. A *startDea* command is sent to begin reporting the DEA housekeeping packets.

10. The ACIS pixel switch is commanded to send pixel streams to the FEPs from the Image Loader.
11. A suitable bias map is created by "*testsuite/makebias.pl*" and written to the Image Loader.
12. *loadTeBlock* and *startScience* commands are sent to control and start a timed-exposure science run that is designed to generate a large number of events, so that *deahktrip* will be tested while the BEP output buffer is saturated.
13. The script waits for the first *dataTeFaint* packet, after which it replaces the bias map in the Image Loader with a frame generated by "*testsuite/makeimage.pl*" that contains multiple event candidates.
14. The script continues to monitor the telemetry packets. After receiving 4 more *deaHousekeepingData* packets, a *writeBep* command sets `ndhk.state` to 10 decimal. This asserts `NDHK_TEST`, and `NDHK_HALT` causing the patch to trip immediately, to halt the science run, and power down the boards.
15. If the script encounters (a) a *bepReadReply* packet with a `commandId` value of `1010`, **and** (b) an *engineeringPseudo* packet with a software bilevel value of 14, **and** (c) a *scienceReport* packet with a *terminationCode* of 17, the test passes.
16. Otherwise, if *expect* times out after 3600 seconds or receives more than 100 *deaHousekeepingData* packets, the test fails.

4.2. Test to verify correct behavior of the patch in Format 1

This test is controlled by "*testsuite/smoke/runtest2.tcl*". It is not one of the acceptance tests since it requires the special *shim500* interface. The user first types "`make shim500`" to set up the Format 1 interface, then "`make report N=2`" to run the script, and finally "`make unshim`" to remove the interface. The steps are identical to those in §4.1 above, except that the "*runtest2.tcl*" parameters differ from those in Appendix C as follows:

```
set ccd_list     {10 0 1 2 3 7}       ; # Use only 5 CCDs and FEPs
set datarate     {20}                 ; # Measure of event rate
set delay        {14400}              ; # Delay in seconds until trip reset
set pmode        {3}                  ; # bepPackingMode (Event Histogram)
set phist        {1650}               ; # histogramCount value for Format 1
```

Note that this patch commands the EU to run 5 video boards and FEPs in timed-exposure event histogram mode and therefore uses the *opt_eventhist* and *opt_smtimedlookup* patches, along with a version of *opt_deahktrip* that has been linked with them. These patches must be built together in the "*patches/release/options/BUILD*" directory and "*opt_deahktrip.bcmd*" and "*opt_deahktrip.map*" must be copied to "*deahktrip/testsuite/smoke*" before running "*runtest2.tcl*" in this directory.

5. Makefile targets

"*testsuite/smoke/Makefile*" defines the following targets for use within "*runtest.tcl*" or from the user's console:

`all` (default)	Execute "*runtest.tcl*" with output to *stdout* (or "*runtest2.tcl*" with "`make N=2`")
`report`	Execute "*runtest.tcl*" with output to "*deahktrip*`.date.time`*.log*"
`shim`	Start the Format 2 interface to the ACIS Engineering Unit
`shim500`	Start the Format 1 interface to the ACIS Engineering Unit
`unshim`	Stop the interface to the ACIS Engineering Unit
`loaderselect`	Command the Pixel Switch to send contents of the Image Loader to the FEPs
`deaselect`	Command the Pixel Switch to send the output of the video boards to the FEPs
`reload`	Set the contents of the `ndhk` structure to the default values
`bias`	Send a bias map image frame to the Image Loader
`image`	Send an image frame containing multiple events to the Image Loader

Appendices

A. The *runtest.tcl* test script

This *expect* script starts a timed-exposure science run using 6 FEPs with input from the ACIS Image Loader. After receiving the first *exposureTeFaint* packet, the bias image is replaced by a one containing many events. The script then counts *deaHousekeepingData* packets and, after the fourth, it sets `ndhk.state` to `$state`, asserting the `NDHK_TRIP` flag and causing the patched `append_Entries`() routine to trip. The script also monitors for *bepReadReply*, *scienceReport* and *engineeringPseudo* packets.

```
#! /usr/bin/env expect
puts “Welcome to deahktrip/testsuite/patches/deahktrip”
# ---- Load options from command line ----
lassign $argv basedir tools patchdir
# ---- Define execution parameters ----
set ccd_list {0 1 2 3 4 5}   ; # CCDs to assign to FEP_0 .. FEP_5
set state    {10}            ; # Initial value of ndhk.state
set datarate {50}            ; # Measure of event rate
set delay    {3600}          ; # Delay in seconds until trip reset
set timeout  {300}           ; # Default timeout in seconds
set cmdId    {1010}          ; # commandId for bepReadReply
set pmode    {0}             ; # bepPackingMode value
set phist    {0}             ; # histogramCount value (when pmode=3)
set ncmd     {0}             ; # commandId for commands sent to EU
set ndeahk   {0}             ; # bepReadReply packet counter
set nbilevel {0}             ; # Bilevel alarm counter
set nscirep  {0}             ; # scienceReport packet counter
# ---- Load expect commands library and pblock definitions ----
source “$basedir/$tools/lib/lib-exp/runtest_support.tcl”
source “$basedir/$patchdir/aux.tcl”
# ---- Start the I/O ----
spawn “$basedir/$tools/bin/cmdclient” $env(ACISSERVER)
set cmd_id $spawn_id
spawn “$basedir/$tools/bin/tlmclient” $env(ACISSERVER) -E$env(ACISTTMFILE)
sleep 1
match_max 400
# ---- Save the hex address of the ndhk structure in $addr ----
lassign [exec grep {D ndhk} "$basedir/$patchdir/opt_deahktrip.map"] addr
# ---- Halt BEP, load patches, warm boot ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcmd\
                 $basedir/$tools/share/opt_printswhouse.bcmd\
                 $basedir/$tools/share/opt_dearepl.bcmd\
                 $basedir/$patchdir/opt_deahktrip.bcmd"
warm_boot
# ---- Upload the initial ndhk structure ----
set ndhk "0 2 $delay $cmdId 12 1 2060 4096 0 0 0"
foreach ii {2297 2314 2266 2289 2274 2281 2306 2259 2281 2266 2266 2306} {
   append ndhk " $ii 4096 0"
}
send -i $cmd_id "write [incr ncmd] 0x$addr {\n$ndhk\n}\n"
command_echo 1 192 {initialize ndhk}
# ---- Power up FEPs and video boards ----
power_on_boards $ccd_list
```

```
expect -re "SWSTAT_FEPMAN_ENDLOAD: 5\[\r\n]+" {} timeout { fail timeout }
# ---- Start DEA housekeeping ----
send -i $cmd_id "load [incr ncmd] dea 4 {[deaHkPblock 10]}\n"
command_echo 1 13 {load dea pblock}
send -i $cmd_id "start [incr ncmd] dea 4\n"
command_echo 1 18 {start dea housekeeping}

# ---- Prepare image loader and load a bias image ----
system make loaderselect bias

# ---- Load and start TE science run ----
send -i $cmd_id "load 0 te 4 {[teImagePblock $ccd_list $pmode $phist]}\n"
command_echo 1 9 {load te pblock}
send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 {start te science run}
# ---- Wait to create bias maps, then load event image ----
set timeout $delay
expect -re “SWSTAT_FEP_STARTDATA[^\r\n]*\[\r\n]+” timeout {fail timeout }
system make image RATE=$datarate
expect -re “dataTe[^\r\n]*\[\r\n]+” timeout { fail timeout }
# ---- Examine the psci monitor output ----
expect {
    -re "bepReadReply\[^\r\n]*commandId=$cmdId\[^\r\n]*\
        requestedAddress=0x$addr\[^\r\n]*\[\r\n]+" {
        incr nbeprep ; exp_continue
    }
    -re "scienceReport\[^\r\n]*terminationCode=17\[\r\n]+" {
        incr nscirep ; exp_continue
    }
    -re "engineeringPseudo\[^\r\n]*bilevels=(\[0-9]+)\[\r\n]+" {
        if {$expect_out(1,string) & 15) == 14} {
            incr nbilevel
        }
        if {$nbilevel < 1 || $nbeprep != 1 || $nscirep != 1} {
            exp_continue
        }
        pass " $nbilevel bilevels, $nbeprep BEP reads, $nscirep sci reports "
    }
    -re "deaHousekeepingData\[^\r\n]*\[\r\n]+" {
        if {[incr ndeahk] == 4} {
            send -i $cmd_id "write [incr ncmd] 0x$addr {\n$state\n}\n"
        }
        if {$ndeahk < 100} { exp_continue }
    }
    timeout { }
}
# ---- Fall through on timeout or 100+ housekeeping packets ----
fail “ timeout: $nbilevel bilevels, $nbeprep BEP reads, $nscirep sci reports “
```

B. Timing

Using the results of the tests in §4, the following table lists the time delays from the moment a temperature channel first exceeded its limit to (a) the time the alarm tripped, the bilevels were set, and the science run was terminated, and (b) the FEP and video boards were powered down. The `ndhk.sample` value was 2. The event rates and histogram counts were selected to cause the data packets to saturate the BEP's output buffer.

Test	Format	Science Stopped	Powered Down	Reported
Timed Exposure Faint 3x3	2	20.0	57.2	190.5
Timed Exposure Event Histogram	2	17.0	35.0	112.6
Timed Exposure Faint 3x3	1	2434.6	2452.8	9302.3
Timed Exposure Event Histogram	1	5000.4	5017.4	11075.9

The delays in the "Science Stopped" column result from the time delay between *deaHousekeepingData* packets which is required by the `ndhk.sample=2` criterion. As soon as the science run is stopped, BEP buffer space becomes available for a new *deaHousekeepingData* packet, during whose construction the FEPs and video boards can be powered down. The "Reported" column then includes time for the remaining science packets to be read out from the BEP.

C. Glossary

bcmd	Command to translate ASCII commands to binary; also the command format itself
BEP	ACIS Back-End Processor — a component of the DPA
smoke	Directory containing tests designed to reproduce the action of a software patch
CCD	Charge Coupled Device
DEA	ACIS Detector Electronics Assembly comprising analog and interface boards
DN	Raw data value, *i.e.*, not converted to C, volts, amps, etc.
DPA	ACIS Digital Processor Electronics comprising front- and back-end processor boards
EU	ACIS Engineering Unit — hardware simulator of the DEA and DPA
expect	Interactive input/output scripting interpreter based on TCL
FEP	ACIS Front-End Processor — a component of the DPA
L-RCTU	Jim Littlefield's Remote Command and Telemetry Unit, interface to the EU
OBC	Chandra On-Board Computer
SCS106	Stored OBC command sequence to protect the ACIS payload from high temperatures
TCL	Tool Command Language, a tiresome scripting language best avoided whenever possible

D. Applicable Documents

1. DPA/DEA Interface Control Document, MIT 36-02205, Revision C, March 10, 1995.

2. ACIS Science Instrument ACIS Software Instrumentation, Program and Command List, MIT 36-53204.0204, Revision N, March 15, 2001.

3. ACIS Science Instrument Software Detailed Design Specification (As Built), MIT 36-53200, Revision 01, February 3, 2000.

4. Creating and Testing ACIS Flight Software Patches, MIT ACIS Report, Revision 1.0, June 6, 2016.

5. Proposed BEP and FEP Limits, Paul Plucinsky, e-mail, September 18, 2017.

```
-------------------------------------------------------------------------------

TITLE: ACIS Flight Software Standard Patch Component Release Notes

DOCUMENT NUMBER: 36-58010        REVISION: G

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
D       36-1039        Add new patches, retest          RFG            09/29/2009
E       36-1042        Update buscrash2, retest         RFG            01/06/2010
F       36-1048        Update buscrash2, remove biastim RFG            12/16/2013
G       36-1053        Update buscrash, retest          RFG            06/29/2018
G       36-1053        Update buscrash
```

```
======================================================================

Title: ACIS Patch Release Notes for Version G

Software Change Order: 36-1053

Build Date:   Fri Jul 13 08:47:25 EDT 2018
Part Number:  36-58010
Version:      G
CVS Tag:      release-G

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

Load Size:    3828 bytes

----------------------------------------------------------------------
Description:
    This is the seventh letter release of the standard patch set and the
    eighth letter release of the optional patches for the ACIS Flight Software.

    The purpose of this release is to update the buscrash patch and add
    the deahktrip patch.

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

              corruptblock    - Fixes SPR 113
              digestbiaserror - Fixes SPR 116
              histogramvar    - Fixes SPR 115
              rquad           - Fixes SPR 121
              histogrammean   - Fixes SPR 123
              zap1expo        - Addresses SPR 122
              condoclk        - Addresses SPR 127
              fepbiasparity2  - Addresses SPR 130
              cornermean      - Fixes SPR 128
              tlmbusy         - Fixes SPR 138
              buscrash        - Fixes SPR 140 and 151
              badpix          - Fixes SPR 141
              buscrash2       - Fixes SPR 133, 142, 148, 150

    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-141
    SPR-127
    SPR-142
    SPR-138
    SPR-116
    SPR-128
    SPR-113
    SPR-121
    SPR-122
    SPR-148
    SPR-115
    SPR-123
    SPR-151
    SPR-130

----------------------------------------------------------------------
Included Patches:
    digestbiaserror
    badpix
    cornermean
    rquad
    histogrammean
    corruptblock
    zap1expo
    tlmbusy
    fepbiasparity2
    buscrash
    histogramvar
    buscrash2
    condoclk

----------------------------------------------------------------------
Additional Release Level Tests:
```

```
======================================================================

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:


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: badpix

```
Part Number: 36-58030.21
Version:     A
SCO:         36-1037
```

Description:

Reason:
This patch fixes software problem report SPR-141.

Symptom:
The known bad pixels and columns supplied to ACIS through its bad pixel and column lists are not always being flagged in the correct locations in the FEP bias maps. The symptom only appears when the instrument is running in timed-exposure mode using sub-arrays whose initial row number is greater than zero.

Symptom Impact:
In most timed-exposure sub-array runs, when the sub-array starts after the first CCD row, bad pixel will be mis-located; the truly bad pixels will be accepted as valid and good pixels will be treated as bad. In practice, this will have little effect since bad pixels will be recognized by the bias map creation algorithm.

Symptom Cause:
The BEP maintains a list of known bad pixels and columns in each CCD. After a bias map is created, the BEP’s loadBadMaps procedure will set the appropriate entries in the FEPs bias maps to 4095, telling the FEP software to ignore the corresponding image pixel, i.e., treat it as if it had zero value. This is in addition to any saturated pixels found during bias map creation, which will also be assigned the bias value 4095.

The code in SmTimedExposure::loadBadMaps() contains an error. It assumes that sub-arrays will be processed in the same relative location in a FEP’s image and bias memory as on the CCD from which the pixels originated. This is not so--the first row of a sub-array is always written into row 0 of a FEP’s image map, and the corresponding bias values are saved in row 0 of its bias map.

SmTimedExposure::loadBadMaps() must be patched in two places, one to correct bad pixels, the other bad columns. The bad pixel correction is applied as follows:

```
  while (badPixelMap.getPixel (index, ccd, row, col) == BoolTrue) {
    if ((row >= start) && (row < end)) {
      row /= sum;
      col /= sum;
      for (FepId fep = FEP_0; fep < FEP_COUNT; fep = FepId(fep+1)) {
        if (fepCcd[fep] == ccd) {
          fepManager.loadBadPixel (fep, row, col);
        }
      }
    }
    index++;
  }
```

and we want to change the "row /= sum" to "row = (row-start) / sum". This can best be done by recognizing that "sum" has only two values, 1 or 2, and the MIPS takes 32 bytes of code to perform an unsigned integer divide, but only 4 bytes to perform a logical right shift.

The original assembler code

```
1774 2400A28F   lw      $2,36($sp)
1778 00000000   divu    $2,$2,$18
177C 1B005200
1780 02004016
1784 00000000
1788 0D000700
1798 2400A2AF   sw      $2,36($sp)
```

can simply be modified as follows:

```
1774 2400A28F   lw      $2,36($sp)
1778 FFFF4326   addu    $3,$18,-1
177c 23105600   subu    $2,$2,$22
1780 06106200   srl     $2,$2,$3
1784 00000000   nop
1788 00000000   nop
178C 00000000   nop
1790 00000000   nop
1794 00000000   nop
1798 2400A2AF   sw      $2,36($sp)
```

The second patch sets the starting value of the row loop to zero:

```
  while (badTeColumnMap.getColumn (index, ccd, col) == BoolTrue) {
    col /= sum;
    for (FepId fep = FEP_0; fep < FEP_COUNT; fep = FepId(fep+1)) {
      if (fepCcd[fep] == ccd) {
        for (unsigned row = start; row < end; row++) {
          fepManager.loadBadPixel (fep, row, col);
        }
      }
    }
    index++;
  }
```

The existing assembler code is

```
$LM1578:
18cc 0000043C   la  $4,fepManager
18d0 00008424
18d4 21282002   move    $5,$17
18d8 3000A78F   lw      $7,48($sp)
18dc 00000000   nop
18e0 0000000C   jal     loadBadPixel
18e4 21300002   move    $6,$16
18e8 01001026   addu    $16,$16,1
18ec 2B101402   sltu    $2,$16,$20
18f0 F6FF4014   bne     $2,$0,$L1578
```

and the patch replaces the row in the loadBadPixel(fepId, row, col) call with row-start. (In the MIPS architecture, the instruction after a branch or call is executed before the branch is taken).

```
18e4 23301602   subu    $6,$16,$22
```

Applicable Reports/Requests:

SPR-141

Test Results:

```
Replaced Functions:


Command Impact:
    None.


Telemetry Impact:
    None.


Science Impact:
    Without this patch, the BEP’s bad pixel and bad column lists will be
    applied incorrectly in timed-exposure sub-array mode when the sub-array
    begins on any but the first row of the CCD. Since almost all science
    runs are made in dithered mode, the impact once the patch is in place
    will be slight.
```

```
======================================================================

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:

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: 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->get
Quadrant (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,$18
                                                .set    macro
                                                .set    reorder

                           05ec 21884000        move    $17,$2
                                            $LM85:
                           ../filesscience/pixel3x3.C ****     doclk[2] = exposure->get
OverclockDelta (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)
                                            $LM86:
                           ../filesscience/pixel3x3.C ****     split[2] = exposure->get
SplitThreshold (quad);
                                                .stabn 68,0,213,$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)
                                            $LM87:
                           ../filesscience/pixel3x3.C ****
                           ../filesscience/pixel3x3.C ****     computePhGrade (doclk, s
plit);
                                                .stabn 68,0,215,$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(FEPeventRe
c3x3 const *, EventExposure *)
                                            $LM91:
```

Applicable Reports/Requests:
    SPR-121

Test Results:

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

    (d) increment mean sum
```

```
                .globl  fepSciTimed_lst_1acc_1adc
                .ent    fepSciTimed_lst_1acc_1adc
        fepSciTimed_lst_1acc_1adc:
1ab0 1B006A00
     02004015
     00000000
     0D000700
     12180000
1acc 34050925       addu    $9,$8,1368-36
1ad0 4805028D       lw  $2,1368-16($8)
1ad4 00000000       nop
1ad8 21104300       addu    $2,$2,$3
1adc 480502AD       sw  $2,1368-16($8)
1ae0 1B00AA01
1ae4 02004015
1ae8 00000000
1aec 0D000700
1af0 12200000
            .end    fepSciTimed_lst_1acc_1adc
```

(e) save stack pointer in R9

```
                .globl  fepSciTimed_lst_1c38_1c38
                .ent    fepSciTimed_lst_1c38_1c38
        fepSciTimed_lst_1c38_1c38:
1c34 1403028E
1c38 48050926       addu    $9,$16,1368-16
1cec 22004010
            .end    fepSciTimed_lst_1c38_1c38
```

(f) load overclock mean sum

```
                .globl  fepSciTimed_lst_1c50_1c50
                .ent    fepSciTimed_lst_1c50_1c50
        fepSciTimed_lst_1c50_1c50:
1c4c 21187200
1c50 0000228D       lw  $2,0($9)
1c54 00000000
            .end    fepSciTimed_lst_1c50_1c50
```

(g) load overclock variance sum

```
                .globl  fepSciTimed_lst_1c84_1c84
                .ent    fepSciTimed_lst_1c84_1c84
        fepSciTimed_lst_1c84_1c84:
1c80 21187200
1c84 1000228D       lw  $2,16($9)
1c88 00000000
            .end    fepSciTimed_lst_1c84_1c84
```

(h) increment R9

```
                .globl  fepSciTimed_lst_1cb8_1cb8
                .ent    fepSciTimed_lst_1cb8_1cb8
        fepSciTimed_lst_1cb8_1cb8:
1cb4 1403028E
1cb8 04002925       addu    $9,$9,4
1cbc 2B106201
            .end    fepSciTimed_lst_1cb8_1cb8
```

Applicable Reports/Requests:

```
    SPR-123


Test Results:


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 mm, 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.
```

```
======================================================================

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:

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: zap1expo

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[iq
uad]));
     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] = 0xfff;
     937 02b0 FF0F0324              li      $3,0x00000fff
     944 02b4 180043A4              sh      $3,24($2)
     189:fepSciTimed.c ****     FIOsetThresholdRegister(iquad, 0xfff);
     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[iq
uad]));
     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] = 0xfff;
     786 021c FF0F0324              li  $3,0x00000fff
     787 0220 180043A4      sh  $3,24($2)
     177:fepSciCClk.c  ****     FIOsetThresholdRegister(iquad, 0xfff);
     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:

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: 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 enqueuing 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:


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:

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 absense 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: buscrash

Part Number: 36-58030.30
Version:     B
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). Release A
    of this fix interacted badly with the buscrash2 patch in a manner that
    could prevent the science run from termination. This was corrected in
    release B of buscrash.



Applicable Reports/Requests:
    SPR-151


Test Results:


Replaced Functions:
    FepManager::loadBadPixel
    FepManager::pollBiasComplete
```

```
Command Impact:
    None.


Telemetry Impact:
    None.


Science Impact:
    None.
```

```
======================================================================

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 occassionally 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 occassionally providing bad variance values
    for the overclocks.

    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:


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
    occassionally be incorrect. Once this patch is installed, the
    Flight Software correctly computes overclock variances.
```

```
======================================================================

Patch Name: buscrash2

Part Number: 36-58030.32
Version:     C
SCO:

Description:

    Reason:
    If ACIS is copying bias maps to telemetry 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 patch that will fix it.

    At the same time, the cause of trickle-bias anomalies has been found to be
    related to the way the BEP task manager relays events to the bias thief task.
    Code has been added to the buscrash2 patch to overcome this problem. Should it
    recur, a test has been added to buscrash2 that will end bias trickling so that
    the anomaly doesn’t cause T-plane latch-ups in FEPs.

    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.

    In addition, the task manager will occasionally run the science and bias thief
    tasks simultaneously, so that bias packets and exposure records will be
    interleaved in ACIS telemetry. This situation is likely to cause the threshold
    crossing planes of one or more FEPs to "latch-up". In this condition, they will
    not correctly identify event candidates, thus preventing events from that CCD to
    be reported.

    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.

    The trickle-bias anomaly is likely to block all events from one or more FEPs
    for that science run and for all subsequent runs until the latched FEP is power-cyc
led.

    Symptom Cause:
    The bus crash has been traced to a flaw in the BiasThief::checkMonitor()
    method. This routine is executed after the FEP bias maps have been
    created and it copies them to telemetry. It uses the memory-mapped
    interface between BEP and FEP to access the maps but, unlike other
    FepManager operations, it does not confirm that a FEP is powered up before
    it reads the maps.  This causes the bus crash.

    The trickle-bias anomaly is most likely caused by the task manager failing to
    merge a pair of events, EV_TASKQUERY and EV_START, sent to the bias thief task.

    Fix Description:
    To prevent a bus crash following an SCS107, call the FepManeger::isEnabled()
```

method to check if the FEPs are powered up before reading from a FEP’s bias memory. This is done by adding the following code to BiasThief::checkMonitor():

```
// ---- Check whether the FEPs are powered up ----
for (unsigned fepid = 0; fepid < FEP_COUNT; fepid++) {
    if (fepInfo[fepid].base != 0 &&
        fepManager.isEnabled(FepId(fepid)) == BoolFalse) {
        swHousekeeper.report(SWSTAT_FEPREC_POWEROFF, fepid);
        retval = BoolFalse;
    }
}
```

To prevent a trickle-bias anomaly from causing FEP T-plane latch-ups, add the following code to BiasThief::checkMonitor():

```
// ---- Check whether BiasThief and Science tasks running together
unsigned start = scienceManager.currentMode->startTimeData;
if (modetype == 0 && start != 0xffffffff) {
    swHousekeeper.report(SWSTAT_SCI_STARTRUN_BUSY,
        systemClock.currentTime());
    memoryServer.readBep(1, (const unsigned int *)this,
        sizeof(BiasThief)/sizeof(unsigned), TTAG_READ_BEP);
    retval = BoolFalse;
}
```

To entirely eliminate the trickle-bias anomaly, the BiasThief::biasReady() method has been updated:

```
void Test_BiasThief::biasReady()
{
    abortFlag = BoolFalse;      // Resolve order conflict with abort()
    notify (EV_START);          // Signal task to start bias
    yield();                    // Start the bias thief
    busyFlag = BoolTrue;        // Bias Thief will be active soon
}
```

and the BiasThief::goTaskEntry() method has been rewritten:

```
void Test_BiasThief::goTaskEntry()
{
    DebugProbe probe;

    // ---- FOREVER ----
    for (;;) {
        // --- Wait for start/abort or query from task monitor ---
        unsigned caught = waitForEvent (EV_START | EV_ABORT | EV_TASKQUERY);

        // --- Consume but ignore EV_ABORT signal ---

        // --- Respond to monitor queries ---
        if (caught & EV_TASKQUERY) {
            taskMonitor.respond ();
        }

        // --- Start bias dump ---
        if ((caught & EV_START) && (abortFlag == BoolFalse)) {
            // -- Ensure busyFlag is set
            busyFlag = BoolTrue;

            // -- Trickle bias for each FEP --
            for (unsigned fepid = 0; fepid < FEP_COUNT; fepid++) {
                if (fepInfo[fepid].base == 0) {
                    continue;   // Skip to next FEP
                } else if (modetype == 0) {     // Timed Exposure
```

```
                            if (trickleTeBias (FepId(fepid)) == BoolFalse) {
                                break;
                            }
                        } else { // Continuous Clocking
                            if (trickleCcBias (FepId(fepid)) == BoolFalse) {
                                break;
                            }
                        }
                    }

                    // --- No longer busy ---
                    busyFlag = BoolFalse;
                }
            } // END FOREVER
        }
```

Note that this version of buscrash2 eliminates the need for the standard biastiming patch and the optional untricklebias patch. Hooray!

Applicable Reports/Requests:

SPR-142
SPR-148

Test Results:

Replaced Functions:

BiasThief::checkMonitor
BiasThief::goTaskEntry
BiasThief::biasReady

Command Impact:

None.

Telemetry Impact:

If an active FEP is found to be unpowered during bias copying, no more bias packets will be produced and a SWSTAT_FEPREC_POWEROFF will be reported in software housekeeping.

If the science task is found to have started event processing while bias maps are being copied to telemetry, a SWSTAT_SCI_STARTRUN_BUSY condition will be noted in software housekeeping and no more bias packets will be produced for the current run. In addition, a bepReadReply packet will be generated with the contents of the "biasThief" object at the time of the anomaly.

Science Impact:

Bias maps will be missing or truncated if either an active FEP is found to be powered off during map copying, or if the science task is found to have started event processing before the last bias map has been copied.

==

Patch Name: condoclk

```
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 (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.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 ioclk */
```

with an inline patch that saves R9-R12:

```
      condoclkCtl(fp);

    subu    $sp,$sp,16
    sw  $9,0($sp)
    sw  $10,4($sp)
    sw  $11,8($sp)
    sw  $12,12($sp)
    jal condoclkCtl
    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 condoclkCtl function:

```
void condoclkCtl(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 overclocks 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:


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.
```

TITLE: ACIS Flight Software Optional Patch Component Release Notes

DOCUMENT NUMBER: 36-58020   REVISION: H

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
D	36-1040	Add new patches, retest	RFG	09/29/2009
E	36-1042	No new patches, retest	RFG	01/06/2010
F	36-1044	Add txings patch, retest	RFG	03/02/2011
G	36-1048	Remove untricklebias, retest	RFG	12/16/2013
H	36-1054	Add deahktrip, retest	RFG	06/29/2018
H	36-1053	Add deahktrip		

```
======================================================================

Title: ACIS Optional Patch Release Notes for Version H

Software Change Order: 36-1053

Build Date:   Fri Jul 13 12:54:43 EDT 2018
Part Number:  36-58020
Version:      H
CVS Tag:      release-G-opt-H

Std Number:   36-58010
Std Version:  G
Std Tag:      release-G
Std SCO:      36-1053

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

----------------------------------------------------------------------
Description:
    This is the eighth letter release of the optional patch set for the ACIS
    Flight Software. The purpose of this release is to add the deahktrip
    patch and test the optional patches with the Rev. G standard 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 invidually, based on the patches provided in this release.

    This release consists of the following optional flight patches:

              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
              reportgrade1   - Addresses SPR 132
              smtimedlookup  - Supports eventhist and ctireport
              teignore       - Ignore Timed Exposure data frames
              txings         - Triggers bilevels on excess threshold crossings
              deahktrip      - Triggers when DPA temperatures exceed limits

    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 attached
              fepthrottle    - Reduces FEP event candidates


----------------------------------------------------------------------
Addressed Problem Reports:
    SPR-124
    SPR-134
    SPR-126
    SPR-120
    SPR-132

----------------------------------------------------------------------
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)
    deahktrip (1940 bytes)
    dearepl (556 bytes, conflicts with deaeng)
    eventhist (5908 bytes, depends on smtimedlookup)
    printswhouse (7240 bytes, depends on tlmio)
    reportgrade1 (816 bytes)
    smtimedlookup (3712 bytes)
    teignore (36 bytes)
    tlmio (10312 bytes)
    txings (3176 bytes)
```

```
======================================================================

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:


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: eventhist

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

Conflicts:
Depends On:  smtimedlookup
Size:        5908 bytes

Bcmd File:   opt_eventhist.bcmd
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:


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: compressall

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

Conflicts:
Depends On:
Size:        2368 bytes

Bcmd File:   opt_compressall.bcmd
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:


Replaced Functions:
    PmTeRaw::digestRawRecord
    PmCcRaw::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: ctireport1

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

Conflicts:
Depends On:  smtimedlookup
Size:        5452 bytes

Bcmd File:   opt_ctireport1.bcmd
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 fepManager.loadRunProgram from Test2_SmTimedExposure::setupCti1Fep.
    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 cti1stk
    structure that holds the row indices and bias-subtracted pixel values
    of the most recently located precursor charge in each CCD column.

    The new FEPtestCti1 routine is called from an inline patch within
    FEPsciTimedEvent in advance of the FEPtestOddPixel or FEPtestEvenPixel
    routines. When a threshold crossing is detected, FEPtestCti1 clears
    the cti1stk array (if this is a new frame), calls FEPtestOddPixel or
    FEPtestEvenPixel, and then pushes the pixel value and row index onto
    cti1stk. If cti1stk is full, the most distant (by row) value is
    dropped.

    FEPappendCti1 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 cti1stk
    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:


Replaced Functions:
    smTimedLookupMode[4]
    smTimedSetupFep[4]
    smTimedTerminate[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_CTI1 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 dataTeVeryFaint except that their formatTag fields contain TTAG_SCI_TE_REC_CTI1 and TTAG_SCI_TE_DAT_CTI1, respectively. When a TTAG_SCI_TE_DAT_CTI1 is received, precursor charge data will be located in the dataTeVeryFaint.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: dearepl

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

Conflicts:   deaeng
Depends On:
Size:        556 bytes

Bcmd File:   opt_dearepl.bcmd
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:


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


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: 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-124
    SPR-126
    SPR-120
```

```
Test Results:


Replaced Functions:
    SmContClocking::terminate
    SmContClocking::setupProcess
    SmContClocking::setupFepBlock


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
    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 exisiting 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: 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.bcmd
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:


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: reportgrade1

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

Conflicts:
Depends On:
Size:        816 bytes

Bcmd File:   opt_reportgrade1.bcmd
Pkts File:   opt_reportgrade1.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:


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: txings

Part Number: 36-58030.33
Version:     A
SCO:         none
Environment: flight

Conflicts:
Depends On:
Size:        3176 bytes

Bcmd File:   opt_txings.bcmd
Pkts File:   opt_txings.pkts

Description:
    With the continuing degradation of Chandra's EPHIN radiation monitor,
    an alternative is needed to permit the observatory to take the actions
    necessary to preserve its instruments during times of high solar
    activity. A recent analysis [Grant et al., 2010] has shown that, in
    some circumstances, the signature of solar events can be detected
    within the counts of CCD threshold crossings that are included in
    downlinked telemetry.

    The txings patch monitors threshold crossings and uses ACIS bi-levels
    to communicate an alarm to the Chandra On-Board Computer (OBC). Event
    records are read from the FEP-BEP ring buffers by the processRecord()
    methods of the PmEvent, PmHist, and PmRaw classes. Each calls
    EventExposure::copyExpEnd() to parse the FEPexpEndRec records that
    contain thresholds, the count of threshold crossings, and expnum, the
    exposure number, but this routine doesn't have access to the ccdId that
    labels the record and which is needed to accumulate the crossings from
    that particular CCD.

    The MIPS CPU architecture makes it relatively easy to make inline
    patches that permit additional arguments to be passed to subroutines.
    In the current case, we patch the routines that call copyExpEnd() in
    order to pass an extra argument. When processRecord() is called with a
    PmEvent object, this argument will be the address of the object, but
    for other callers, i.e., PmHist or PmRaw, the argument will be null to
    show that these modes don't count threshold crossings. Since PmEvent is
    a subclass of ProcessMode, the ccdId value can then be determined by a
    call to getCcdId(). A replacement for copyExpEnd() is called with
    an object of class EventExposure, and it calls saveTXings() with a
    static TXings object named txings in which the threshold crossing
    accumulators are stored.

    The saveTXings() method is called once for each event-mode exposure
    frame. The first time that it is called in a science run, it
    determines the number of read-out rows, the maximum anticipated number
    of non-pathological threshold crossings per frame, and the frame
    exposure time in units of the FEP pixel clock (i.e., 10 us), and it
    increments the tx.threshold_accum and tx.exposure_accum accumulators.
    Integration times of less than 2000 seconds are guaranteed not to
    overflow either accumulator. Since the number of rows per frame and the
    frame exposure time are constant in continuous clocking mode, they are
    initialized in the TX structure, but in timed-exposure mode, the frame
    time depends on the dutyCycle, primaryExposure, and secondaryExposure
    parameters. These are extracted from the external pramTe object, where
    they were copied from the science run parameter block when the run
    started.
```

The radiation triggering algorithm is run in the triggerRadmon() routine It is called every 64 seconds whether or not a science run is in progress. If it isn’t, tx.count is set to zero until a subsequent call to saveTXings() from copyExpEnd() reloads the TX parameter structure from TXnext.

After the TXings patch has been uploaded and the BEP warm-booted, the tx.count field will be initialized to zero by the patch loader. The first time an event-mode science run reads a FEPexpEndRec record from the FEP-BEP ring buffer, it will call saveTXings(), which will reinitialize the radiation filter parameters from the TXnext structure. This makes it easy to change the filter parameters for subsequent science runs. When a trigger occurs, triggerRadmon() sets tx.triggered to BoolTrue and commands the memory manager thread to send a bepReadReply packet to telemetry, reporting the values of the txings parameters and variables. Then Test_Leds::show() sets the software bi-level channels to LED_BOOT_SPARE1, which persists for the remainder of the science run. After the science run ends, the next call to Leds::show() calls triggerRadmon() which sets tx.count to zero and tx.triggered to BoolFalse, canceling the special bilevel value and preventing threshold crossing triggers until the next science task starts, calls saveTXings(), and reloads the TX structure.

Once it is included in a patch load, and the BEP is warm-booted, the txings patch will be active during all subsequent science runs. When triggered by high and increasing threshold crossings, it sets the ACIS software bilevel values to LED_BOOT_SPARE1 until the science run ends, or until the tx.triggered field is explicitly cleared by a writeBep command. This guarantees that it will appear in Chandra major frame readouts (once per 32.4 seconds). The OBC should be patched to examine the ACIS bi-levels. It should safe the instruments if (a) RADMON is enabled, and (b) the bi-level channels (1STAT3ST-1STAT0ST) have the LED_BOOT_SPARE1 values (1, 1, 0, 1).

Applicable Reports/Requests:

Test Results:

Replaced Functions:

EventExposure::copyExpEnd
Leds::show

Command Impact:

The default filter parameters can be overridden by sending single writeBep command to ACIS to change the contents of the TXinit structure, whose address will depend on the ACIS flight software patch level (e.g., 0x8003dc30 in the current level E-F-G version). The command

```
write 0 0x8003dc30 {
  0
}
```

will, for instance, suspend the threshold crossing filter, and

```
write 0 0x8003dc30 {
  5
}
```

will turn it on again with an integration time of 5 minutes.

After a trigger, the bi-levels are not reset until Leds::show() is called when a science run is not in process. In the unlikely event that there is less than 64 seconds between the end of the triggering run and the start of the next, the bi-levels will continue to report LED_BOOT_SPARE1. This can be prevented by issuing a writeBep command to clear the counters:

```
  write 0 0x8003dc90 {
    0 0
  }
```

prior to the second startScience.

In normal operation, most science runs can be conducted with txings enabled, but exceptionally bright targets observed by few CCDs may lead to false triggers. It might be best to disable txings for short runs where the risk of radiation damage is small, or turn on additional CCDs for longer runs to reduce the likelihood of a false trigger. To change the trigger parameters for the next science run only, a writeBep command should update the fields in TXnext rather than TXinit, and this must be done before the science run has started to report events. In the current level E-F-G version, TXnext is located at 0x8003dc50.

Telemetry Impact:

When a threshold crossing trigger occurs, triggerRadmon() commands the BEPs memory manager to write a bepReadReply packet to telemetry, reporting the contents of the TX and tx structures. If this action is blocked for any reason, a SWSTAT_CMDECHO_DROPPED event will be reported in software housekeeping.

The current version of the patch reports bepReadReply packets with a formatTag of TTAG_READ_BEP. If this causes confusion, a new TlmFormatTag value could be defined, but the CXC Data System would need to be reconfigured to handle it. Similarly, if SWSTAT_CMDECHO_DROPPED is confusing, a new SwStatistic value could be defined.

Science Impact:

None

```
======================================================================

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.bcmd
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:


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 Continous 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: teignore

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

Conflicts:
Depends On:
Size:        36 bytes

Bcmd File:   opt_teignore.bcmd
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:


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: printswhouse

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

Conflicts:
Depends On:  tlmio
Size:        7240 bytes

Bcmd File:   opt_printswhouse.bcmd
Pkts File:   opt_printswhouse.pkts

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



Applicable Reports/Requests:
    TOOL-PENDING

Test Results:


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: deahktrip

Part Number: 36-58030.34
Version:     A
SCO:         none
Environment: flight

Conflicts:
Depends On:
Size:        1940 bytes

Bcmd File:   opt_deahktrip.bcmd
Pkts File:   opt_deahktrip.pkts

Description:



Applicable Reports/Requests:


Test Results:


Replaced Functions:
    Tf_Dea_Housekeeping_Data::append_Entries


Command Impact:
    To update the ’ndhk’ block that defines the limits of each of the DEA
    housekeeping channels that this patch puts under surveillance, upload
    the following command packet. The values shown are the defaults.

      write ‘n’ 0x8003dd20 {
        2

        2
        3600
        1010
        12
        1
        2060 4096
        0    0
        0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
        2289 4096 0
      }

    The starting address of the block, 0x8003dd20, may vary with the patch
    release. This value is appropriate for release GHI.
```

The patch replaces Tf_Dea_Housekeeping_Data::append_Entries() which is called to handle each DEA housekeeping value that has been requested by the housekeeping task. It defines a new class:

```
class Test_Tf_Dea_Housekeeping_Data : public Tf_Dea_Housekeeping_Data {
public:
  Test_Tf_Dea_Housekeeping_Data() : Tf_Dea_Housekeeping_Data() {};
  virtual void append_Entries(unsigned Ccd_Id, unsigned Query_Id, unsigned Value);
};
```

and a static ‘ndhk’ structure:

```
typedef struct {
  unsigned low;            // low DN limit value
  unsigned high;           // high DN limit value
  unsigned count;          // count of consecutive trips
} NDHK_VAL;

struct {                   // static channel limit table
  unsigned state;          // NDHK_{TRIP,HALT,NBLV,TEST}
  unsigned size;           // number of channels used in lim array
  unsigned min;            // index of lowest channel id
  unsigned lowvalid;       // lowest valid DN value (red high)
  unsigned highvalid;      // highest valid DN value (red low)
  unsigned tick1;          // bepTickCounter of first tripped packet
  unsigned tick2;          // bepTickCounter of second tripped packet
  unsigned spare;          // for debugging purposes
  NDHK_VAL lim[NDHKT];     // red-high, red-low values
} ndhk;                    // see above for initial ‘ndhk’ values
```

Note that the higher the DN value, the colder the physical temperature. The patch inserts the following code into append_Entries():

```
  // Check that we’re not in a triggered state and channel is Board 11/12
  if ((ndhk.state & NDHK_TRIP) == 0 && Ccd_Id == 10 && ndhk.size > 0) {
    int ii = Query_Id-ndhk.base;
    // Execute if this is a desired channel
    if (ii >= 0 && ii < ndhk.size && ii < NDHKT) {
      // Check if the value violates a limit
      if (ndhk.state & NDHK_TEST) {
        ndhk.state |= NDHK_TRIP;
      } else if ((Value > ndhk.lowvalid && Value <= ndhk.lim[i i].low) ||
                 (Value < ndhk.highvalid && Value >= ndhk.lim[ii] .high)) {
        // Increment the counter and trip if over sample limit
        if (++ndhk.lim[ii].count >= ndhk.sample) {
          ndhk.state |= NDHK_TRIP;
        }
      } else {
        ndhk.lim[ii].count = 0;
      }
    }
  }
```

Once the algorithm has "tripped", it compares the value of the BEP interrupt timer (in units of ~0.1 seconds) against the values of ndhk.tick1 and ndhk.tick2 to select three times:

1. When the alert is first triggered. If NDHK_HALT is set, any science run in progress is immediately halted along with the biasthief task, if running.

2. While filling the next deaHousekeepingData packet after the one in which the alert is first triggered. It writes the 47-word ndhk block into a bepReadReply packet. If NDHK_HALT is set, all FEPs and video boards are powered down.

3. While filling the deaHousekeepingData packet that is more that ndhk.delay seconds after the alert is first triggered. Up until this time, the software bilevels ‘1STAT3ST’ through ‘1STAT0ST’ will be set to ’1110’ (14) unless NDHK_BLVL is set. After this time, NDHK_TRIP will be cleared and tick1 and tick2 zeroed.

Since ACIS bilevels are also rewritten at 64-second intervals by the SoftwareHousekeeper task, they will switch between the trigger values and the usual values (0-12 and 15). If the ‘txings’ patch is also active and triggered, it will reset the bilevels to 13 every 64 seconds, so if ‘deahktrip’ is also triggered, the bilevels will switch between 13 and 14. We leave it to the OBC to figure out what to do in this circumstance.

Telemetry Impact:

If any of the binary values of the selected housekeeping channels lies within the range the minimum and maximum valid channel values and outside the range of the minimum and maximum non-trip values, the patch can terminate the current science run with a terminationCode of 17, power down the FEPs and video boards, and set the 4-bit software bilevel field to ‘LED_BOOT_SPARE2’ (14). it also writes the 47-word ‘ndhk’ block to a bepReadReply packet with a commandId of ndhk.cmdid (default 1010).

Science Impact:

If the NDHK_HALT flag is set, the component temperature alert will cause the remainder of the science run to be lost. However, if the algorithm has been ’reset’ after ndhk.delay, and the OBC hasn’t reacted by halting the stored science commands, the following observation should run as normal.

==

Patch Name: smtimedlookup

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

Conflicts:
Depends On:
Size: 3712 bytes

Bcmd File: opt_smtimedlookup.bcmd
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::setupCti1
```

instructs the linker to insert the address of the setupCti1() method of the Test2_SmTimedExposure class into slot 4 of the smTimedLookupMode table, so that setupCti1() will be called when FepMode == 4.

Applicable Reports/Requests:

Test Results:

Replaced Functions:

SmTimedExposure::setupFepBlock
SmTimedExposure::terminate
SmTimedExposure::setupProcess

Command Impact:

None.

```
Telemetry Impact:
    None.


Science Impact:
    None.
```

```
======================================================================

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.bcmd
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 fepManager.loadRunProgram from Test3_SmTimedExposure::setupCti1Fep.
    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:


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

```
/* ============================================================
 *
 * $$Source: /nfs/acis/h3/acisfs/configcntl/patches/buscrash/buscrash.C,v $$
 *
 * 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
 *
 * $$Log: buscrash.C,v $
 * $Revision 1.6  2016/03/17 19:18:23  pgf
 * $Force BoolFalse return from pollBiasComplete() when all FEPs powered off
 * $
 * $Revision 1.5  2016/03/11 20:36:36  pgf
 * $Replace pollBiasComplete() to abort science run if all FEPs powered down
 * $
 * $Revision 1.4  2007/08/14 16:09:36  pgf
 * $Add friend statement
 * $
 * $Revision 1.3  2007/07/14 22:48:29  pgf
 * $Change method from static to virtual
 * $
 * $Revision 1.2  2007/04/18 21:10:57  pgf
 * $Call fepManager.isEnabled to prevent bus crash.
 * $
 * $Revision 1.1  2007/04/17 18:52:35  pgf
 * $Initial version.
 * $$
 *
 * ============================================================ */

#include <stdio.h>
#include "acis_h/interface.h"
#include "filesprotocols/fepmanager.H"
#include "filesswhouse/swhousekeeper.H"
#define protected public
#include "filesscience/sciencemode.H"
#undef protected

class Test_FepManager
{
public:
  void loadBadPixel(FepId fepid, unsigned row, unsigned col);
  Boolean pollBiasComplete();
  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);
    }
}

Boolean Test_FepManager::pollBiasComplete()
{
    DebugProbe probe;

    Boolean retval = BoolFalse;  // Assume not ready
    retval = fepManager.pollOperationComplete();

    if (retval == BoolTrue && fepManager.anyEnabled() == BoolFalse) {
    Task * curTask = taskManager.queryCurrentTask();
    if (curTask != 0) {
        curTask->notify(ScienceMode::EV_SM_ABORT_RUN);
        retval = BoolFalse;
    }
    }

    // ---- Return BoolTrue if bias ready, else BoolFalse ----
    return retval;
}
```

```
# ---------------------------------------------------
#
# $$Source: /nfs/acis/h3/acisfs/configcntl/patches/buscrash/buscrash.pkg,v $$
#
# Bias Timing Patch Specification File
#
# Version:
#   The part number and version of this release are
#   described below under the "partnumber" and
#   "version" keywords.
#
# Description:
#   This is a Patch Specification File. The detailed
#   documentation for this file is provided after the
#   NOTES: keyword below.
#
# Format:
#   This is a line-oriented file.
#
#   Comments are indicated by a leading ’#’.
#   Blank lines are ignored.
#
#   Keyword pairs are assigned as "keyword = value",
#   where:
#   ident          - The CVS/RCS identification string
#       partnumber     - The partnumber of the patch
#       version        - The release version of the patch
#       environment    - Either "flight", or "engineering"
#
#   Lists of information consist of the list name
#   followed by the next item to be placed into the
#   list. The lists are:
#       source <name> <partext> - This specifies a source file
#                       which should be reviewed when
#                       the package is released. At this time,
#                       these entries are only used for documentation
#                       purposes and aren’t used to build run-time
#                       products. The run-time products are produced
#                       by the .mak file. <partext> refers to the part
#                       number extension of the file relative to the
#                       base part number of the patch.
#
#   object <name>   - This specifies an object file
#                       which must be built and linked for
#                       the patch, where <name> is the name
#                       of the file to be built and linked with.
#
#   func <oldname>  <newname> -
#                       This specifies a function
#                       which must be overridden for the
#                       patch to work. <oldname> is the
#                       old subroutine name, and <newname>
#                       is the new subroutine which replaces
#                       the old.
#
#       bcmd <name>     - This specifies a literal bcmd input
#                       file which must be built and included
#                       in the load for the patch. These typically
#                       hold independent specially built patches
#                       which do not have to be linked with the
#                       reset of the system in order to work, such
#                       as inline patches.
#
#       spr <number>    - This identifies a Software Problem Report
#                       which is addressed by this patch.
#
#       ser <number>    - This identifies a Software Enhancement Request
#                       which is addressed by this patch.
#
#       tool <number>   - This identifies a Software Diagnostic Tool
#                       which is addressed by this patch.
#
#       test <name> <subdir> <command line> -
#                       This specifies a test to run on the package.
#                       <name> indicates the test name, <subdir> is
#                       the subdirectory of the package that the test
#                       should be run in, and <command line> is the command
#                       to execute to run the test. All tests shall
#                       print either "PASS" or "FAIL", depending on the
#                       result of the tests. Incomplete tests should always
#                       print "FAIL".
#
#   At the end of the file, the ’NOTES:’ keyword
#   delimits the notes section of the file. All lines
#   following this keyword line are treated as the
#   release notes for this patch. These notes should be
#   included in all patch releases and option suite documentation.
#
#   The notes sections are delimited by section keywords. Any text
#   from the start of the NOTES section until the first keyword is
#   treated as a general description of the patch.
#
#      COMMAND IMPACT:  - This section describes the impact of the patch
#                       on commanding of the instrument.
#
#      TELEMETRY IMPACT: - This section describes the impact of the patch
#                        on the telemetry produced by the instrument.
#
#      SCIENCE IMPACT:   - This sections describes the impact of the patch
#                        on the science data produced by the instrument.
#
#      :END              - Delimits the end of the notes section
#
# Version Log:
# $$Log: buscrash.pkg,v $
# $Revision 1.11  2018/06/29 19:48:19  pgf
# $Update with date changes
# $
# $Revision 1.10  2018/06/29 18:56:44  pgf
# $Document release B
# $
# $Revision 1.9  2016/04/02 16:26:05  pgf
# $Fix typo
# $
# $Revision 1.8  2016/03/18 17:02:48  pgf
# $Add second test to reproduce bug in standard patch F
# $
# $Revision 1.7  2016/03/11 21:08:05  pgf
# $Replace pollBiasComplete() to abort science run if all FEPs powered down
# $
# $Revision 1.6  2010/01/14 18:57:36  pgf
# $Fix typo.
# $
# $Revision 1.5  2007/08/14 16:57:39  pgf
# $Released as part of Standard Patch C
# $
# $Revision 1.4  2007/08/14 16:54:28  pgf
# $remove bcmd from package
# $
# $Revision 1.3  2007/07/16 19:18:27  pgf
# $Add buscrash.bcmd to list of buildables.
# $
# $Revision 1.2  2007/07/11 15:42:05  pgf
# $Review versions.
# $
# $Revision 1.1  2007/04/17 18:52:36  pgf
# $Initial version.
# $$
# --------------------------------------------------

# Identification Information
ident = $$Id: buscrash.pkg,v 1.11 2018/06/29 19:48:19 pgf Exp $$

partnumber  = 36-58030.30
version     = B
environment = flight
eco         = 36-1051
reason      = Cleanup comments, release tests

# Release history information
approval A 36-1034 PGF 08/09/2007 Released
approval B 36-1051 RFG 06/29/2018 Released

# Product and source file information
object buscrash.o
func   FepManager::loadBadPixel Test_FepManager::loadBadPixel
func   FepManager::pollBiasComplete Test_FepManager::pollBiasComplete
source buscrash.pkg 01
source buscrash.mak 02
source buscrash.C   03
docref eco-1051.pdf

# Test information
test reproduce testsuite/bug-hw make ACISSERVER=$(ACISSERVER) TOOLS=$(TOOLS) PATCHDIR=$
(PATCHDIR)
test reproduce2 testsuite/bug-hw    make SCRIPT=runtest2 ACISSERVER=$(ACISSERVER) TOOLS
=$(TOOLS) PATCHDIR=$(PATCHDIR)
test fix testsuite/fix-hw make ACISSERVER=$(ACISSERVER) TOOLS=$(TOOLS) PATCHDIR=$(PATCH
DIR)

# Initiating action information
spr    151

#--------------------------------------------------
NOTES:

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). Release A
of this fix interacted badly with the buscrash2 patch in a manner that
could prevent the science run from termination. This was corrected in
release B of buscrash.

COMMAND IMPACT:
None.

TELEMETRY IMPACT:
None.

SCIENCE IMPACT:
None.
```

```
#! /bin/sh

genObjectImage -e $* <<!
    Rows        = 1024
    Columns     = 256
    Mode        = ABCD
    Overclocks  = 16
    Seed        = 12345678
    Noop        = 4 before Oclks
    Noop        = 0 before HSYNC
    Noop        = 8 after  HSYNC
    Noop        = 4104 before VSYNC
    Noop        = 3 after VSYNC

    Begin Node  = A
      Bias      = 210
      dBias     = 0
      OverClock = 200
      dOverClock    = 0
    End Node    = A

    Begin Node  = B
      Bias      = 310
      dBias     = 0
      OverClock = 300
      dOverClock    = 0
    End Node    = B

    Begin Node  = C
      Bias      = 410
      dBias     = 0
      OverClock = 400
      dOverClock    = 0
    End Node    = C

    Begin Node  = D
      Bias      = 510
      dBias     = 0
      OverClock = 500
      dOverClock    = 0
    End Node    = D
!
```

```
#! /bin/env expect

puts "Welcome to buscrash/testsuite/bug-hw/runtest2.tcl"

# ---- Split off the command arguments ----
lassign $argv basedir tools patchdir

# ---- 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 } {
    expect { -timeout $secs 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.bcmd\
         $basedir/$tools/share/opt_printswhouse.bcmd\
         $basedir/$tools/share/opt_dearepl.bcmd\
         standardF.bcmd"
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"

puts "\n# Starting test\n"

send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 "start science run"
system make bias

expect {
    -timeout 360
    -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
expect {
    -timeout 60
    -re "bepStartupMessage.*\[\r\n]*" {
    fail "Bus crash reproduced"
    }
    -re "scienceReport.*\[\r\n]*" {
    fail "Science run ends without bus crash"
    }
    timeout {
    pass "No crash or stopScience"
    }
}

puts "Done"
```

```
#! /bin/env expect

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

# ---- Split off the command arguments ----
lassign $argv basedir tools patchdir

# ---- 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 } {
    expect { -timeout $secs 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.bcmd\
         $basedir/$tools/share/opt_printswhouse.bcmd\
         $basedir/$tools/share/opt_dearepl.bcmd"
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 "\n# Starting test\n"

send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 "start science run"

expect {
    -timeout 360
    -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
expect {
    -timeout 360
    -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 ----
lassign $argv basedir tools patchdir

# ---- 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 } {
    expect { -timeout $secs 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.bcmd\
         $basedir/$tools/share/opt_printswhouse.bcmd\
         $basedir/$tools/share/opt_dearepl.bcmd\
         ./buscrash.bcmd"
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 "\n# Starting test\n"

send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 "start science run"
expect {
    -timeout 360
    -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
expect {
    -timeout 360
    -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: $Source: /nfs/acis/h3/acisfs/configcntl/patches/deahktrip/deahktrip.C,v $
 *
 *  Name:   deahktrip
 *
 *  Author:  Peter G. Ford <pgf@space.mit.edu>
 *
 *  Description:
 *  Monitor the DPA thermal component readouts from Board 11/12. If any
 *  exceed specified thresholds, set the ACIS bilevels to LED_BOOT_SPARE2
 *  to signal to the OBC that ACIS should be safed.

 *  The patch replaces Tf_Dea_Housekeeping_Data::append_Entries() with
 *  a copy that tests the calling arguments and checks the values of
 *  DPA thermal channels, calling bepReg.showLeds(LED_BOOT_SPARE2) when
 *  a channel is out of limits.
 *
 * References:
 *   Refer to deahktrip.pages
 *
 *  Log:    $Log: deahktrip.C,v $
 *  Log:    Revision 1.5  2018/07/12 18:37:07  pgf
 *  Log:    Update with correct red alarm limits
 *  Log:
 *  Log:    Revision 1.4  2018/05/14 19:20:44  pgf
 *  Log:    Remove ’virtual’ attribute
 *  Log:
 *  Log:    Revision 1.3  2018/05/09 17:45:43  pgf
 *  Log:    Add swHouseKeeping packet when first tripped with NDHK_TEST
 *  Log:
 *  Log:    Revision 1.2  2018/05/02 20:45:22  pgf
 *  Log:    Add option to abort science run and power down FEP and video boards
 *  Log:
 * -------------------------------------------------------------------------- */
#ifndef Test_Tf_Dea_Housekeeping_Data_H
#define Test_Tf_Dea_Housekeeping_Data_H 1
#define private public
#define protected public
#include "filesscience/sciencemode.H"
#include "filesscience/sciencemanager.H"
#include "filessysconfig/sysconfigtable.H"
#include "fileswhouse/swhousekeeper.H"
#include "filesmemserver/memoryserver.H"
#undef private
#undef protected

#define NDHKT       (12)    // maximum number of channels checked
#define NDHK_NERR   (17)    // science run error code
#define NDHK_TRIP   (1)     // channel alarm tripped
#define NDHK_HALT   (2)     // halt science run, power down boards
#define NDHK_NBLV   (4)     // suppress report via bilevels
#define NDHK_TEST   (8)     // force alarm

typedef struct {
    unsigned low;           // low DN limit value
    unsigned high;          // high DN limit value
    unsigned count;         // count of consecutive trips
} NDHK_VAL;

struct {                    // static channel limit table
    unsigned state;         // state flags
    unsigned sample;        // conditioning sample size
    unsigned delay;         // seconds before resuming testing
    unsigned cmdid;         // commandId for bepReadReply packet
    unsigned size;          // number of channels used in lim array
    unsigned base;          // index of lowest channel id
    unsigned lowvalid;      // lowest valid DN value (red high)
    unsigned highvalid;     // highest valid DN value (red low)
    unsigned tick1;         // bepTickCounter of first tripped packet
    unsigned tick2;         // bepTickCounter of second tripped packet
    unsigned spare;         // for debugging purposes
    NDHK_VAL lim[NDHKT];    // lowest,highest,count channel limit values
} ndhk = { 0, 2, 3600, 1010, 12, 1, 2060, 4096, 0, 0, 0,
  {
    { 2297, 4096, 0 },  /* BEP_PCB */       { 2314, 4096, 0 },  /* BEP_OSC */
    { 2266, 4096, 0 },  /* FEP0_MONG */     { 2289, 4096, 0 },  /* FEP0_PCB */
    { 2274, 4096, 0 },  /* FEP0_ACTEL */    { 2281, 4096, 0 },  /* FEP0_RAM */
    { 2306, 4096, 0 },  /* FEP0_FB */       { 2259, 4096, 0 },  /* FEP1_MONG */
    { 2281, 4096, 0 },  /* FEP1_PCB */      { 2266, 4096, 0 },  /* FEP1_ACTEL */
    { 2266, 4096, 0 },  /* FEP1_RAM */      { 2306, 4096, 0 },  /* FEP1_FB */
  }
};

// Temperatures vs DN values
// Cold>=3880, -10C=3313, 0C=3098, 6.5C=2953, 45C=2332, 50C=2289, Hot<=2060

// ------------------------------------------------------------------------------
// Class Test_Tf_Dea_Housekeeping_Data, friend of Tf_Dea_Housekeeping_Data
// ------------------------------------------------------------------------------

class Test_Tf_Dea_Housekeeping_Data : public Tf_Dea_Housekeeping_Data
{
public:
    void append_Entries(unsigned Ccd_Id, unsigned Query_Id, unsigned Value);
};

// ------------------------------------------------------------------------------
// Test_Tf_Dea_Housekeeping_Data -- where all the work is done
// ------------------------------------------------------------------------------

void Test_Tf_Dea_Housekeeping_Data::append_Entries(unsigned Ccd_Id,
    unsigned Query_Id, unsigned Value)
{
    DebugProbe probe;

    // Check that we’re not in a triggered state and channel is Board 11/12
    if ((ndhk.state & NDHK_TRIP) == 0 && Ccd_Id == 10 && ndhk.size > 0) {
        int ii = Query_Id-ndhk.base;
        // Execute if this is a desired channel
        if (ii >= 0 && ii < ndhk.size && ii < NDHKT) {
            // Check if the value violates a limit
            if (ndhk.state & NDHK_TEST) {
                ndhk.state |= NDHK_TRIP;
            } else if ((Value > ndhk.lowvalid && Value <= ndhk.lim[ii].low) ||
                (Value < ndhk.highvalid && Value >= ndhk.lim[ii].high)) {
                // Increment the counter and trip if over sample limit
                if (++ndhk.lim[ii].count >= ndhk.sample) {
                    ndhk.state |= NDHK_TRIP;
                }
            } else {
                ndhk.lim[ii].count = 0;
            }
        }
    }

    // Check for trigger
    if (ndhk.state & NDHK_TRIP) {
        unsigned tick = (getBufPtr())[4];   // get bepTickCounter from TlmForm
        if ((ndhk.state & NDHK_NBLV) == 0) {
            // set the software bilevels
            bepReg.showLeds(LED_BOOT_SPARE2);
        }
        if (ndhk.tick1 == 0) {
            // execute once in same housekeeping packet as trigger
            ndhk.tick1 = tick;
            ndhk.tick2 = 0;
            ndhk.spare = (unsigned)scienceManager.currentMode;
            // If science mode running, stop it and any associated BiasThief
            if ((ndhk.state & NDHK_HALT) && scienceManager.currentMode != 0) {
                *(unsigned *)&scienceManager.currentMode->termReason = NDHK_NERR;
                scienceManager.notify(ScienceMode::EV_SM_ABORT_RUN);
            }
            if (ndhk.state & NDHK_TEST) {
                swHousekeeper.report(SWSTAT_CMDECHO_DROPPED, ndhk.cmdid);
                ndhk.state &= ~NDHK_TEST;
            }
        } else if (tick != ndhk.tick1 && ndhk.tick2 == 0) {
            // execute once in next housekeeping packet following trigger
            ndhk.tick2 = tick;
            if (ndhk.state & NDHK_HALT) {
                // Power down the boards
                sysConfigTable.changeEntry(SYSSET_DEA_POWER, 0);
                sysConfigTable.changeEntry(SYSSET_FEP_POWER, 0);
            }
            // Report the power-down
            unsigned *a = (unsigned *)&ndhk;
            unsigned w = sizeof(ndhk)/sizeof(unsigned);
            CmdResult rc = memoryServer.readBep(ndhk.cmdid, a, w, TTAG_READ_BEP);
            if (rc != CMDRESULT_OK) {
                swHousekeeper.report(SWSTAT_CMDECHO_DROPPED, ndhk.cmdid);
            }
        } else if (tick != ndhk.tick1 && tick != ndhk.tick2 &&
            tick > ndhk.tick1+ndhk.delay*Acis::TICKS_PER_SECOND) {
            // execute once at least ndhk.delay seconds after trigger packet
            ndhk.state &= ~NDHK_TRIP;
            ndhk.tick1 = ndhk.tick2 = 0;
            // Clear the channel counters
            for (int ii = 0; ii < NDHKT; ii++) {
                ndhk.lim[ii].count = 0;
            }
        }
    }

    // ---- Range check index argument ----
    ASSERT(isFull() == BoolFalse);

    // ----- Value range checks ----
    ASSERT(Ccd_Id <= 10);

    // ---- Ccd_Id :: Offset = 160, Width = 8 ----
    appendField (Ccd_Id, 160, 8, 0x0ff, _appended, 32);

    // ----- Value range checks ----
    ASSERT(Query_Id <= 255);

    // ---- Query_Id :: Offset = 168, Width = 8 ----
    appendField (Query_Id, 168, 8, 0x0ff, _appended, 32);

    // ----- Value range checks ----
    ASSERT(Value <= 65535);
    // ---- Value :: Offset = 176, Width = 16 ----
    appendField (Value, 176, 16, 0x0ffff, _appended, 32);

    _appended++;

    return;
};

// ------------------------------------------------------------------------------
// End of deahktrip patch
// ------------------------------------------------------------------------------
#endif /* Test_Tf_Dea_Housekeeping_Data_H */
```

```
# --------------------------------------------------
#
# $Source: /nfs/acis/h3/acisfs/configcntl/patches/deahktrip/deahktrip.pkg,v $
#
# Version:
#   The part number and version of this release are
#   described below under the "partnumber" and
#   "version" keywords.
#
# Description:
#   This is a Patch Specification File. The detailed
#   documentation for this file is provided after the
#   NOTES: keyword below.
#
# Format:
#   This is a line-oriented file.
#
#   Comments are indicated by a leading ’#’.
#   Blank lines are ignored.
#
#   Keyword pairs are assigned as "keyword = value",
#   where:
#       ident          - The CVS/RCS identification string
#       partnumber     - The partnumber of the patch
#       version        - The release version of the patch
#       environment    - Either "flight", or "engineering"
#       sco            - The software change order of the released patch
#       reason         - Short reason for this version
#
#   Lists of information consist of the list name
#   followed by the next item to be placed into the
#   list. The lists are:
#
#       source <name> <partext> - This specifies a source file
#                       which should be reviewed when
#                       the package is released. At this time,
#                       these entries are only used for documentation
#                       purposes and aren’t used to build run-time
#                       products. The run-time products are produced
#                       by the .mak file. <partext> refers to the part
#                       number extension of the file relative to the
#                       base part number of the patch.
#
#       object <name>   - This specifies an object file
#                       which must be built and linked for
#                       the patch, where <name> is the name
#                       of the file to be built and linked with.
#
#       func <oldname> <newname> -
#                       This specifies a function
#                       which must be overridden for the
#                       patch to work. <oldname> is the
#                       old subroutine name, and <newname>
#                       is the new subroutine which replaces
#                       the old.
#
#       bcmd <name>     - This specifies a literal bcmd input
#                       file which must be built and included
#                       in the load for the patch. These typically
#                       hold independent specially built patches
#                       which do not have to be linked with the
#                       reset of the system in order to work, such
#                       as inline patches.
#
```

```
#       spr <number>    - This identifies a Software Problem Report
#                       which is addressed by this patch.
#
#       ser <number>    - This identifies a Software Enhancement Request
#                       which is addressed by this patch.
#
#       tool <number>   - This identifies a Software Diagnostic Tool
#                       which is addressed by this patch.
#
#       docref <number> - This identifies an existing design or
#                       requirements reference which is pertinent
#                       to the patch.
#
#       approval <rev> <sco> <signer> <date> <text>
#                       - Sign-off on a previous release
#
#   At the end of the file, the ’NOTES:’ keyword
#   delimits the notes section of the file. All lines
#   following this keyword line are treated as the
#   release notes for this patch. These notes should be
#   included in all patch releases and option suite documentation.
#
#   The notes sections are delimited by section keywords. Any text
#   from the start of the NOTES section until the first keyword is
#   treated as a general description of the patch.
#
#      COMMAND IMPACT:  - This section describes the impact of the patch
#                       on commanding of the instrument.
#
#      TELEMETRY IMPACT: - This section describes the impact of the patch
#                        on the telemetry produced by the instrument.
#
#      SCIENCE IMPACT:   - This sections describes the impact of the patch
#                        on the science data produced by the instrument.
#
#      :END              - Delimits the end of the notes section
#
# Version Log:
# $Log: deahktrip.pkg,v $
# Revision 1.4  2018/06/29 19:47:45  pgf
# Documentation for release A
#
# Revision 1.3  2018/05/14 19:23:01  pgf
# Update patch description and impact
#
# Revision 1.2  2018/05/02 20:45:22  pgf
# Add option to abort science run and power down FEP and video boards
#
# Revision 1.1  2018/04/02 17:33:12  pgf
# Initial version
#
# --------------------------------------------------

# Identification Information
ident = $Id: deahktrip.pkg,v 1.4 2018/06/29 19:47:45 pgf Exp $

partnumber  = 36-58030.34
version     = A
environment = flight
sco         = none
reason      = New Patch

# Release history information
approval A 36-1052 RFG 06/29/18 Initial letter release
```

```
# Product and source file information

object deahktrip.o
func Tf_Dea_Housekeeping_Data::append_Entries Test_Tf_Dea_Housekeeping_Data::append_Ent
ries
source deahktrip.pkg  01
source deahktrip.mak  02
source standalone.mak 03
source deahktrip.C    04
docref eco-1052.pdf

# Test information
test smoke testsuite/smoke make ACISSERVER=$(ACISSERVER) TOOLS=$(TOOLS) PATCHDIR=$(PATC
HDIR) ACISTOOLSDIR=/nfs/acis/h4/tools ACISTTMFILE=acisEUbilevels.ttm

# Initiating action information
ser  None
spr  None

#---------------------------------------------------
NOTES:

COMMAND IMPACT:
To update the ’ndhk’ block that defines the limits of each of the DEA
housekeeping channels that this patch puts under surveillance, upload
the following command packet. The values shown are the defaults.

  write ‘n’ 0x8003dd20 {
    2            # state flags: =1 tripped, =2 halt science,
                 # =4 don’t change bilevels, =8 test with EU
    2            # sample size
    3600         # reset alarm after delay in seconds
    1010         # commandId for error messages
    12           # number of channels to be tested
    1            # starting channelId value
    2060 4096    # minimum and maximum valid channel values
    0    0       # BEP timer for alarm and power-down
    0            # spare for testing
    2289 4096 0 # BEP_PCB minimum, maximum, samples non-trip values
    2289 4096 0 # BEP_OSC minimum, maximum, samples non-trip values
    2289 4096 0 # FEP0_MONG minimum, maximum, samples non-trip values
    2289 4096 0 # FEP0_PCB minimum, maximum, samples non-trip values
    2289 4096 0 # FEP0_ACTEL minimum, maximum, samples non-trip values
    2289 4096 0 # FEP0_RAM minimum, maximum, samples non-trip values
    2289 4096 0 # FEP0_FB minimum, maximum, samples non-trip values
    2289 4096 0 # FEP1_MONG minimum, maximum, samples non-trip values
    2289 4096 0 # FEP1_PCB minimum, maximum, samples non-trip values
    2289 4096 0 # FEP1_ACTEL minimum, maximum, samples non-trip values
    2289 4096 0 # FEP1_RAM minimum, maximum, samples non-trip values
    2289 4096 0 # FEP1_FB minimum, maximum, samples non-trip values
  }

The starting address of the block, 0x8003dd20, may vary with the patch
release. This value is appropriate for release GHI.

The patch replaces Tf_Dea_Housekeeping_Data::append_Entries() which is
called to handle each DEA housekeeping value that has been requested by
the housekeeping task. It defines a new class:

class Test_Tf_Dea_Housekeeping_Data : public Tf_Dea_Housekeeping_Data {
public:
  Test_Tf_Dea_Housekeeping_Data() : Tf_Dea_Housekeeping_Data() {};
  virtual void append_Entries(unsigned Ccd_Id, unsigned Query_Id, unsigned Value);
};
```

and a static ‘ndhk’ structure:

```
#define NDHKT      12     // maximum number of channels to check
#define NDHK_NERR  17     // science run error code
#define NDHK_TRIP  1      // channel alarm tripped
#define NDHK_HALT  2      // halt science run, power down boards
#define NDHK_NBLV  4      // suppress report via bilevels
#define NDHK_TEST  8      // force alarm

typedef struct {
  unsigned low;           // low DN limit value
  unsigned high;          // high DN limit value
  unsigned count;         // count of consecutive trips
} NDHK_VAL;

struct {                  // static channel limit table
  unsigned state;         // NDHK_{TRIP,HALT,NBLV,TEST}
  unsigned size;          // number of channels used in lim array
  unsigned min;           // index of lowest channel id
  unsigned lowvalid;      // lowest valid DN value (red high)
  unsigned highvalid;     // highest valid DN value (red low)
  unsigned tick1;         // bepTickCounter of first tripped packet
  unsigned tick2;         // bepTickCounter of second tripped packet
  unsigned spare;         // for debugging purposes
  NDHK_VAL lim[NDHKT];    // red-high, red-low values
} ndhk;                   // see above for initial ‘ndhk’ values
```

Note that the higher the DN value, the colder the physical temperature.
The patch inserts the following code into append_Entries():

```
  // Check that we’re not in a triggered state and channel is Board 11/12
  if ((ndhk.state & NDHK_TRIP) == 0 && Ccd_Id == 10 && ndhk.size > 0) {
    int ii = Query_Id-ndhk.base;
    // Execute if this is a desired channel
    if (ii >= 0 && ii < ndhk.size && ii < NDHKT) {
      // Check if the value violates a limit
      if (ndhk.state & NDHK_TEST) {
        ndhk.state |= NDHK_TRIP;
      } else if ((Value > ndhk.lowvalid && Value <= ndhk.lim[i i].low) ||
                 (Value < ndhk.highvalid && Value >= ndhk.lim[ii] .high)) {
        // Increment the counter and trip if over sample limit
        if (++ndhk.lim[ii].count >= ndhk.sample) {
          ndhk.state |= NDHK_TRIP;
        }
      } else {
        ndhk.lim[ii].count = 0;
      }
    }
  }
```

Once the algorithm has "tripped", it compares the value of the BEP interrupt timer (in units of ~0.1 seconds) against the values of ndhk.tick1 and ndhk.tick2 to select three times:

1. When the alert is first triggered. If NDHK_HALT is set, any science run in progress is immediately halted along with the biasthief task, if running.

2. While filling the next deaHousekeepingData packet after the one in which the alert is first triggered. It writes the 47-word ndhk block into a bepReadReply packet. If NDHK_HALT is set, all FEPs and video boards are powered down.

```
3. While filling the deaHousekeepingData packet that is more that
   ndhk.delay seconds after the alert is first triggered. Up until this
   time, the software bilevels ‘1STAT3ST’ through ‘1STAT0ST’ will be set
   to ’1110’ (14) unless NDHK_BLVL is set. After this time, NDHK_TRIP
   will be cleared and tick1 and tick2 zeroed.

Since ACIS bilevels are also rewritten at 64-second intervals by the
SoftwareHousekeeper task, they will switch between the trigger values
and the usual values (0-12 and 15). If the ‘txings’ patch is also active
and triggered, it will reset the bilevels to 13 every 64 seconds, so if
‘deahktrip’ is also triggered, the bilevels will switch between 13 and
14. We leave it to the OBC to figure out what to do in this circumstance.

TELEMETRY IMPACT:
If any of the binary values of the selected housekeeping channels
lies within the range the minimum and maximum valid channel values and
outside the range of the minimum and maximum non-trip values, the patch
can terminate the current science run with a terminationCode of 17,
power down the FEPs and video boards, and set the 4-bit software bilevel
field to ‘LED_BOOT_SPARE2’ (14). it also writes the 47-word ‘ndhk’ block
to a bepReadReply packet with a commandId of ndhk.cmdid (default 1010).

SCIENCE IMPACT:
If the NDHK_HALT flag is set, the component temperature alert will cause
the remainder of the science run to be lost. However, if the algorithm has
been ’reset’ after ndhk.delay, and the OBC hasn’t reacted by halting the
stored science commands, the following observation should run as normal.

:END
```

```
#! /usr/bin/env perl
#
# $Source: /nfs/acis/h3/acisfs/configcntl/patches/deahktrip/testsuite/makebias.pl,v $
#

$rows = $ARGV[0] ? $ARGV[0] : 1024;
$noop = $ARGV[1] ne ’’ ? $ARGV[1] : 33743;

print <<EOF;
    Rows        = $rows
    Columns     = 256
    Mode        = ABCD
    Overclocks  = 16
    Seed        = 12345678
    Noop        = 4 before Oclks
    Noop        = 0 before HSYNC
    Noop        = 8 after  HSYNC
    Noop        = $noop before VSYNC
    Noop        = 3 after VSYNC

    Begin Node  = A
      Bias      = 210
      dBias     = 0
      OverClock = 200
      dOverClock    = 0
    End Node    = A

    Begin Node  = B
      Bias      = 310
      dBias     = 0
      OverClock = 300
      dOverClock    = 0
    End Node    = B

    Begin Node  = C
      Bias      = 410
      dBias     = 0
      OverClock = 400
      dOverClock    = 0
    End Node    = C

    Begin Node  = D
      Bias      = 510
      dBias     = 0
      OverClock = 500
      dOverClock    = 0
    End Node    = D
EOF

exit 0;
```

```
#! /usr/bin/env perl
#
# $Source: /nfs/acis/h3/acisfs/configcntl/patches/deahktrip/testsuite/makeimage.pl,v $
#

$rows = $ARGV[0] ? $ARGV[0] : 1024;
$noop = $ARGV[1] ne ’’ ? $ARGV[1] : 34626;
$incr = $ARGV[2] ? $ARGV[2] : 100;
$dim  = $ARGV[3] ? $argv[3] : 1;

print <<EOF;
    Rows        = $rows
    Columns     = 256
    Mode        = ABCD
    Overclocks  = 16
    Seed        = 12345678
    Noop        = 4 before Oclks
    Noop        = 0 before HSYNC
    Noop        = 8 after  HSYNC
    Noop        = $noop before VSYNC
    Noop        = 3 after VSYNC

    Begin Node  = A
      Bias      = 220
      dBias     = 0
      OverClock = 201
      dOverClock    = 0
    End Node    = A

    Begin Node  = B
      Bias      = 320
      dBias     = 0
      OverClock = 302
      dOverClock    = 0
    End Node    = B

    Begin Node  = C
      Bias      = 420
      dBias     = 0
      OverClock = 403
      dOverClock    = 0
    End Node    = C

    Begin Node  = D
      Bias      = 520
      dBias     = 0
      OverClock = 504
      dOverClock    = 0
    End Node    = D
EOF

$r1 = 2 * $dim + 1;
$c1 = 2 * $dim + 1;
$n = 1;

for ($r = $rows - $r1 - 1; $r > $r1; $r -= $incr) {
    for ($c = $c1; $c < $rows - $c1; $c += $incr) {
    local($v) = ’’;
    for $r2 (-$dim..$dim) {
        for $c2 (-$dim..$dim) {
        $v .= ($r2 || $c2) ? " 1" : " $n";
        }
    }
    print <<EOE;
    Begin Event = event_$n
      Rows      = $r1
      Columns   = $c1
      Value     =$v
    End Event   = event_$n

    event_$n $r $c
EOE
    $n = ($n < 4093) ? $n+1 : 1;
    }
}

exit 0;
```

```
#! /usr/bin/env expect
#
# $Source: /nfs/acis/h3/acisfs/configcntl/patches/deahktrip/testsuite/smoke/runtest.tcl
,v $
#
# Test of deahktrip patch
#

send_user "Welcome to deahktrip/testsuite/smoke/runtest.tcl\n"

# ---- Launch the command and telemetry server processes ----
lassign $argv basedir tools patchdir

# ---- Run Parameters ----
set ccd_list    {0 1 2 3 4 5}   ; # CCDs to assign to FEPs
set state   10      ; # Starting ndhk.state value
set datarate    50      ; # Measure of event rate
set delay   3600        ; # Delay in seconds until trip reset
set timeout 300         ; # Default timeout
set cmdId   1010        ; # commandId for bepReadReply
set pmode   0       ; # te faint 3x3 mode
set phist   0       ; # exposures per histogram (unused)
set ncmd    0       ; # commandId for send commands
set ndeahk  0       ; # deaHousekeepingData counter
set nbeprep 0       ; # bepReadReply counter
set nbilevel    0       ; # Bilevel alarm counter
set nscirep 0       ; # scienceReport counter

# ---- Embed the Procedure Library ----
source $basedir/$tools/lib/lib-exp/runtest_support.tcl
source ./aux.tcl

# ---- Start the Command Pipe ----
cmdspawn $basedir/$tools/bin/cmdclient $env(ACISSERVER)
set cmd_id $spawn_id

# ---- Start the Telemetry Pipe, passing -E. option to psci ----
spawn $basedir/$tools/bin/tlmclient $env(ACISSERVER) -E$env(ACISTTMFILE)
sleep 1
match_max 400

# ---- Locate address of ndhk strucure ----
lassign [exec grep {D ndhk} "./opt_deahktrip.map"] addr

# ---- Load Patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_printswhouse.bcmd\
         $basedir/$tools/share/opt_tlmio.bcmd\
         $basedir/$tools/share/opt_dearepl.bcmd\
         ./opt_deahktrip.bcmd"
warm_boot

# ---- Upload initial ndhk structure with red limits of 9/18/2017 ----
set ndhk "0 2 $delay $cmdId 12 1 2060 4096 0 0 0"
foreach ii {2297 2314 2266 2289 2274 2281 2306 2259 2281 2266 2266 2306} {
    append ndhk " $ii 4096 0"
}
send -i $cmd_id "write [incr ncmd] 0x$addr {\n$ndhk\n}\n"
command_echo 1 192 {initialize ndhk}

# ---- Power up FEPs and video boards ----
power_on_boards $ccd_list
set timeout 300
expect -re "SWSTAT_FEPMAN_ENDLOAD: 5\[\r\n]+" {} timeout { fail timeout }
```

```
# ---- Start DEA Housekeeping ----
send -i $cmd_id "load [incr ncmd] dea 4 {[deaHkPblock 10]}\n"
command_echo 1 13 "load dea"
send -i $cmd_id "start [incr ncmd] dea 4\n"
command_echo 1 18 "start dea housekeeping"

# ---- Prepare image loader and load a bias image
system make loaderselect bias

# ---- Load and start TE science run ----
send -i $cmd_id "load [incr ncmd] te 4 {[teImagePblock $ccd_list $pmode $phist]}\n"
command_echo 1 9 "load te"
send -i $cmd_id "start [incr ncmd] te 4\n"
command_echo 1 14 "start te science run"

# ---- Wait for data packets, then send image with events to image loader ----
set timeout $delay
expect -re "SWSTAT_FEP_STARTDATA\[^\r\n]*\[\r\n]+" {} timeout { fail timeout }
system make image RATE=$datarate
expect -re "dataTe\[^\r\n]*\[\r\n]+" {} timeout { fail timeout }

# ---- Examine the remaining packets ----
expect {
    -re "bepReadReply\[^\r\n]*commandId=$cmdId\[^\r\n]*\
    requestedAddress=0x$addr\[^\r\n]*\[\r\n]+" {
    incr nbeprep ; exp_continue
    }
    -re "scienceReport\[^\r\n]*\
    terminationCode=17\[\r\n]+" {
    incr nscirep ; exp_continue
    }
    -re "engineeringPseudo\[^\r\n]*\
    bilevels\=(\[0-9]+)\[\r\n]+" {
    if {[expr ($expect_out(1,string) & 15) == 14]} {
        incr nbilevel
    }
    if {$nbilevel < 1 || $nbeprep != 1 || $nscirep != 1} {
        exp_continue
    }
    pass " $nbilevel bilevels, $nbeprep BEP reads, $nscirep sci reports "
    }
    -re "deaHousekeepingData\[^\r\n]*\[\r\n]+" {
    if {[incr ndeahk] == 4} {
        send -i $cmd_id "write [incr ncmd] 0x$addr {\n$state\n}\n"
    }
    if {$ndeahk < 100} { exp_continue }
    }
    timeout { }
}

# ---- Fall through on timeout or 100+ housekeeping packets ----
fail " $nbilevel bilevels, $nbeprep BEP reads, $nscirep sci reports "
```

```
# ---- Return standard Board 11 DEA Housekeeping Block ----
proc deaHkPblock {rate} {
    set str "\ndeaBlockId = 0x0000abcd\nsampleRate = 10\n"
    for {set id 0} {$id <= 40} {incr id} {
    if {$id != 13 && $id != 14 && ($id < 21 || $id > 24)} {
        append str " queries = {\n\tccdId = 10\n\tqueryId = $id\n }\n"
    }
    }
    return $str
}

# ---- Return TE faint with image loader parameter block ----
proc teImagePblock {ccds pmode phist} {
  return "
  parameterBlockId           = 0x00000000
  fepCcdSelect               = $ccds
  fepMode                    = 2 # FEP_TE_MODE_EV3x3
  bepPackingMode             = $pmode
  onChip2x2Summing           = 0
  ignoreBadPixelMap          = 0
  ignoreBadColumnMap         = 0
  recomputeBias              = 1
  trickleBias                = 0
  subarrayStartRow           = 0
  subarrayRowCount           = 1023
  overclockPairsPerNode      = 8
  outputRegisterMode         = 0 # QUAD_FULL
  ccdVideoResponse           = 0 0 0 0 0 0
  primaryExposure            = 32
  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
  fep4SplitThreshold         = 50 50 50 50
  fep5SplitThreshold         = 50 50 50 50
  lowerEventAmplitude        = 0
  eventAmplitudeRange        = 3750
  gradeSelections            = 0xffffffff 0xffffffff 0xffffffff 0xffffffff\
                               0xffffffff 0xffffffff 0xffffffff 0xffffffff
  windowSlotIndex            = 65535
  histogramCount             = $phist
  biasCompressionSlotIndex   = 1 1 1 1 1 3
  rawCompressionSlotIndex    = 2
  ignoreInitialFrames        = 2
  biasAlgorithmId            = 1 1 1 1 1 1
  biasArg0                   = 1 1 1 1 1 1
  biasArg1                   = 2 2 2 2 2 2
  biasArg2                   = 0 0 0 0 0 0
  biasArg3                   = 0 0 0 0 0 0
  biasArg4                   = 0 0 0 0 0 0
  fep0VideoOffset            = 92 33 33 33
  fep1VideoOffset            = 43 33 43 23
  fep2VideoOffset            = 33 33 33 33
  fep3VideoOffset            = 33 33 33 33
  fep4VideoOffset            = 33 33 33 33
  fep5VideoOffset            = 33 33 33 33
  deaLoadOverride            = 0
  fepLoadOverride            = 0
"
}

# ---- Return TE with DEA parameter block ----
proc teDeaPblock {ccds del} {

  foreach ii [list \
    { 0 {21 14 16 14} {121 34 37 37} 1 } \
    { 1 {10 10 12 10} { 51 29 51 21} 1 } \
    { 2 {10 10 10 10} { 40 31 50 50} 1 } \
    { 3 {13 13 14 13} { 34 38 38 37} 1 } \
    { 4 {14 15 14 13} { 41 30 50 42} 1 } \
    { 5 {14 14 14 13} { 40 35 43 34} 3 } \
    { 6 {10 10 11 10} { 32 48 38 32} 1 } \
    { 7 {14 14 15 13} { 35 41 44 40} 3 } \
    { 8 { 9 10  9  9} { 21 42 14 34} 1 } \
    { 9 {12 12 12 11} { 35 36 35 35} 1 } \
    {10 { 0  0  0  0} {  0  0  0  0} 0 } \
  ] {
    lassign $ii n
    lassign $ii m thr($n) vid($n) cmp($n)
  }

  set fep 0
  foreach n $ccds {
    append ft "\n    fep${fep}EventThreshold         = $thr($n)"
    append fv "\n    fep${fep}VideoOffset            =";
    foreach ii $vid($n) {
      append fv " [expr $ii + $del]"
    }
    append bc " $cmp($n)"
    incr fep
  }

  return "
    parameterBlockId           = 0x00fff024
    fepCcdSelect               = $ccds
    fepMode                    = 2 # FEP_TE_MODE_EV3x3
    bepPackingMode             = 0 # BEP_TE_MODE_FAINT
    onChip2x2Summing           = 0
    ignoreBadPixelMap          = 0
    ignoreBadColumnMap         = 0
    recomputeBias              = 1
    trickleBias                = 0
    subarrayStartRow           = 0
    subarrayRowCount           = 1023
    overclockPairsPerNode      = 8
    outputRegisterMode         = 0 # QUAD_FULL
    ccdVideoResponse           = 0 0 0 0 0 0
    primaryExposure            = 32
    secondaryExposure          = 0
    dutyCycle                  = 0$ft
    fep0SplitThreshold         =    5    5    5    5
    fep1SplitThreshold         =    5    5    5    5
    fep2SplitThreshold         =    5    5    5    5
    fep3SplitThreshold         =    8    8    8    8
    fep4SplitThreshold         =    8    8    8    8
    fep5SplitThreshold         =    5    5    5    5
    lowerEventAmplitude        = 5
    eventAmplitudeRange        = 50
    gradeSelections            = 0xffffffff 0xffffffff 0xffffffff 0xffffffff\
                                 0xffffffff 0xffffffff 0xffffffff 0x7fffffff
    windowSlotIndex            = 65535
    histogramCount             = 0
    biasCompressionSlotIndex   = $bc
    rawCompressionSlotIndex    = 2
    ignoreInitialFrames        = 5
    biasAlgorithmId            =    1    1    1    1    1    1
    biasArg0                   =    1    1    1    1    1    1
    biasArg1                   =    3    3    3    3    3    3
    biasArg2                   =   20   20   20   20   20   20
    biasArg3                   =    0    0    0    0    0    0
    biasArg4                   =   20   20   20   20   20   20$fv
    deaLoadOverride            = 0
    fepLoadOverride            = 0
"
}

# ---- Construct TE block from a CCD list ----
proc teDeaPblock2 {ccds del} {
  # ---- CcdId, EventThresholds, VideoOffsets ----
  # CcdId 0, 1, 2, 3, 5, 7 calibrated
  foreach ii [list \
    { 0 { 7  7  7  7 } { 80 33 33 33 } 1 50 } \
    { 1 { 4  4  4  4 } { 43 33 43 23 } 1 50 } \
    { 2 { 4  4  4  4 } { 33 33 33 33 } 1 50 } \
    { 3 { 6  6  6  6 } { 33 33 33 33 } 1 50 } \
    { 4 { 4  4  4  4 } { 33 33 33 33 } 1 50 } \
    { 5 { 7  7  7  7 } { 33 33 33 33 } 3 26 } \
    { 6 { 4  4  4  4 } { 33 33 33 33 } 1 50 } \
    { 7 { 7  7  7  7 } { 33 33 33 33 } 3 26 } \
    { 8 { 4  4  4  4 } { 33 33 33 33 } 1 50 } \
    { 9 { 4  4  4  4 } { 33 33 33 33 } 1 50 } \
    {10 { 0  0  0  0 } {  0  0  0  0 } 0  0 } \
  ] {
    lassign $ii nn thr($nn) off($nn) bc($nn) ba3($nn)
  }
  foreach ii {0 1 2 3 4 5} {
    set cc [lindex $ccds $ii]
    append th1 "\n  fep${ii}EventThreshold       =$thr($cc)"
    append th2 "\n  fep${ii}SplitThreshold       = 4 4 4 4"
    append vid "\n  fep${ii}VideoOffset          =$off($cc)"
    append bcomp " $bc($cc)"
    append barg3 " $ba3($cc)"
  }
  return "
  parameterBlockId         = 0x00000001
  fepCcdSelect             = $ccds
  fepMode                  = 2 # FEP_TE_MODE_EV3x3
  bepPackingMode           = 0 # BEP_TE_MODE_FAINT
  onChip2x2Summing         = 0
  ignoreBadPixelMap        = 0
  ignoreBadColumnMap       = 0
  recomputeBias            = 1
  trickleBias              = 1
  subarrayStartRow         = 0
  subarrayRowCount         = 1023
  overclockPairsPerNode    = 8
  outputRegisterMode       = 0 # QUAD_FULL
  ccdVideoResponse         = 0 0 0 0 0 0
  primaryExposure          = 32
  secondaryExposure        = 0
  dutyCycle                = 0 $th1 $th2
  lowerEventAmplitude      = 0
  eventAmplitudeRange      = 3750
  gradeSelections          = 0xfeffffff 0xffffffff 0xfffffffb 0xfffff7ff\
                 0xffffffff 0xffffffff 0xffbfffff 0x7fffffff
  windowSlotIndex          = 65535
  histogramCount           = 0
  biasCompressionSlotIndex =$bcomp
  rawCompressionSlotIndex  = 2
  ignoreInitialFrames      = 100
  biasAlgorithmId          = 1  1  1  1  1  1
  biasArg0                 = 2  2  2  2  2  2
  biasArg1                 = 5  5  5  5  5  5
  biasArg2                 = 0  0  0  0  0  0
  biasArg3                 =$barg3
  biasArg4                 = 20 20 20 20 20 20 $vid
  deaLoadOverride          = 0
  fepLoadOverride          = 0
"
}

proc report {id} {
  global ndeahk nbilevel nbeprep nscirep
  puts "---- $id $ndeahk hk $nbilevel blv $nbeprep bep $nscirep sci ----"
}
```

MIT CSR ACIS	ENGINEERING CHANGE ORDER	ECO No. 36-1054

KAVLI INSTITUTE FOR ASTROPHYSICS AND SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DRAWING NO.	REVISION	DRAWING TITLE
36-58021.04	I	Flight Software Patch Release G-H-I Certification

REASON FOR CHANGE:

Certification of standard patch release G, which includes the updated *buscrash* patch, along with the optional patch *deahktrip*.

DESCRIPTION OF CHANGE:

Two optional patch combinations are certified as release G-H-I: the first is *smtimedlookup, eventhist, cc3x3, compressall* and *txings*, which includes the updated *buscrash* patch; the second comprises *smtimedlookup, eventhist, cc3x3, compressall, txings* and *deahktrip*, also with the new *buscrash*. The certification tests are made with this combination of optional release H patches, along with the full set of standard patches, release F.

	SIGNATURE	DATE	REMARKS
ORIGINATOR	PGF	06/29/18	Signature on file
MECHANICAL			
ELECTRICAL	DA	06/29/18	Signature on file
SOFTWARE	JEF	06/29/18	Signature on file
STRUCTURE			
FABRICATION			
SCIENCE			
SYSTEMS ENG.			
QUALITY	RB	06/29/18	Signature on file
PROJ. ENGINEER	RFG	06/29/18	Signature on file
DEPUTY PM			
PROJ. MANAGER			
APM RELEASE			

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

DOCUMENT NUMBER: 36-58021.03   REVISION: I

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

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
G	36-1046	Certify Rev-E-Opt-F patches	RFG	03/02/2011
H	36-1049	Certify Rev-F-Opt-G patches	RFG	12/16/2013
I	36-1054	Certify Rev-G-Opt-H patches	RFG	06/29/2018
I	36-1054	Certify Rev-G-Opt-H patches		

```
======================================================================

Title: ACIS cc3x3, eventhist, txings, compressall, smtimedlookup Patch Certification Re
lease Notes for Version I

Software Change Order: 36-1054

Build Date:   Mon Jul  2 17:45:42 EDT 2018
Part Number:  36-58021.03
Version:      I
CVS Tag:      cc3x3+eventhist+compressall+txings-G-H-I

Std Number:   36-58010
Std Version:  G
Std Tag:      review-release-G
Std SCO:      36-1048

Opt Number:   36-58020
Opt Version:  H
Opt Tag:      review-release-G-opt-H
Opt SCO:      36-1048

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, Science Mode Timed Lookup, and
    Threshold Crossing Trigger Patches.

    The certification consists of six 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 CC1x3. This suite
    performs CC1x3 tests to verify that the modifications to the existing
    BEP Continuous Clocking functions do not break the existing CC1x3
    functionality. Since the FEP software only contains CC3x3 code during
    CC3x3 runs (this is verified by the CC1x3 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 "zap1expo" 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 "zap1expo" 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 Threshold Crossing Trigger patch, txings, conducts a series of science runs -- timed exposure 3x3, event histogram, and raw, and continuously clocked 3x3, 1x3, and raw, increasing the threshold crossing rate and monitoring the ACIS bi-levels for the trigger signal, accompanied by the appropriate bepReadReply packet.

```
----------------------------------------------------------------------
Included Patches:
    cc3x3
    eventhist
    txings
    compressall
    smtimedlookup

----------------------------------------------------------------------
Test Support Patches:
    tlmio
    dearepl
    printswhouse

----------------------------------------------------------------------
Test Results:
    smtimedlookup --> PASS
    cc3x3 --> PASS
    eventhist --> PASS
    eventhist2 --> PASS
    compressall --> PASS
    txings --> PASS

----------------------------------------------------------------------
Regression Results:
    corruptblock --> PASS
    digestbiaserror --> PASS
    histogramvar --> PASS
    rquad --> PASS
    histogrammean --> PASS
    zap1expo --> PASS
    condoclk --> PASS
    fepbiasparity2 --> PASS
    fepbiasparity2 --> PASS
```

```
cornermean --> PASS
tlmbusy --> PASS
buscrash --> PASS
badpix --> PASS
buscrash2 --> PASS
buscrash2 --> PASS
buscrash2 --> PASS
```

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

DOCUMENT NUMBER: 36-58021.03   REVISION: I

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

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
G	36-1046	Certify Rev-E-Opt-F patches	RFG	03/02/2011
H	36-1049	Certify Rev-F-Opt-G patches	RFG	12/16/2013
I	36-1054	Certify Rev-G-Opt-H patches	RFG	06/29/2018
I	36-1054	Certify Rev-G-Opt-H patches		

```
======================================================================

Title: ACIS cc3x3, eventhist, txings, compressall, deahktrip, smtimedlookup Patch Certi
fication Release Notes for Version I

Software Change Order: 36-1054

Build Date:   Mon Jul  2 22:53:41 EDT 2018
Part Number:  36-58021.03
Version:      I
CVS Tag:      cc3x3+eventhist+compressall+txings-deahktrip-G-H-I

Std Number:   36-58010
Std Version:  G
Std Tag:      review-release-G
Std SCO:      36-1048

Opt Number:   36-58020
Opt Version:  H
Opt Tag:      review-release-G-opt-H
Opt SCO:      36-1048

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, Science Mode Timed Lookup,
    Threshold Crossing Trigger, and DEA H/K Temperature Alert Patches.

    The certification consists of seven tests, copied from the original test
    run during the Options Release. The tests have been modified to load
    all five 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 CC1x3. This suite
    performs CC1x3 tests to verify that the modifications to the existing
    BEP Continuous Clocking functions do not break the existing CC1x3
    functionality. Since the FEP software only contains CC3x3 code during
    CC3x3 runs (this is verified by the CC1x3 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 "zap1expo" 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 "zap1expo" 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 Threshold Crossing Trigger patch, txings, conducts a series of
    science runs -- timed exposure 3x3, event histogram, and raw, and
    continuously clocked 3x3, 1x3, and raw, increasing the threshold
    crossing rate and monitoring the ACIS bi-levels for the trigger signal,
    accompanied by the appropriate bepReadReply packet.

    The DEA H/K Thermal Trip patch starts a Timed Exposure Faint 3x3 event
    mode science run and, once the BEP output buffers are full, sends a
    "writeBep" command to trigger an alert. The test succeeds if it reads
    a "scienceReport" and a "bepReadReply" packet and if the software
    bilevels report a "1110" (14) value.


----------------------------------------------------------------------
Included Patches:
    cc3x3
    eventhist
    txings
    compressall
    deahktrip
    smtimedlookup

----------------------------------------------------------------------
Test Support Patches:
    tlmio
    dearepl
    printswhouse

----------------------------------------------------------------------
Test Results:
    smtimedlookup --> PASS
    cc3x3 --> PASS
    eventhist --> PASS
    eventhist2 --> PASS
    compressall --> PASS
    txings --> PASS
    deahktrip --> PASS

----------------------------------------------------------------------
Regression Results:
    corruptblock --> PASS
```

```
     digestbiaserror --> PASS
     histogramvar --> PASS
     rquad --> PASS
     histogrammean --> PASS
     zap1expo --> PASS
     condoclk --> PASS
     fepbiasparity2 --> PASS
     fepbiasparity2 --> PASS
     cornermean --> PASS
     tlmbusy --> PASS
     buscrash --> PASS
     badpix --> PASS
     buscrash2 --> PASS
     buscrash2 --> PASS
     buscrash2 --> PASS
```