MIT CSR ACIS	**ENGINEERING CHANGE ORDER**	**ECO No. 36-1048**

KAVLI INSTITUTE FOR ASTROPHYSICS AND SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DRAWING NO.	REVISION	DRAWING TITLE
36-58010	G	Flight Software Standard Patch Release F, Optional Release G

REASON FOR CHANGE:

The *buscrash2* patch has been updated to prevent any "trickle bias" anomaly (TBA). As a result, the *biastiming* and *untricklebias* patches are redundant and have been removed.

DESCRIPTION OF CHANGE:

Code has been added to the `checkMonitor()` method of *buscrash2* to stop bias reporting if an active FEP has been powered down or a TBA has occurred. In addition, the `goTaskEntry()` and `biasReady()` methods have been restructured, eliminating TBAs entirely.

	SIGNATURE	DATE	REMARKS
ORIGINATOR	RFG	12/16/13	Signature on file
MECHANICAL			
ELECTRICAL			
SOFTWARE			
STRUCTURE			
FABRICATION			
SCIENCE			
SYSTEMS ENG.			
QUALITY			
PROJ. ENGINEER			
DEPUTY PM			
PROJ. MANAGER			
APM RELEASE			

Existing ACIS Flight Software Patches

ID	Name	Rev	Size	Part	ECO	SPR
Standard Release F						
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	A	296	36–58030.30	1034	140
xii	badpix	A	60	36–58030.31	1037	141
xiii	buscrash2	C	1576	36–58030.32	1047	148,150
Optional Release G						
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
leaf	teignore	A	36	36-58030.09	1003	N/A
leaf	ccignore	A	36	36-58030.10	1004	N/A
Under Development						
11	fepbiasparity1	2	N/A	36–58030.18	1014	N/A
12	hybrid	3	6104	36–58030.13	1010	N/A
13	squeegy	6	4412	36-58030.23	1023	N/A
14	forcebiastrickle	1	N/A	36-58030.29	1024	133
Engineering Unit Utility Patches						
9	tlmio	2	10312	36–58030.07	1010	N/A
10	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
buscrash2	36–58032.32 (ECO 36–1047)	Prevent Trickle-Bias anomalies and BEP crashes	0	0	Reviewed and signed off
S/W Review	36–58010 (ECO 36–1048)	Documentation accompanying the individual patch ECOs	0	0	Reviewed and signed off
Certification	36–58021.04 (ECO 36–1049)	Documentation describing the multi-patch certification tests	0	0	Reviewed and signed off

a. typographical errors in the documentation

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

ACIS SOFTWARE PROBLEM REPORT

CENTER FOR SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FOR: Part Number	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	02/17/13		

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

Premature end of biasThief output

During OBSID 14887 on February 17, 2013, the output of *dataTeBiasMap* packets from the *biasThief* task ended abnormally after the first 53 packets from FEP_1, the first bias map to be reported. No further bias packets were produced. When processed through *psci*, the partial bias map from 14887 was confused with the FEP_1 bias map from the following run, OBSID 15610. The two maps were subsequently recovered, one partial and the other in full, by custom processing. ACIS was running with the E-F-G patch load, without the untricklebias patch (see ECO-1028).

Further analysis showed that *dataTeBiasMap* and *exposureTeVeryFaint* packets were interleaved, indicating a further instance of the *trickleBias* anomaly (see software problem report #133, M00062901). Also, the interval during which *biasThief* stopped creating *dataTeBiasMap* packets coincided with a software housekeeping report of a `SWSTAT_FEPREC_POWEROFF` with a count of 1 and an unexpected value of `0x00010002`. The value should have been a *fepId* in the range 0-5, *i.e.*, in this instance, 1 since FEP_1's map was being reported.

Corrective Action:

A **-Z** flag was added to the *psci* program to force bias maps and bias log to be closed at the end of each science run. OBSIDs 14887 and 15610 were then reprocessed at MIT with this flag. The CXCDS recovered the correct bias maps by separately processing the telemetry from each run.

Preliminary analysis with the ACIS engineering unit shows that the SWSTAT_FEPREC_POWEROFF with its illegal value is consistently generated by the *Test2_BiasThief::checkMonitor()* method of the *buscrash2* patch whenever *dataTeBiasMap* packets are alternated with *dataTe** event packets. The reason has been traced to a coding error in the inline part of the *buscrash2* patch, which should therefore be corrected.

It should be remarked that the result of the second anomaly - the premature termination of the bias packets - prevented the worst effects of the first anomaly, *i.e.*, the interleaving of bias and event packets, since the latter would otherwise have led to a "FEP T-Plane Latch-Up" condition on each of the active FEPs, leaving the FEPs inoperative until power-cycled. The loss of the bias maps is much less serious since they can almost certainly be replaced by recent maps created under similar conditions.

Problem closed on: Date:	Refer to ECO #:	Refer to Patch ID:
	36-1047	`buscrash2`

Problem ID: **M13021701**	Status: Open	Sheet: 148 of 150

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	08/08/13		

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

Missing exposures after trickle-bias anomaly in OBSID 53531

The observation used the 6 ACIS-S CCDs. FEP0 through FEP5 were assigned to S5,S3,S1,S2,S0,S4, Bias maps were created but a trickle-bias anomaly occurred during the copying of the CCD_S1 (FEP2) bias map to telemetry. The *buscrash2* patch reacted by halting the copy operation and posting a `SWSTAT_FEPREC_POWEROFF` event in software housekeeping. The first exposure records followed almost immediately.

After exposure 35, *exposureTeFaint* packets from FEPs 0,3,4,5 begin to lag behind those from FEPs 1,2; *i.e.*, the FEPs receiving data from front-illuminated CCDs lagged behind those from back-illuminated ones. Then from exposure 59 through 86, there were no exposures from FEPs 0,3,4,5, while none was missing from FEPs 1,2. From exposures 87 through 109, FEPs 0,3,4,5 dropped occasional exposured. From exposure 110 through the end of the run at exposure 212, all FEPs reported all exposures.

The fields in the *exposureTeFaint* and *dataTeFaint* packets were entirely nominal, similar in value to the preceding and following runs, and there was no indication of any anomaly in DEA or software housekeeping.

Corrective Action:

1. Check the telemetry archive for prior incidences of this anomaly.
2. Attempt to emulate in the ACIS engineering unit.

Problem closed on: Date:	Refer to ECO #:	Refer to Patch ID:
	36-1047	`buscrash2`

Problem ID: **M13080801**	Status: Open	Sheet: 150 of 150

MIT CSR ACIS	**ENGINEERING CHANGE ORDER**	**ECO No. 36-1047**

KAVLI INSTITUTE FOR ASTROPHYSICS AND SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DRAWING NO.	REVISION	DRAWING TITLE
36-58032.32	C	*buscrash2* patch to prevent Trickle-Bias anomalies and BEP crashes

REASON FOR CHANGE:

A BEP "trickle-bias" anomaly (TBA) is said to occur when bias copying and event processing run simultaneously in the science and bias thief tasks. A TBA exposes a bug in the *buscrash2inline.S* component of the *buscrash2* patch, causing the patched `checkMonitor()` method to transmit a bad `FepId` value to `checkMonitor()`, which halts further bias trickling. The bug has been traced to incorrect register usage. Further, TBAs have been found to be related to the DEA housekeeping task. Although the precise mechanism is currently unknown, a small change to *buscrash2* appears to prevent them entirely.

DESCRIPTION OF CHANGE:

Update the standard *buscrash2* patch: in the `checkMonitor()` method, already rewritten in response to SPR-142, add code to check all currently active FEPs, and examine the `fepInfo[]` structure to determine whether the science task has begun to process events, *i.e.*, that a TBA has occurred. If an active FEP is found to be powered down, or if event processing is active, `checkMonitor()` will return a `BoolFalse` value, terminating further bias reporting from the current science run. In addition, a call to `yield()` has been added to the `biasReady()` method (previously modified by the *biastiming* patch) that effectively eliminates the TBA. The updated *buscrash2* patch obviates the need for *biastiming* and also for the optional *untricklebias* patch.

	SIGNATURE	DATE	REMARKS
ORIGINATOR	RFG	08/13/13	Reviewed and accepted
MECHANICAL			
ELECTRICAL			
SOFTWARE			
STRUCTURE			
FABRICATION			
SCIENCE			
SYSTEMS ENG.			
QUALITY			
PROJ. ENGINEER			
DEPUTY PM			
PROJ. MANAGER			
APM RELEASE			

1. Reasons for the Patch

During OBSID 14887 on February 17th 2013, a Very Faint Timed Exposure ACIS observation, all expected bias maps were missing from telemetry except for the FEP_1 map, which was truncated. Using replacement bias maps, it was discovered that the science events were not corrupted and the CXCDS was able to process the run with these maps, using initial quadrant overclocks derived from corner pixel averages. The telemetry timeline for this science run was as follows:

sec	packet	comment
0	*dumpedTeBlock*	Start of the science run
104	–	Start of bias computation estimated from reported BEP clock value
739	–	Start of event processing estimated from reported BEP clock value
740	*dataTeBiasMap*	Receipt of the first bias packet from FEP_1
775	*exposureTeVeryFaint*	Receipt of the first exposure packet
777	*dataTeVeryFaint*	Receipt of the first event packet
816	*dataTeBiasMap*	Receipt of the last bias packet from FEP_1
828	*swHouseKeeping*	Reporting `FEPREC_POWEROFF` with value = `0x00010002`
12361	*commandEcho*	Command received to stop the science run
12365	*scienceReport*	ACIS reports that the science run has ended

The analysis of the anomaly began with the observation that the telemetry contained alternating *dataTeBiasMap* and *exposureTeVeryFaint* packets, which should not be the case, although this behavior occurred on 11 previous occasions, as tabulated in Appendix A. In the past, this "trickle-bias anomaly" (TBA) has often caused threshold-crossing plane latch-ups in one or more FEPs. This behavior was prevented in the recent run because bias trickling stopped almost as soon as exposure packets appeared, preventing the known anomaly in FEP firmware from latching the threshold-crossing planes.

The action moved to the ACIS engineering unit. A *bcmd* command was prepared to force the science task to begin event processing before bias trickling had ended:

```
write 0 0x8009b5d4 {
    0x00001021
}
```

with a companion command to reset the BEP to its normal behavior:

```
write 0 0x8009b5d4 {
    0x2c420001
}
```

It became clear that when the BEP ran with the *buscrash2* patch applied, the bias map packets stopped as soon as the first event packet appeared; without *buscrash2*, a complete set of bias map packets was written and, when the FEPs were subjected to high threshold-crossing rates, their firmware latched up in the manner seen in previous incidents of TBA. The reason for the anomalous behavior of *buscrash2* will be explained below, but we should mention here that, by stopping the reporting of bias maps, the anomaly prevented a nearly certain latch-up of most if not all of the active FEPs, which would have corrupted event data from that run, and from all succeeding runs until power to the latched FEPs could be recycled.

While the serendipitous action of *buscrash2* may prevent worse damage in the event of a TBA, it is more reliable to update the patch so that its operation is fully predictable, while adding code that explicitly stops writing bias map packets if a TBA is detected.

2. Description of the Original Patch

Two instances have been identified in which the unpatched BEP code attempts to address FEP memory via the memory-mapped interface without first checking that the FEP is powered up and is not in a reset state.

One instance is when updating new bias maps with the bad pixel and column lists. This is performed by the `FepManager::loadBadMaps()` method called from the `SmTimedExposure` and `SmContClocking` classes. The problem was identified in 2006 and the *buscrash* patch was developed in which the `loadBadMaps()` method was replaced with a version that checked whether a FEP was powered up and running before attempting to write into its bias map.

The current patch fixes the second identified cause of bus crashes, which occurs within the `trickleTeBias()` and `trickleCcBias()` methods of the `BiasThief` class. This is more difficult than patching `loadBadMaps()` because the `BiasThief` routines can either execute in their own `biasThief` task or, if the optional *untricklebias* patch has been applied, as part of the `scienceManager` task.

The `trickleTeBias()` and `trickleCcBias()` methods are quite lengthy, and if they were entirely rewritten, the resulting patch would be several kilobytes long and in danger of using up the remaining BEP storage available for patches. It was more economical to replace the minimum number of methods, and use inline patches where possible.

`BiasThief::checkMonitor()` was chosen as the simplest method to patch, since it is called immediately prior to allocating buffer space for each bias map packet. It reads as follows:

```
Boolean BiasThief::checkMonitor() {
    Boolean retval = BoolTrue;  // Assume no abort
    unsigned caught = requestEvent(EV_TASKQUERY | EV_ABORT);

    if (caught & EV_TASKQUERY) taskMonitor.respond();
    if (caught & EV_ABORT) retval = BoolFalse;

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

The patch replaces this method by constructing a new `Test2_BiasThief` class and making the replacement `checkMonitor()` a method of this class. This new method expects the active `FepId` to be passed as an argument, and this is achieved by means of inline patches.

```
Boolean Test2_BiasThief::checkMonitor(FepId fepid)
{
    Boolean retval = BoolTrue;  // Assume no abort

    if (fepid >= FEP_COUNT || fepManager.isEnabled(fepid) == BoolFalse) {
        swHousekeeper.report(SWSTAT_FEPREC_POWEROFF, fepid);
        retval = BoolFalse;
    } else {
        unsigned caught = requestEvent(EV_TASKQUERY|EV_ABORT);
        if (caught & EV_TASKQUERY) taskMonitor.respond();
        if (caught & EV_ABORT) retval = BoolFalse;
    }

    // ---- Return BoolTrue if no abort, ---
    // ---- BoolFalse if aborted or FEP not powered ----
    return retval;
}
```

The inline patches, which are discussed in detail in ECO 36-1041, are designed to pass the current `FepId` to general register 5 on entry to `checkMonitor()`, which is called from three separate locations in the unpatched

code: once each from `trickleTeBias`(), `trickleCcBias`(), and `getBuffer`(). The first two patches are simple since the `FepId` value is readily available in the calling routines and there are convenient `nop` instructions (no operation) that can be altered to copy the value into register 5. This is not the case with `getBuffer`(): it must itself be passed the `FepId` (in register 6), save it while it calls `waitForBuffer`(), and load it again into register 5 before calling `checkMonitor`().

3. Problem with the Original Patch

Although the patch was thoroughly tested, reviewed and certified by the ACIS instrument team, its behavior after a bias-thief anomaly was never tested, so a bug in the patched version of the `getBuffer`() method went unrecognized. Its original C++ code was as follows:

```
Boolean BiasThief::getBuffer(TlmForm&form)
{
    while (form.waitForBuffer() == BoolFalse) {
        if (checkMonitor() == BoolFalse)
            return BoolFalse;
    }
    return BoolTrue;
}
```

which compiled into the following MIPS (R3000) assembler code:

```
        .ent    BiasThief::getBuffer
        subu    $sp,$sp,40              # move stack pointer
        sw      $31,32($sp)             # save return address
        sw      $17,28($sp)             # save $17
        sw      $16,24($sp)             # save $16
        move    $16,$4                  # copy thief address
        move    $17,$5                  # copy form address
L260:   lw      $5,64($16)              # load form.waitForBuffer()
        nop
        jal     TlmForm::waitForBuffer # call form.waitForBuffer()
        move    $4,$17                  # copy form address
        bne     $2,$0,L272              # test return code in $2
        li      $2,1                    # load 1 into $2
        lw      $2,8($16)               # load thief virtual addresses
        nop
        lw      $2,64($2)               # load thief.checkMonitor()
        nop
        jal     $31,$2                  # call thief.checkMonitor()
        move    $4,$16                  # copy thief address
        bne     $2,$0,L260              # test return code in $2
        move    $2,$0                   # set $2 to zero
L272:   lw      $31,32($sp)             # restore return address
        nop
        lw      $17,28($sp)             # restore $17 to entry value
        nop
        lw      $16,24($sp)             # restore $16 to entry value
        nop
        addu    $sp,$sp,40              # restore stack pointer
        j       $31                     # return with code in $2
        .end    BiasThief::getBuffer
```

The original *buscrash2* patch needed to pass the `FepId` to `checkMonitor`(), which it did by altering two of the `nop` instructions, as follows:

```
        .ent    BiasThief::getBuffer
         subu    $sp,$sp,40              # move stack pointer
         sw      $31,32($sp)             # save return address
         sw      $17,28($sp)             # save $17
         sw      $16,24($sp)             # save $16
         move    $16,$4                  # copy thief address
         move    $17,$5                  # copy form address
L260:    lw      $5,64($16)              # load form.waitForBuffer()
         st      $6,36($sp)              # save FepId in stack
         jal     TlmForm::waitForBuffer  # call form.waitForBuffer()
         move    $4,$17                  # copy form address
         bne     $2,$0,L272              # test return code in $2
         li      $2,1                    # load 1 into $2
         lw      $2,8($16)               # load thief virtual addresses
         nop
         lw      $2,64($2)               # load thief.checkMonitor()
         lw      $5,36($sp)              # load FepId (arg1)
         jal     $31,$2                  # call thief.checkMonitor()
         move    $4,$16                  # copy thief address
         bne     $2,$0,L260              # test return code in $2
         move    $2,$0                   # set $2 to zero
L272:    lw      $31,32($sp)             # restore return address
         nop
         lw      $17,28($sp)             # restore $17 to entry value
         nop
         lw      $16,24($sp)             # restore $16 to entry value
         nop
         addu    $sp,$sp,40              # restore stack pointer
         j       $31                     # return with code in $2
         .end    BiasThief::getBuffer
```

The error occurs when `waitForBuffer()` returns zero (`BoolFalse`) in `$2` after which `checkMonitor()` is called and returns non-zero (`BoolTrue`), and the contents of `$6` will most probably have changed, so if `checkMonitor()` is called a second time, the `FepId` value will be bad. The correct patch would have transposed the "`st`" and "`L260: lw`" instructions, and reassembled the "`bne $2,$0,L260`" instruction to account for the changed "`L260`" address.

4. The Science Task and the Bias Thief

Once the science task has loaded microcode into the DEA boards PRAM and SRAM memory and "jittered" the DEA DACs, it calls the `computeBias()` method to create bias maps and, optionally, instruct the BiasThief task to copy them to send them to telemetry. The reason for assigning the copy operation to a separate task was the original hope that the FEP bias maps could be read by the BEP while the FEPs were processing event data. Sadly, this was not to be: once the threshold crossing rate exceeded a modest threshold, a design flaw in the FEPs' Actel FPGAs caused all thresholding to cease, effectively ruining that FEP's contribution to the run. The problem was discovered before the final BEP flight code was burned into the EEPROMs and a `waitForBias()` method was added to `computeBias()` which was designed to prevent the Science Task from starting FEP event processing before the BiasThief task had completed reading the FEP bias maps.

In subsequent testing, it was discovered that the Science Task didn't always wait for the BiasThief to end. The reason was believed to lie in `biasReady()`, the method that sets the processing flags and commands the task manager to start the BiasThief. If the BiasThief receives an `EV_TASKQUERY` signal — the means by which the task manager ensures that all tasks are "alive" — before `biasReady()` tells it to start, it can reset its flags and fool the Science Task into thinking that it has ended, before going on to copy the maps. A simple patch, *biastiming*, was developed to prevent this condition, but it hasn't always been successful — see Appendix A for a list of the 13 anomalies that have occurred in the flight unit since 1999 with *biastiming* firmly in place.

The `computeBias()` method of the Science Task controls the creation and reading of FEP bias maps. In the following, code not related this function has been suppressed.

```
Boolean ScienceMode::computeBias() {
    Boolean retval = BoolTrue;

    // ---- Tell enabled FEPs to start bias processing ----
    retval = fepManager.invokeBiasProcess();

    // ---- Wait for bias computation to complete ----
    retval = waitForBias();

    // ---- If bias thief needed, tell it that bias data is ready ----
    if (useBiasThief() == BoolTrue) {
        setupBiasThief();
        biasThief.biasReady();

        // --- Possibly block until trickle is complete or run aborted ---
        retval = waitForBiasTrickle();
        if (retval == BoolFalse)
            deaManager.stopSequencer();
    }
    return retval;
}
```

The original `biasReady()` method was as follows:

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

and was changed by the *biastiming* patch to be:

```
void Test_BiasThief::biasReady() {
    abortFlag = BoolFalse;        // Resolve order conflict with abort()
    notify (EV_START);            // Signal task to start bias
    busyFlag = BoolTrue;          // Bias Thief is being activated
}
```

The `waitForBiasTrickle()` method checks the values of `busyFlag` and `abortFlag`:

```
Boolean ScienceMode::waitForBiasTrickle() {
    // ---- Wait until bias thief is idle, or an abort ----
    while (biasThief.isBusy() == BoolTrue) {
        // ---- now check for query or abort signals from task manager
        ... // return BoolFalse if abort
    }
    return BoolTrue;
}

Boolean BiasThief::isBusy() const {
    if ((busyFlag == BoolTrue) && (abortFlag == BoolFalse)) {
        return BoolTrue;
    }
    return BoolFalse;
}
```

Meanwhile, the main `goTaskEntry()` method of the BiasThief task is waiting in an infinite loop:

```
void BiasThief::goTaskEntry() {
    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)) {

            // -- Trickle bias for each FEP --
            for (unsigned fepid = 0; fepid < FEP_COUNT; fepid++) {

                // - FEP has no bias -
                if (fepInfo[fepid].base == 0) {
                    continue;        // Skip to next FEP
                }

                 // - Tricke mode type -
                if (modetype == 0) { // Timed Exposure
                    if (trickleTeBias (FepId(fepid)) == BoolFalse) {
                        break;       // Aborted
                    }
                } else { // Continuous Clocking
                    if (trickleCcBias (FepId(fepid)) == BoolFalse) {
                        break;       // Aborted
                    }
                }
            }
        }

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

5. Trickle-Bias Anomaly Simulation

While testing a preliminary version of the improved *buscrash2* patch, the trickle-bias anomaly was simulated by issuing a *writeBep* command that caused the BEP Science task to bypass the call to `waitForBiasTrickle()` and thereby guarantee that the Science and BiasThief tasks executed simultaneously. To ensure that the patch was sufficiently "robust", the test included repeated stressing of the task management system, as follows:

- DEA housekeeping was enabled, with reporting rates increased to one per 8 seconds.
- A patch was applied to the Software Housekeeping task (see below) to increase its reporting rate to once per second.
- *readBep* commands were sent to the BEP at a rate of several times per second.

```
# ---- Increase S/W Housekeeping rate to one per second (accumInterval=1)----
send -i $cmd_id "write 0 0x80004204 {\n 1\n}\n"
command_echo 1 192 "Increase S/W Housekeeping Rate"
```

The improved patch proved fully capable of detecting the simulated anomalies. Also, under lengthy testing, it reported two instances of the trickle-bias anomaly when none had been anticipated. Since this had never before been seen in the ACIS engineering unit, further tests were run to try to reproduce the anomaly. After adding, varying, and removing the various stressors, the following proved to be the environment most likely to produce a trickle-bias anomaly:

- **DEA housekeeping that reports all 35 valid channels from the DEA interface board at rates of one per 8–24 seconds.**
- **Normal rates of software housekeeping – once per 64 seconds**
- **No additional memory readout commands**

No other "stressing" was necessary, although some ingenuity was required to shorten the execution time of each science run to minimize the average time between anomalies. The trick was to decide as soon as possible that the anomaly had not occurred, and then immediately to abort the BiasThief task by simulating a FEP power-down. See Section 10.5 for details.

The anomaly disappeared if only a handful of channels was reported in DEA housekeeping, and was most frequent when reporting 35 channels (the maximum number of physical readouts from the interface board). It was unaffected by increasing the Software Housekeeping reporting rate or by sending *readBep* commands. It occurred more frequently when the physical DEA was used (*i.e.*, with the *deaeng* patch) than with the virtual DEA (*i.e.*, with the *dearepl* patch and input from the Image Loader).

Now that it has become possible to reproduce the trickle-bias anomaly within the Engineering Unit, changes can be made to the *buscrash2* patch and to the earlier *biastiming* patch to see whether the anomaly can be eliminated entirely. The minimal changes that appear to accomplish this task are to add a `yield`() call within the `biasReady`() method, and a command to set `busyFlag` to `BoolTrue` in `checkMonitor`(). See Section 7 for details.

6. DEA Housekeeping

To understand how DEA housekeeping might provoke a trickle-bias anomaly, we must examine the code that runs in the DEA housekeeper task. As with the other BEP tasks, a top-level `goTaskEntry`() method runs in an infinite loop responding to task manager events. Once housekeeping is started, it calls `doHousekeeping`().

```
void DeaHousekeeper::doHousekeeping()
{
    // ---- Perform housekeeping until told to stop ----
    do {
        Tf_Dea_Housekeeping_Data form;
        // --- Obtain a packet buffer while responding to monitor queries ---

        waitGetPacket(form);

        // --- Perform queries ---
        for (unsigned ii = 0; ii < itemCount; ii++) {

            // -- Get board/query for next item --
            CcdId ccd = CcdId(deaBlock.get_Ccd_Id(ii));
            unsigned queryIndex = deaBlock.get_Query_Id(ii);
            unsigned value = DEAHOUSE_VALUE_INVALID;

            if (ccd < CCD_COUNT) {
                deaManager.queryCcd(ccd, DeaQueryCcdId(queryIndex), value);
            } else {
                deaManager.queryCntl(DeaQueryCntlId(queryIndex), value);
            }
```

```
            // -- Append to telemetry packet buffer --
            form.append_Entries (ccd, queryIndex, value);

            // -- If aborted, bump index and break out of loop --
            if (stopFlag == BoolTrue) {
               ii++;                  // form index to count of items
               break;                 // out of FOR loop
            }
            sleep(2);                  // sleep for 0.2 seconds
         }

         // --- If anything written, send the packet ---
         if (ii != 0) {
            form.set_Entries_Written (ii);
            form.post();
         }

         // -- Space out queries by the sample interval --
         waitForInterval();
      } while (stopFlag == BoolFalse);
   }

   Boolean DeaManager::queryCntl(unsigned queryid, unsigned& value) {
       DeaIoGuard guard(lock);

       // ---- Get exclusive access ----
       if (guard.waitFor(DEATIME_LOCK_WAIT) == BoolFalse) {
           deaInterface.setError(DeaBoard::DEAERR_LOCK_TIMEOUT);
           return BoolFalse;
       }

       DeaInterfaceClockEnable clockEnable;
       if (deaInterface.query(queryid, value)  == BoolFalse) {
           return BoolFalse;
       }
       return BoolTrue;
   }

   void DeaHousekeeper::waitForInterval() {

       // ---- Check for queries and commands prior to snooze ----
       unsigned eventMask = requestEvent(EV_DEA_COMMAND | EV_TASKQUERY);

       // ---- Respond to taskMonitor queries ----
       if ( (eventMask & EV_TASKQUERY) != 0) {
           taskMonitor.respond();
       }

       // ---- Abort if command is present ----
       if ( (eventMask & EV_DEA_COMMAND) != 0) {
           return;
       }

       // ---- Snooze for sample period ----
       sleep(sampleRate);
   }
```

Except for a brief interval at the start of each science run, only the interface board channels are interrogated, *i.e.*, calls are made to `queryCntl`(), not to `queryCcd`(). The former calls `DeaInterfaceController`::`query`() which calls two `DeaIoManager` methods: `waitForPort`() before sending the command, and `isReplyReady`() to wait for a reply. These routines contain loops that result in minimum delays of 0.22 and 3.0 milliseconds,

respectively, but they run with interrupts enabled: the execution time can be longer if the task manager switches tasks while waiting. Simultaneous access to the DEA by another task is prevented by the call to `Semaphore::waitFor()` in `queryCntl()`, which sets the `lock` semaphore in `queryCntl()` and releases it in the destructor of the `guard` object.

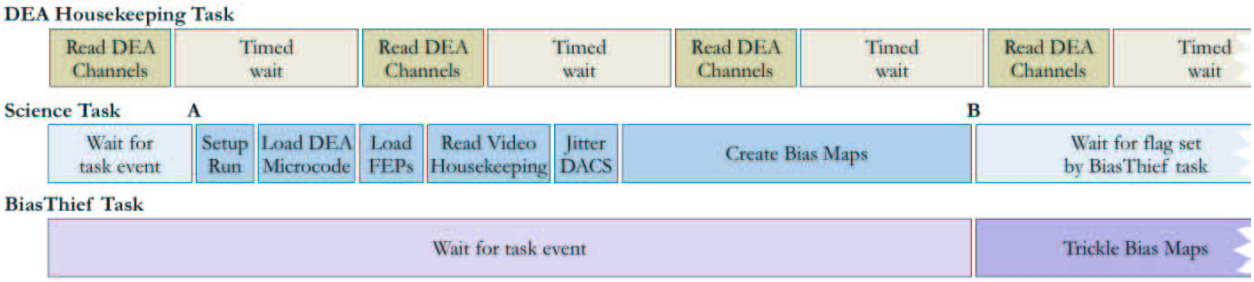

Figure 1. Example of a BEP task timeline: while the DEA Housekeeping task alternates between data collection and sleeping, a Science task is started at "A", loads and tests the DEA video boards and FEPs (relative duration not to scale), creates bias maps, and hands control to the BiasThief task at "B". In the engineering unit tests described in Sections 10.5 and 10.6, a trickle-bias anomaly is frequently, but not always, found to occur when "B" is towards the end of a 4-second timed wait in the DEA Housekeeping task.

The test described in Section 10.5 that reproduces TBAs in the engineering unit also reports the values of the BEP's 0.1 second interrupt timer at the start of DEA housekeeping collection, and at the time of the TBA. Fig. 1 shows how the tasks relate to each other. Results of the test are shown in Appendix B. Most, but not all, anomalies occurred in an 11-second readout cycle, and most of these within the last second (10 BEP interrupt intervals) of the cycle. At this point, the housekeeper has read the DEA channels, posted the telemetry packet, and is calling `waitForInterval()`, which checks for pending task events and, if none, calls `Task::sleep()`, a wrapper to the RTX routine, `NU_sleep()`. Inspecting the BEP source code, there is only one other instance in which `Task::sleep()` is ever called to sleep for more than 1 second (10 interrupt ticks), an 11 second wait in `DacJitter::waitForFlush()` called from `ScienceMode::jitterDacs()` while charge is flushed out of the CCDs at the start of each science run.

If the RTX operating system has trouble with these lengthy waits, perhaps the problem could be eliminated by splitting them into shorter intervals. `DeaHousekeeper::waitForInterval()` was therefore replaced by code that called `Task::sleep()` once per second. Alas! When this was tested in the manner described in Section 10.6, TBAs occurred at much the same rate as before, also preferentially in 11-second readout cycles and mostly within the last second of the 4-second waiting period.

Although Test 10.6 showed that splitting the waiting time didn't prevent TBAs, we now have a reliable – although probabilistic and imperfectly understood – way of triggering TBAs within the Engineering Unit and patches designed to circumvent it. From the discussion in Section 4, it is clear that we should also strengthen the "hand-over" between the call to `biasReady()` from the Science Task and the response by `goTaskEntry()` in the BiasThief task. We've added a `yield()` call to `biasReady()`, giving the task manager the opportunity to start the BiasThief task as soon as it has been scheduled. `BiasThief::goTaskEntry()` has also been patched to prevent an `EV_TASKQUERY` event from setting `busyFlag` to `BoolFalse` before the `EV_START` is received. Testing is described in Section 10.7.

7. Update to the Buscrash2 Patch

The new *buscrash2* patch replaces the previous standard patches, *biastiming* and *buscrash2*, and removes the necessity for an optional *untricklebias* patch. The new code is in **red**, below, with methods in **blue**. Since there is only one patch to the `BiasThief` class, we no longer need a separate `Test2_BiasThief` class.

```
#define private public
#include "filesscience/sciencemode.H"
#include "filesscience/sciencemanager.H"
```

```
#undef private
#include "filesmemserver/memoryserver.H"
#include "filesswhouse/swhousekeeper.H"
#include "filesexecutive/systemclock.H"

class Test_BiasThief : public BiasThief {
public:
    Test_BiasThief(unsigned taskid) : BiasThief(taskid) {};
    void goTaskEntry();
    void biasReady();
    Boolean checkMonitor();
};

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
}

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);
        // --- 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
}

Boolean Test_BiasThief::checkMonitor() {
    Boolean retval = BoolTrue;        // Assume no abort
    unsigned caught = requestEvent(EV_TASKQUERY | EV_ABORT);
```

```
        if (caught & EV_TASKQUERY) {
            taskMonitor.respond ();    // Task heartbeat
        }
        // ---- Check for task abort
        if (caught & EV_ABORT) {
            abortFlag = BoolTrue;
            retval = BoolFalse;    // Abort commanded
        }
        // ---- 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;
            }
        }
        // ---- Check if BiasThief and TE Science tasks are 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;
        }
        // ---- Return BoolFalse to stop bias trickle ----
        return retval;
    }
```

The call to `yield()` in `biasReady()` permits the task manager to pause the Science task and to call the BiasThief task to begin trickling the bias maps. Resetting the value of `busyFlag` to `BoolTrue` in `biasReady()` ensures that the Science task will see that the BiasThief is running. No inline patches are required since `checkMonitor()` examines the addresses of the FEP data buffers in `thief.fepInfo[fepid].base`, which is set to zero for all inactive FEPs. It then checks that each of them is powered up. The replacement version of `checkMonitor()` also determines whether a trickle-bias anomaly has occurred, *i.e.*, if event processing has started, by examining `scienceManager.currentMode->startTimeData`, which is initialized to `0xffffffff` by `scienceManager()` and then replaced with the value of the BEP clock once event processing begins.

8. Controlled Sources

buscrash2	
Makefile	Generate a stand-alone *buscrash2.bcmd* file
buscrash2.C	Source code for the `Test_BiasThief` class
buscrash2.mak	Makefile script to generate flight patch
buscrash2.pkg	Script to describe patch release
eco-1047.doc	Engineering change order describing the *buscrash2* patch
spr148.pdf	Originating software problem report
buscrash2/testsuite	
makebiascc	Generate a continuous clocking bias image
makebiaste	Generate a timed exposure bias image
makeimagecc	Generate a continuous clocking event image

makeimagete	Generate a timed exposure event image
buscrash2/testsuite/bug-hw	
Makefile	Run a test without any *buscrash2* patch
runtest.tcl	expect script to demonstrate a BEP bus crash
buscrash2/testsuite/fix-hw	
Makefile	Run a test with the *buscrash2* patch
buscrash2.bcmd	Stand-alone *buscrash2* patch with modifications to eliminate the trickle-bias anomaly
runtest.tcl	*expect* script to demonstrate prevention of BEP bus crash in timed exposure mode
runtest2.tcl	*expect* script to demonstrate prevention of BEP bus crash in continuous clocking mode
runtest3.tcl	*expect* script to demonstrate detection of the trickle-bias anomaly
buscrash2/testsuite/smoke	
Makefile	Run trickle-bias anomaly tests
buscrash2.bcmd	Stand-alone *buscrash2* patch with modifications to eliminate the trickle-bias anomaly
buscrash2nofix.bcmd	Stand-alone *buscrash2* patch without modifications to eliminate the trickle-bias anomaly
buscrash2nofix2.bcmd	Stand-alone *buscrash2* patch with modified `waitForInterval`() (see Section 10.6)
buscrash2-list.pl	PERL script to list trickle-bias anomalies in output packet file
runtest11.tcl	*expect* script to demonstrate trickle-bias anomalies with *buscrash2nofix.bcmd* patch
runtest11-2.tcl	*expect* script to demonstrate trickle-bias anomalies with *buscrash2nofix2.bcmd* patche
runtest12.tcl	*expect* script to demonstrate suppression of trickle-bias anomalies

9. Examining buscrash2 output

Should *buscrash2* detect a trickle-bias anomaly condition, it will write the contents of the `biasThief` object to a *bepReadReply* packet. *psci* will write the packet to a **.bepReadReply.*.dat* file, whose contents can be displayed by the *lbthief* command. *lbthief* is also invoked from *ltlm* when the `–v` flag is specified, *e.g.*, as follows:

```
bepReadReply[0] = {
  synch                     = 0x736f4166
  telemetryLength           = 99
  formatTag                 = 1 # TTAG_READ_BEP
  sequenceNumber            = 4798
  commandId                 = 1
  bepTickCounter            = 0x00017cad
  requestedAddress          = 0x80006b90
  requestedWordCount        = 92
  readAddress               = 0x80006b90
  biasThiefBlock = {
    task            = {
      taskId        = 7
      eventGroupId  = 7
      vtab          = 0x800024a0
    }
    timestamp       = 0xc6d3fe0b
    blockid         = 0x00000014
    modetype        = 0
    abortFlag       = 0
    biasHuffman     = {
      currentRef    = 0x800c4cfc
      numLast       = 0
      numOut        = 19
      accOut        = 524287
      leftOut       = 0
```

```
        lowLimit      = 3837
        highLimit     = 4348
        huffTruncCode = 27161
        vtab          = 0x800024f0
      }
      buffer_timeout  = 10
      maxrows         = 10
      pixels_per_row  = 1024
      busyFlag        = 0
      fepInfo[0] = {
        base          = 0xa8c00000
        ccd           = 7
        rowpixels     = 1024
        rowcnt        = 1024
        ccdrowoffset  = 0
        scale         = 1
        biasoffset    = { 0 0 0 0 }
        compress      = 1
        compressId    = 0xfffffffe
      }
      fepInfo[1] = {
        ...
      }
    }
  }
```

10. Testing

Because of the relatively low probability of occurrence of the trickle-bias anomaly, the demonstration and fix tests have not been changed since the previous version of *buscrash2*. Instead, additional "smoke" tests (10.5, 10.6 and 10.7) have been added. These lengthy tests should be performed before the patch load certification stage.

Exhaustive testing has shown that it is possible to maximize the TBA rate for a particular combination of DEA housekeeping parameters by varying the time delay between the *startDea* and *startScience* commands. However, this has been shown to depend on the precise BEP configuration and also on the delays within the interface between the engineering unit and the computer running the *expect* script. The "smoke" tests, 10.5, 10.6 and 10.7, therefore cycle through a succession of DEA housekeeping delays which, although not optimal, are guaranteed to generate a few TBAs within a reasonable time unless, in the case of Test 10.7, they don't!

All tests are performed on the ACIS Engineering Unit using 6 FEPs and the L-RCTU interface. Tests 10.1 through 10.4 also use the image loader. Tests 10.5, 10.6 and 10.7 use a single FEP and a flight-like video board in the engineering DEA. 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.

10.1. Test to reproduce a BEP bus crash

An *expect* procedure, "*bug-hw/runtest.tcl*", performs a timed-exposure science run with the *opt_tlmio*, *opt_printswhouse*, and *opt_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, DEA replacement, and standard flight patches are applied.

5. ACIS is warm-booted.
6. FEPs 0 through 5 are powered up.
7. A bias map containing the same value in each pixel of a given quadrant is written to the image loader.
8. A parameter block is sent to ACIS, calling for 6 FEPs to be run in timed-exposure faint 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 *dataTeBiasMap* packet has been received, three commands are sent to ACIS: two *stopScience* commands at 2-second intervals, followed by a 10-second delay and 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 (1) occurs; otherwise, the test fails.

10.2. Fix Test in TE Mode

This test, controlled by the *expect* procedure "*fix-hw/runtest.tcl*", is identical to the previous test except in two steps:

4. "*fix-hw/buscrash2.bcmd*" is added to the patch load.
12. The test passes if case (2) occurs; otherwise it fails.

10.3. Fix Test in CC Mode

This test, controlled by the *expect* procedure "*fix-hw/runtest2.tcl*", is identical to the previous test except in two steps:

8. A *cc_1x3* parameter block is sent.
10. The script waits for a *dataCcBiasMap* packet.

10.4. Fix Test with simulated Trickle-Bias Anomaly

This test, controlled by the *expect* procedure "*fix-hw/runtest3.tcl*", is similar to test 10.2 except in the following respects:

4. "*fix-hw/buscrash2.bcmd*" is added to the patch load and the write command described in Section 1 is sent to the FEP to simulate a trickle-bias anomaly.
10. "*make imagete*" is commanded to replace the bias image with one containing simulated x-ray events. The FEPs are not powered-down.
11. The script also notes whether a `SWSTAT_SCI_STARTRUN_BUSY` software housekeeping event has been reported, indicating that a trickle-bias anomaly was detected. The test passes if both this and the *scienceReport* was found, otherwise it fails.

10.5. Test to reproduce trickle-bias anomalies

This test, "*smoke/runtest11.tcl*" is identical to test 10.1 except in the following steps:

4. The *"smoke/buscrash2nofix.bcmd"* patch is applied. It modifies `Test_BiasThief::checkMonitor()` to detect and report TBAs, but it doesn't try to prevent them.
6. Only FEP_0 is powered up.
7. The image loader is not used. Pixel input to the FEP comes from the DEA. Since the only CCD used, (S3, `ccdId=7`), is attached to a flight-type video board, the *deaeng* patch was not needed.

8. A parameter block is sent to ACIS, calling for FEP_0 to be run in timed-exposure faint mode, with 3.2 second full-frame exposures from CCD_S3. The bias map will be created from the first image received.
9. The *expect* script loops over 100 tests, each divided into 17 sub-tests, *id* = 17 to 1 by –1.
10. Each sub-test begins by loading a DEA housekeeping block requesting the values of 35 channels from the interface board with a sample delay of *id* seconds., *i.e.*, since each channel is read in 0.2 seconds, the housekeeping packets will be spaced at intervals of *id*+7 seconds.
11. A TE science run is started.
12. A count is kept of the number of *dataTeBiasMap* packets produced. After the 10th, an *execBep* command is sent to the BEP to run the `FepManager::disableFep`() method, which will cause the BiasThief task to quit immediately. This greatly shortens each science run, permitting more to be executed in a given time.
13. If a *bepReadReply* packet containing `commandId=1` is received, a trickle-bias anomaly is presumed to have occurred.
14. It is assumed that the science run will have terminated. An *execBep* command is sent to the BEP to run the `FepManager::enableFep`() method, followed by a *stopDea* to terminate DEA housekeeping. The *id* value is decremented and execution continues with step 10, above.
15. After all tests are run, the expect script report a "PASS" if at least one trickle-bias anomaly has been detected; otherwise it reports a "FAIL".

Since the trickle-bias anomaly occurs in less than once per 100 runs, this "smoke" test is designed to execute each science run in as short a time as possible. After the *expect* script has terminated, the TBAs are examined by running the *"pkts.raw"* output file through the *"smoke/buscrash2-list.pl"* script (see Appendix C). Results from a typical run are tabulated in Appendix B. Because of the length (about 48 hours) and probabilistic nature of this test, it is not advisable to include it in the standard set of acceptance tests.

10.6. Test of trickle-bias anomaly suppression by update to the DeaHousekeeper task

This test, controlled by the *expect* procedure "*smoke/runtest11-2.tcl*". It is identical to test 10.5, except that the *"smoke/buscrash2nofix2.bcmd"* patch replaces *"smoke/buscrash2nofix.bcmd"* in Step 4. This patch modifies `waitForInterval`() by splitting the single call to `sleep(sampleRate)` with a succession of 1-second sleeps, as described in Section 6.

10.7. Test of trickle-bias anomaly suppression by updates to the BiasThief task

This test, controlled by the *expect* procedure "*smoke/runtest12.tcl*", is identical to test 10.6, except that the final *"fix-sw/buscrash2.bcmd"* patch replaces *"smoke/buscrash2nofix.bcmd"* in Step 7. In step 15, it reports "FAIL" if a trickle-bias anomaly is detected; otherwise, it reports "PASS".

Appendices

A. Instances of the Trickle-Bias Anomaly in the Flight Unit

obsid	date	mode		rows	cfg	feps	ccds	tx_fi	tx_bi	comments
371	2000-06-26	Te3x3	F	700	S6	3	4,6	3.71	0.13	Affected 659, 861, 795, 443, 62011
2310	2001-01-21	Te3x3	F	256	S6	–	–	3.69	0.14	No latch-ups
1997	2001-03-14	Te3x3	G	300	S1	–	–	–	23.14	No latch ups; one small bias map
3403	2001-10-29	Te3x3	F	1024	S6	3,4	6,8	3.72	0.13	Affected 1902, 2034, 61423
2010	2001-11-04	Te3x3	F	256	I6	3	6	154.40	3.90	Terminated by EPHIN/SCS107
5560	2005-07-09	Te5x5	F	1024	I5	3,5	1,6	4.90	–	FEPs power-cycled following latch-up
6730	2006-05-09	Te3x3	F	1024	I5	–	–	3.98	0.25	No latch-ups
9847	2008-04-19	Te3x3	G	512	S6	–	–	4.35	0.42	No latch-ups
12539	2011-03-25	Te3x3	F	256	S6	–	–	3.63	0.25	No latch-ups
13735	2012-03-28	Te3x3	F	256	S6	–	–	2.98	0.21	No latch-ups
14352	2012-06-22	Te3x3	F	512	S6	–	–	2.88	0.21	No latch-ups
14887	2013-02-17	Te5x5	F	1024	I5	–	–	2.78	0.22	No latch-ups; bias maps truncated/lost
15542	2013-04-01	Te5x5	F	512	S4	–	–	1.65	0.20	No latch-ups; no bias maps missing
53531	2013-07-24	Te3x3	F	1024	S6	–	–	3.27	0.18	No latch-ups; 4 maps truncated/lost

obsid ID of observation in process
date Date of the trickle-bias anomaly (TBA)
mode Event processing mode (Timed exposures, 3x3 or 5x5 pixel samples, F = faint, G = graded)
rows Number of 1024-pixel rows in each CCD output frame
cfg Configuration and number of CCDs used (S = mostly ACIS-S, I = mostly ACIS-I)
feps Indices of FEPs that suffered T-plane latch-ups as a result of the TBA
ccds Indices of CCDs whose FEPs suffered T-plane latch-ups
tx_fi Average threshold crossing rate (counts per row per second) for front-illuminated CCDs
tx_bi Average threshold crossing rate (counts per row per second) for back-illuminated CCDs

B. Trickle-Bias Anomalies in *runtest11* (see Section 10.5)

./;

Rate Length in seconds of each DEA data collection period
Date Date of the trickle-bias anomaly
UTC Time (UTC) of the trickle-bias anomaly
DEA-1 BEP interrupt counter at start of preceding DEA housekeeping interval
TBA BEP interrupt counter at time of this trickle-bias anomaly
DEA-2 BEP interrupt counter at start of following DEA housekeeping interval
DT Number of BEP ticks in current DEA housekeeping interval
DT1 Number of BEP ticks from start of DEA housekeeping interval to this TBA
DT2 Number of BEP ticks from this TBA to start of the next DEA housekeeping interval

C. *buscrash2-list.pl* – a PERL script to list Trickle-Bias Anomaly Timing

This script reads the EU's output telemetry, the standard output from *shim*, saving *pseudoScience* packets for their timing fields, *deaHousekeeping* packets for their BEP timer and delay fields, and *bepReadReply* packets as evidence of TBAs. Diagnostics are printed after receiving the first housekeeping packet after a TBA.

```
#! /usr/bin/env perl

$year = `date +%Y`+0;
@MO = ( 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, 9999 );
$MO[1]++ unless $year % 4;
$T = " N ID    DATE      UTC       DEA-1      TBA    DEA-2   DT  DT1  DT2\n";

# Open packet file
! $ARGV[0] || open(STDIN, $ARGV[0]) || die "$ARGV[0]: $!\n";

# Read each packet
while (read(STDIN, $buf, 8) == 8) {
  local($sync, $hdr) = unpack('V2', $buf);
  die "$ARGV[0]: bad sync\n" unless $sync == 0x736f4166;
  local($len, $type) = (4*($hdr & 1023)-8, ($hdr >> 10) & 63);
  last unless read(STDIN, $buf, $len) == $len;

  if ($type == 62) { # pseudoScience packet
    $sci = $buf;

  } elsif ($type == 11) { # deaHousekeepingData packet
    ($id, $dea1) = unpack('Vx4V', $buf);
    next unless $id < 20;
    if ($bep > 0 && $bep < $dea1) { # any previous bepReadReply ?
      print $T unless $n++;
      printf "%2d %2d %s %8d %8d %8d %4d %4d %4d\n", $n, $id,
        $dat, $dea0, $bep, $dea1, $dea1-$dea0, $bep-$dea0, $dea1-$bep;
      $bep = 0;
    }
    $dea0 = $dea1;

  } elsif ($type == 1) { # bepReadReply packet
    ($id, $bep) = unpack('vx2V', $buf);
    next unless $id == 1;

    # decode most recent irigB time field
    local($irig, $mo, $day, $hr, $min, $sec) = unpack('x10V', $sci);
    $day = ($irig >> 5) & 0x7ff;
    while ($day > $MO[$mo]) { $day -= $MO[$mo++]; }
    $sec = ($irig >> 20) | (($irig & 0x1f) << 12);
    $hr = int($sec/3600);
    $sec %= 3600;
    $min = int($sec/60);
    $dat = sprintf("%4d/%02d/%02d %02d:%02d:%02d",
      $year, $mo+1, $day, $hr, $min, $sec % 60);
  }
}

close(STDIN);

exit(0);
```

D. Glossary

Actel	A brand of low-power FPGA used in ACIS BEPs, FEPs, and DEA video boards
bcmd	EGSE software command to convert ACIS command from ASCII to binary
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
`bne`	BEP assembler command: branch if not equal
bug-hw	Directory containing tests designed to reproduce an ACIS hardware error
CXCDS	Chandra X-Ray Center Data System
DAC	Digital-to-Analog Converter — a component of the DEA video boards
DEA	ACIS Detector Electronics Assembly comprising analog and interface boards
deaeng	Patch to BEP software to initialize non-flight design DEA video boards
dearepl	Patch to BEP software to initialize non-flight design DEA video boards
destructor	Routine called to remove a C++ class object
DPA	ACIS Digital Processor Assembly comprising BEPs and FEPs
EEPROM	ACIS BEP's Electrically Erasable Programmable Read-Only Memory
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
FPGA	Field-Programmable Gate Array, a.k.a. firmware
housekeeper	BEP software task that writes ancillary data packets at predetermined intervals
L-RCTU	Jim Littlefield's Remote Command and Telemetry Unit, interface to the DPA
lbthief	EGSE software command to convert binary BiasThief data record to ASCII
ltlm	EGSE software command to convert DPA serial digital output to ASCII
`lw`	BEP assembler command: load (full) word
MIPS	Microprocessor without Interlocked Pipeline Stages — a reduced instruction set CPU design
`nop`	BEP assembler command: do nothing (and hence can be patched)
Nucleus	The original creators of RTX, long since bought out by IBM
OBSID	Chandra observation ID
PRAM	DEA's Programmable Random Access Memory, containing CCD sequencing microcode
psci	EGSE software command to split DPA serial digital output to separate data files
R3000	Type of reduced-instruction set CPU used in ACIS BEPs and FEPs
RTX	Commercial multi-threaded operating system used in the ACIS BEP
semaphore	Variable used by RTX for reliable inter-thread communication
shim	UNIX process to transmit commands to the EU and receive telemetry back
smoke	Directory containing tests designed to reproduce and eliminate trickle-bias anomalies
SRAM	DEA's Sequencer Random Access Memory, containing CCD sequencing primitives
`st`	BEP assembler command: store (full) word
TBA	Trickle-Bias Anomaly, when the BiasThief and Science task run simultaneously
TCL	Tool Command Language, a tiresome scripting language best avoided whenever possible

```
/* ============================================================
 *
 * $$Source: /nfs/acis/h3/acisfs/configcntl/patches/buscrash2/buscrash2.C,v $$
 *
 * Patch Name: Bus Crash Prevention, Part II
 *
 * Description:
 * This defines C++ replacement functions for
 * BiasThief::checkMonitor() and BiasThief::getBuffer()
 *
 * References:
 *
 * $$Log: buscrash2.C,v $
 * $Revision 1.4  2013/08/26 14:02:26  pgf
 * $Remove inline patches.
 * $Abort bias copying on powered-down FEPs and trickle-bias anomalies.
 * $Add yield() to biasReady().
 * $Rearrange goTaskEntry() to eliminate trickle-bias anomaly.
 * $
 * $Revision 1.3  2013/03/08 02:54:13  pgf
 * $Fix bug when bias and events are alternated.
 * $
 * $Revision 1.2  2008/08/27 18:48:53  pgf
 * $Rename Test_BiasThief class to Test2_BiasThief.
 * $
 * $Revision 1.1  2008/08/27 17:25:39  pgf
 * $Initial release.
 * $$
 * ============================================================ */

#define private public
#include "filesscience/sciencemode.H"
#include "filesscience/sciencemanager.H"
#undef private
#include "filesmemserver/memoryserver.H"
#include "filesswhouse/swhousekeeper.H"
#include "filesexecutive/systemclock.H"

class Test_BiasThief : public BiasThief
{
public:
    Test_BiasThief(unsigned taskid) : BiasThief(taskid) {};
    void goTaskEntry();
    Boolean checkMonitor();
    void biasReady();
};

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
}

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
}

Boolean Test_BiasThief::checkMonitor()
{
    DebugProbe probe;
    Boolean retval = BoolTrue;  // Assume no abort

    // ---- Check for task heartbeat
    unsigned caught = requestEvent(EV_TASKQUERY | EV_ABORT);
    if (caught & EV_TASKQUERY) {
    taskMonitor.respond (); // Task heartbeat
    }

    // ---- Check for task abort
    if (caught & EV_ABORT) {
    abortFlag = BoolTrue;   // Ensure task will abort
    retval = BoolFalse; // Abort commanded
    }

    // ---- 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;
    }
    }

    // ---- Check if BiasThief and TE 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;
    }

    // ---- Return BoolFalse to stop bias trickle ----
    return retval;
}
```

```
# ------------------------------------------------
#
# $$Source: /nfs/acis/h3/acisfs/configcntl/patches/buscrash2/buscrash2.pkg,v $$
#
# Bus Crash in Bias Thief 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: buscrash2.pkg,v $
# $Revision 1.7  2013/08/26 14:02:26  pgf
# $Remove inline patches.
# $Abort bias copying on powered-down FEPs and trickle-bias anomalies.
# $Add yield() to biasReady().
# $Rearrange goTaskEntry() to eliminate trickle-bias anomaly.
# $
# $Revision 1.6  2010/01/14 19:00:12  pgf
# $Fix typo.
# $
# $Revision 1.5  2009/11/03 16:34:34  pgf
# $Update description for Rev B
# $
# $Revision 1.4  2009/10/01 15:21:24  pgf
# $Update for Standard-D Release
# $
# $Revision 1.3  2008/08/28 04:45:12  pgf
# $Change names of fix-hw tests.
# $
# $Revision 1.2  2008/08/27 18:48:54  pgf
# $Rename Test_BiasThief class to Test2_BiasThief.
# $
# $Revision 1.1  2008/08/27 17:25:40  pgf
# $Initial release.
```

```
# $$
# ------------------------------------------------

# Identification Information
ident = $$Id: buscrash2.pkg,v 1.7 2013/08/26 14:02:26 pgf Exp $$

partnumber  = 36-58030.30
version     = C
environment = flight
eco         = 36-1047
reason      = Updated standard patch

# Release history information
approval A 36-1038 RFG 09/29/2009 Accepted
approval B 36-1041 RFG 01/06/2010 Accepted
approval C 36-1047 RFG 08/13/2013 Accepted

# Product and source file information
object buscrash2.o
func BiasThief::goTaskEntry Test_BiasThief::goTaskEntry
func BiasThief::checkMonitor Test_BiasThief::checkMonitor
func BiasThief::biasReady Test_BiasThief::biasReady
source buscrash2.pkg    01
source buscrash2.mak    02
source buscrash2.C  03

# Test information
test reproduce testsuite/bug-hw make ACISSERVER=$(ACISSERVER) TOOLS=$(TOOLS) PATCHDIR=$(PATC
HDIR)
test fixTe testsuite/fix-hw make ACISSERVER=$(ACISSERVER) TOOLS=$(TOOLS) PATCHDIR=$(PATCHDIR
)
test fixCc testsuite/fix-hw make ACISSERVER=$(ACISSERVER) TOOLS=$(TOOLS) PATCHDIR=$(PATCHDIR
) SCRIPT=runtest2
test fixTba testsuite/fix-hw make ACISSERVER=$(ACISSERVER) TOOLS=$(TOOLS) PATCHDIR=$(PATCHDI
R) SCRIPT=runtest3

# Initiating action information
spr    142
spr    148

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

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-cycled.

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!

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.

```
#! /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/sh

genObjectImage -e $* <<!
    Rows        = 512
    Columns     = 256
    Mode        = ABCD
    Overclocks  = 16
    Seed        = 12345678
    Noop        = 4 before Oclks
    Noop        = 0 before HSYNC
    Noop        = 8 after  HSYNC
    Noop        = 8 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/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/sh
#
# $Source: /nfs/acis/h3/acisfs/configcntl/patches/buscrash2/testsuite/makeimagecc,v $
#

genObjectImage $* <<!
    Rows        = 512
    Columns     = 256
    Mode        = ABCD
    Overclocks  = 16
    Seed        = 12345678
    Noop        = 4 before Oclks
    Noop        = 0 before HSYNC
    Noop        = 8 after  HSYNC
    Noop        = 8 before VSYNC
    Noop        = 3 after VSYNC

    Begin Node  = A
      Bias      = 212
      dBias     = 0
      OverClock = 201
      dOverClock    = 0
    End Node    = A

    Begin Node  = B
      Bias      = 314
      dBias     = 0
      OverClock = 302
      dOverClock    = 0
    End Node    = B

    Begin Node  = C
      Bias      = 416
      dBias     = 0
      OverClock = 403
      dOverClock    = 0
    End Node    = C

    Begin Node  = D
      Bias      = 518
      dBias     = 0
      OverClock = 504
      dOverClock    = 0
    End Node    = D

    Begin Event = event_0
      Rows      = 3
      Columns   = 3
      Value     = 200 175 0 0 500 0 0 0 0
    End Event   = event_0

    Begin Event = event_1
      Rows      = 3
      Columns   = 3
      Value     = 0 0 0 150 400 0 0 100 0
    End Event   = event_1

    Begin Event = event_2
      Rows      = 3
      Columns   = 3
      Value     = 0 0 0 0 350 125 0 0 275
    End Event   = event_2

    event_2 20 254
```

```
event_0 40 255
event_1 60 255
event_2 80 255
event_1 100 256
event_0 120 256
event_2 140 256
event_1 160 257
event_2 180 254
event_0 200 255
event_1 220 255
event_2 240 255
event_1 260 256
event_0 280 256
event_2 300 256
event_1 320 257
event_0 340 255
event_1 360 255
event_2 380 255
event_1 400 256
event_0 420 256
event_2 440 256
event_1 460 257
event_2 480 254
event_0 500 255

event_2 20 510
event_0 40 511
event_1 60 511
event_2 80 511
event_1 100 512
event_0 120 512
event_2 140 512
event_1 160 513
event_2 180 510
event_0 200 511
event_1 220 511
event_2 240 511
event_1 260 512
event_0 280 512
event_2 300 512
event_1 320 513
event_2 340 510
event_0 360 511
event_1 380 511
event_2 400 511
event_1 420 512
event_0 440 512
event_2 460 512
event_1 480 513
event_2 500 510

event_2 20 766
event_0 40 767
event_1 60 767
event_2 80 767
event_1 100 768
event_0 120 768
event_2 140 768
event_1 160 769
event_2 180 766
event_0 200 767
event_1 220 767
event_2 240 767
event_1 260 768
```

```
    event_0 280 768
    event_2 300 768
    event_1 320 769
    event_2 340 766
    event_0 360 767
    event_1 380 767
    event_2 400 767
    event_1 420 768
    event_0 440 768
    event_2 460 768
    event_1 480 769
    event_2 500 766
!

genObjectImage -e $* <<!
    Rows        = 512
    Columns     = 256
    Mode        = ABCD
    Overclocks  = 16
    Seed        = 12345678
    Noop        = 4 before Oclks
    Noop        = 0 before HSYNC
    Noop        = 8 after  HSYNC
    Noop        = 8 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/sh
#
# $Source: /nfs/acis/h3/acisfs/configcntl/patches/buscrash2/testsuite/makeimagete,v $
#

genObjectImage $* <<!
    Rows        = 1024
    Columns     = 256
    Mode        = ABCD
    Overclocks  = 16
    Seed        = 12345678
    Noop        = 4 before Oclks
    Noop        = 0 before HSYNC
    Noop        = 8 after  HSYNC
    Noop        = 25192 before VSYNC
    Noop        = 3 after VSYNC

    Begin Node  = A
      Bias      = 212
      dBias     = 0
      OverClock = 201
      dOverClock    = 0
    End Node    = A

    Begin Node  = B
      Bias      = 314
      dBias     = 0
      OverClock = 302
      dOverClock    = 0
    End Node    = B

    Begin Node  = C
      Bias      = 416
      dBias     = 0
      OverClock = 403
      dOverClock    = 0
    End Node    = C

    Begin Node  = D
      Bias      = 518
      dBias     = 0
      OverClock = 504
      dOverClock    = 0
    End Node    = D

    Begin Event = event_0
      Rows      = 3
      Columns   = 3
      Value     = 200 175 0 0 500 0 0 0 0
    End Event   = event_0

    Begin Event = event_1
      Rows      = 3
      Columns   = 3
      Value     = 0 0 0 150 400 0 0 100 0
    End Event   = event_1

    Begin Event = event_2
      Rows      = 3
      Columns   = 3
      Value     = 0 0 0 0 350 125 0 0 275
    End Event   = event_2

    event_2 20 254
```

```
event_0 40 255
event_1 60 255
event_2 80 255
event_1 100 256
event_0 120 256
event_2 140 256
event_1 160 257
event_2 180 254
event_0 200 255
event_1 220 255
event_2 240 255
event_1 260 256
event_0 280 256
event_2 300 256
event_1 320 257
event_0 340 255
event_1 360 255
event_2 380 255
event_1 400 256
event_0 420 256
event_2 440 256
event_1 460 257
event_2 480 254
event_0 500 255
event_1 520 255
event_2 540 255
event_1 560 256
event_0 580 256
event_2 600 256
event_1 620 257
event_0 640 255
event_1 660 255
event_2 680 255
event_1 700 256
event_0 720 256
event_2 740 256
event_1 760 257
event_2 780 254
event_0 800 255
event_1 820 255
event_2 840 255
event_1 860 256
event_0 880 256
event_2 900 256
event_1 920 257
event_1 940 256
event_0 960 256
event_2 980 256
event_1 999 257

event_2 20 510
event_0 40 511
event_1 60 511
event_2 80 511
event_1 100 512
event_0 120 512
event_2 140 512
event_1 160 513
event_2 180 510
event_0 200 511
event_1 220 511
event_2 240 511
event_1 260 512
event_0 280 512
```

```
event_2 300 512
event_1 320 513
event_2 340 510
event_0 360 511
event_1 380 511
event_2 400 511
event_1 420 512
event_0 440 512
event_2 460 512
event_1 480 513
event_2 500 510
event_0 520 511
event_1 540 511
event_2 560 511
event_1 580 512
event_0 600 512
event_2 620 512
event_1 640 513
event_2 660 510
event_0 680 511
event_1 700 511
event_2 720 511
event_1 740 512
event_0 760 512
event_2 780 512
event_1 800 513
event_2 820 510
event_0 840 511
event_1 860 511
event_2 880 511
event_1 900 512
event_0 920 512
event_2 940 512
event_1 960 513
event_2 980 512
event_1 999 513

event_2 20 766
event_0 40 767
event_1 60 767
event_2 80 767
event_1 100 768
event_0 120 768
event_2 140 768
event_1 160 769
event_2 180 766
event_0 200 767
event_1 220 767
event_2 240 767
event_1 260 768
event_0 280 768
event_2 300 768
event_1 320 769
event_2 340 766
event_0 360 767
event_1 380 767
event_2 400 767
event_1 420 768
event_0 440 768
event_2 460 768
event_1 480 769
event_2 500 766
event_0 520 767
event_1 540 767
```

```
    event_2 560 767
    event_1 580 768
    event_0 600 768
    event_2 620 768
    event_1 640 769
    event_2 660 766
    event_0 680 767
    event_1 700 767
    event_2 720 767
    event_1 740 768
    event_0 760 768
    event_2 780 768
    event_1 800 769
    event_2 820 766
    event_0 840 767
    event_1 860 767
    event_2 880 767
    event_1 900 768
    event_0 920 768
    event_2 940 768
    event_1 960 769
    event_2 980 768
    event_1 999 769
!

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        = 25192 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 buscrash2/testsuite/bug-hw/runtest.tcl"

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

# ---- Launch the command and telemetry server processes ----
set first_fep 0         ; # first FEP under test
set last_fep 5          ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD"  ; # desired outputRegisterMode
set ccd_list "4 5 6 7 8 9"  ; # desired fepCcdSelect

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

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

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

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

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

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.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                     =     2     2     2     2     2     2
    biasArg1                     =     5     5     5     5     5     5
    biasArg2                     =    20    20    20    20    20    20
    biasArg3                     =    26    26    50    50    50    50
    biasArg4                     =    20    20    20    20    20    20
    fep0VideoOffset              =    65    65    65    65
    fep1VideoOffset              =    65    65    65    65
    fep2VideoOffset              =    65    65    65    65
    fep3VideoOffset              =    65    65    65    65
    fep4VideoOffset              =    65    65    65    65
    fep5VideoOffset              =    65    65    65    65
    deaLoadOverride              = 0
    fepLoadOverride              = 0
}
"
command_echo 1 9 "load te"
system make bias

puts ""
puts "# Starting test 1"
puts ""
send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 "start science run"
set timeout 3600
expect {
    -re "dataTeBiasMap.*\[\r\n]" { }
    timeout { fail "Bias Failure" }
}
gotosleep 2

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 10

puts "# powering boards off"
send -i $cmd_id "change 0 systemConfig {
    entries = {
        itemId = 0
        itemValue = 0x0
    }
    entries = {
        itemId = 1
        itemValue = 0x0
    }
}
"
set timeout 60
expect {
    -re "SWSTAT_FEPMAN_POWEROFF.*\[\r\n]" { }
    timeout { fail "Power-down Failure" }
}
puts "# Powered off"

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

puts "Done"
```

```
#! /bin/env expect

puts "Welcome to buscrash2/testsuite/fix-hw/runtest.tcl"

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

# ---- Launch the command and telemetry server processes ----
set first_fep 0         ; # first FEP under test
set last_fep 5          ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD"  ; # desired outputRegisterMode
set ccd_list "4 5 6 7 8 9"  ; # desired fepCcdSelect

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

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

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

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

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

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcmd\
         $basedir/$tools/share/opt_printswhouse.bcmd\
         $basedir/$tools/share/opt_dearepl.bcmd\
         ./buscrash2.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                     =     2     2     2     2     2     2
    biasArg1                     =     5     5     5     5     5     5
    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 biaste

puts "\n# Starting test 1\n"
send -i $cmd_id "start 0 te 4\n"
command_echo 1 14 "start science run"
set timeout 3600
expect {
    -re "dataTeBiasMap.*\[\r\n]" { }
    timeout { fail "Bias Failure" }
}

puts "\n# powering FEP_2 off\n"
power_on_boards "4 5 10 7 8 9"

set timeout 60
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"
puts "\n# stopScience\n"
```

```
set timeout 60
expect {
    -re "bepStartupMessage.*\[\r\n]" {
    fail "Unexpected bus crash"
    }
    -re "scienceReport.*\[\r\n]" {
    pass "Science run ends without bus crash"
    }
    timeout {
    fail "No crash or scienceReport"
    }
}

puts "Done"
```

```
#! /bin/env expect

puts "Welcome to buscrash2/testsuite/fix-hw/runtest2.tcl"

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

# ---- Launch the command and telemetry server processes ----
set first_fep 0             ; # first FEP under test
set last_fep 5              ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD"      ; # desired outputRegisterMode
set ccd_list "4 5 6 7 8 9"      ; # desired fepCcdSelect

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

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

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

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

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

# ---- Apply patches ----
cold_boot
load_patch_list "$basedir/$tools/share/opt_tlmio.bcmd\
         $basedir/$tools/share/opt_printswhouse.bcmd\
         $basedir/$tools/share/opt_dearepl.bcmd\
         ./buscrash2.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 cc 4 {
    parameterBlockId            = 0x00000014
    fepCcdSelect                = $ccd_list
    fepMode                     = 1 # FEP_CC_MODE_EV1x3
    bepPackingMode              = 0 # BEP_CC_MODE_FAINT
    ignoreBadColumnMap          = 0
    recomputeBias               = 1
    trickleBias                 = 1
    rowSum                      = 0
    columnSum                   = 0
    overclockPairsPerNode       = 8
```

```
    outputRegisterMode          = $quad_mode
    ccdVideoResponse            =     0     0     0     0     0     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         = 24000
    gradeSelections             = 0x000f
    windowSlotIndex             = 65535
    rawCompressionSlotIndex     = 0
    ignoreInitialFrames         = 2
    biasAlgorithmId             =     0     0     0     0     0     0
    biasRejection               =     5     5     5     5     5     5
    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 10 "load cc"
system make biascc

puts "\n# Starting test 1\n"
send -i $cmd_id "start 0 cc 4\n"
command_echo 1 16 "start science run"
set timeout 3600
expect {
    -re "dataCcBiasMap.*\[\r\n]" { }
    timeout { fail "Bias Failure" }
}

puts "\n# powering FEP 2 off\n"
power_on_boards "4 5 10 7 8 9"

set timeout 60
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"
puts "\n# stopScience\n"

set timeout 60
expect {
    -re "bepStartupMessage.*\[\r\n]" {
    fail "Unexpected bus crash"
    }
    -re "scienceReport.*\[\r\n]" {
    pass "Science run ends without bus crash"
    }
    timeout {
    fail "No crash or scienceReport"
    }
```

```
}

puts "Done"
```

```
#! /bin/env expect

puts "Welcome to buscrash2/testsuite/fix-hw/runtest3.tcl"

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

# ---- Launch the command and telemetry server processes ----
set first_fep 1         ; # first FEP under test
set last_fep 4          ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD"  ; # desired outputRegisterMode
set ccd_list "10 7 5 6 8 10"    ; # desired fepCcdSelect

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

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

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

# ---- 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\
    ./buscrash2.bcmd"
warm_boot

# ---- Power on the FEPs and CCDs ----
power_on_boards "$ccd_list"
set timeout 120
expect {
    -re ".*SWSTAT_FEPMAN_ENDLOAD: $last_fep\[\r\n]" { }
    timeout { fail "Power-up Failure" }
}

# ---- Force trickle-bias anomaly behavior ----
send -i $cmd_id "write 0 0x8009b5d4 {\n0x00001021\n}\n"
command_echo 1 192 "Force Interleaving"

# ---- Load pBlock for Science Run ----
send -i $cmd_id "load 0 te 4 {
    parameterBlockId             = 0x00000014
    fepCcdSelect                 = $ccd_list
    fepMode                      = 3 # FEP_TE_MODE_EV5x5
    bepPackingMode               = 0 # BEP_TE_MODE_FAINT
    recomputeBias                = 1
    trickleBias                  = 1
    onChip2x2Summing             = 0
    ignoreBadPixelMap            = 0
    ignoreBadColumnMap           = 0
    overclockPairsPerNode        = 8
    outputRegisterMode           = $quad_mode
    ccdVideoResponse             =     0     0     0     0     0     0
```

```
    subarrayStartRow             = 256
    subarrayRowCount             = 511
    primaryExposure              = 17
    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
    windowSlotIndex              = 65535
    rawCompressionSlotIndex      = 0
    ignoreInitialFrames          = 2
    gradeSelections              = 0xffffffff 0xffffffff 0xffffffff 0xffffffff
                       0xffffffff 0xffffffff 0xffffffff 0xffffffff
    histogramCount               = 0
    eventAmplitudeRange          = 65535
    biasCompressionSlotIndex     =     1     3     1     3     1     1
    biasAlgorithmId              =     1     1     1     1     1     1
    biasArg0                     =     2     2     2     2     2     2
    biasArg1                     =     5     5     5     5     5     5
    biasArg2                     =    20    20    20    20    20    20
    biasArg3                     =    50    26    50    26    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"

# ---- Load a Bias File ----
system make biaste ROWS=512

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

# ---- Wait for end of bias readout ----
set timeout 3600
set sw 0
expect {
    -re "dataTeBiasMap\[^\r]*fepId=(\[0-9]+)\[^\r]*\
    ccdRow=(\[0-9]+) ccdRowCount=(\[0-9]+)\[^\r]*\[\r\n]+" {
    set nfep $expect_out(1,string)
    set nrow $expect_out(2,string)
    set rows $expect_out(3,string)
    if {$nfep != $last_fep || $nrow != $rows} {
        exp_continue -continue_timer
    }
```

```
    }
    -re "exposure\[^\r]*fepId=(\[0-9]+)\[^\r]*\
    exposureNumber=(\[0-9]+)\[^\r]*\[\r\n]+" {
    set nfep $expect_out(1,string)
    set nexp $expect_out(2,string)
    if {$nfep == $last_fep && $nexp == 5} {
        system make imagete ROWS=512
    }
    if {$nexp < 50} {
        exp_continue -continue_timer
    }
    }
    -re "SWSTAT_SCI_STARTRUN_BUSY\[^\r]*\[\r\n]+" {
    incr sw
    }
    -re "SWSTAT_FEPREC_POWEROFF\[^\r]*\[\r\n]+" {
    fail "Unexpected FEPREC_POWEROFF"
    }
    timeout {
    fail "Event Failure"
    }
}

# ---- Stop the Run ----
send -i $cmd_id "stop 0 science\n"
command_echo 1 19 "stop science run"

# ---- Wait for End ----
set timeout 120
expect {
    -re "bepStartupMessage\[^\r]*\[\r\n]+" {
    fail "Bus crash"
    }
    -re "scienceReport\[^\r]*\[\r\n]+" {
    if {$sw == 0} {
        fail "Science runs ends without STARTRUN_BUSY"
    } else {
        pass "Science run ends normally"
    }
    }
    timeout {
    fail "No crash or scienceReport"
    }
}

puts "Done"
```

```
#! /usr/bin/env perl

$year = ‘date +%Y‘+0;
@MO = ( 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, 9999 );
$MO[1]++ unless $year % 4;
$T = " N ID    DATE      UTC       DEA-1      TBA    DEA-2   DT  DT1  DT2 WAIT RC\n";

! $ARGV[0] || open(STDIN, $ARGV[0]) || die "$ARGV[0]: $!\n";

while (read(STDIN, $buf, 8) == 8) {
  local($sync, $hdr) = unpack(’V2’, $buf);
  die "$ARGV[0]: bad sync\n" unless $sync == 0x736f4166;
  local($len, $type) = (4*($hdr & 1023)-8, ($hdr >> 10) & 63);
  last unless read(STDIN, $buf, $len) == $len;
  if ($type == 62) {
    $sci = $buf;
  } elsif ($type == 7) {
    local($rc0, $op) = unpack(’x4Vx4v’, $buf);
    $rc = $rc0 if $op == 18;
  } elsif ($type == 10) {
    local(@sw, $ii) = unpack(’x8V*’, $buf);
    for ($ii = 0; $ii <= $#sw; $ii += 3) {
      $dea2 = $sw[$ii+2] if $sw[$ii] == 42;
    }
  } elsif ($type == 11) {
    ($id, $dea1) = unpack(’Vx4V’, $buf);
    next unless $id < 20;
    if ($bep > 0 && $bep < $dea1) {
      print $T unless $n++;
      $dea2 = ($dea1-$dea2) % ($dea1-$dea0) if $dea2;
      printf "%2d %2d %s %8d %8d %8d %4d %4d %4d %4d %2d\n",
    $n, $id, $dat, $dea0, $bep, $dea1, $dea1-$dea0,
    $bep-$dea0, $dea1-$bep, $dea2, $rc;
      $bep = 0;
    }
    $dea0 = $dea1;
  } elsif ($type == 1) {
    ($id, $bep) = unpack(’vx2V’, $buf);
    next unless $id == 1;
    local($irig, $mo, $day, $hr, $min, $sec) = unpack(’x10V’, $sci);
    $day = (($irig >> 5) & 0x7ff)+1;
    while ($day > $MO[$mo]) { $day -= $MO[$mo++]; }
    $sec = ($irig >> 20) | (($irig & 0x1f) << 12);
    $hr = int($sec/3600);
    $sec %= 3600;
    $min = int($sec/60);
    $dat = sprintf("%4d/%02d/%02d %02d:%02d:%02d",
      $year, $mo+1, $day, $hr, $min, $sec % 60);
  }
}

close(STDIN);
exit(0);
```

```
#! /bin/env expect

set run "11-2"

puts "Welcome to buscrash2/testsuite/smoke/runtest$run.tcl"

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

# ---- Launch the command and telemetry server processes ----
set first_fep 0         ; # first FEP under test
set last_fep 0          ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD"  ; # desired outputRegisterMode
set ccd_list "7 10 10 10 10 10" ; # desired fepCcdSelect

set patch_list "./buscrash2nofix2.bcmd"

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

# ---- Sleep while reporting packets ----
proc gotosleep { secs } {
    set timeout $secs
    expect { timeout { } }
}
# ---- Send commands slowly ----
proc send {arg1 arg2 arg3} {
    global cmd_id send_slow
    set send_slow {10 0.001}
    exp_send -s $arg1 $arg2 $arg3
}

# ---- Start DEA housekeeping at specified rate ----
proc startDeaHkp {rate} {
    global cmd_id

    # ---- Load DEA Housekeeping Block ----
    set str "
    deaBlockId   = $rate
    sampleRate   = $rate
    "
    for {set id 0} {$id <= 40} {incr id} {
        if {$id != 13 && $id != 14 && ($id < 21 || $id > 24)} {
        append str "queries = {\n\tccdId = 10\n\tqueryId = $id\n    }\n"
        }
    }
    send -i $cmd_id "load 0 dea 4 {$str}\r"
    command_echo 1 13 "load dea"
    gotosleep 2

    # ---- Start DEA Housekeeping ----
    send -i $cmd_id "start 0 dea 4\r"
    command_echo 1 18 "start dea housekeeping"
}

# ---- Stop DEA Housekeeping ----
proc stopDeaHkp {} {
    global cmd_id
    send -i $cmd_id "stop 0 dea\r"
    command_echo 1 20 "stop dea housekeeping"
    gotosleep 2
}
```

```
# ---- Indicate FEP disabled ----
proc disableFep {nfep} {
    global cmd_id
    send -i $cmd_id "exec 0 0x80089964 {\r0x80003da4\r$nfep\r}\r"
}

# ---- Enable FEP ----
proc enableFep {nfep} {
    global cmd_id
    send -i $cmd_id "exec 0 0x8008a0c8 {\r0x80003da4\r$nfep\r}\r"
    command_echo 1 195 "Enable FEP $nfep"
}

# ---- 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)
log_user 0
match_max 32768
gotosleep 2

# ---- Select Input from DEA ----
system make deaselect

# ---- Apply patches ----
cold_boot
load_patch_list $patch_list
warm_boot

# ---- Power on the FEPs and CCDs ----
power_on_boards "$ccd_list"
gotosleep 15

# ---- Load pBlock for Science Run ----
send -i $cmd_id "load 0 te 4 {
    parameterBlockId             = 0x00000014
    fepCcdSelect                 = $ccd_list
    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           = $quad_mode
    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
    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     =     1     1     1     1     1     1
    rawCompressionSlotIndex      = 0
    ignoreInitialFrames          = 0
    biasAlgorithmId              =     1     1     1     1     1     1
    biasArg0                     =     2     2     2     2     2     2
    biasArg1                     =     2     2     2     2     2     2
    biasArg2                     =    20    20    20    20    20    20
    biasArg3                     =     0     0     0     0     0     0
    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"
set anom 0

# ---- Loop over runs ----
for {set ntest 1} {$ntest <= 100} {incr ntest} {

    for {set rate 17} {$rate >= 1} {incr rate -1} {
    set timeout 120

    system "echo Starting runtest$run test $ntest rate $rate on ‘date‘"

    # ---- Set DEA Housekeeping ----
    startDeaHkp $rate

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

    # ---- Wait for bias readout or TBA ----
    expect {
        -re "dataTeBiasMap\[^\r]*dataPacketNumber=10 \[^\r]*\[\r\n]+" {
        disableFep $last_fep
        exp_continue -continue_timer
        }
        -re "bepReadReply\[^\r]*commandId=1\[^\r]*\[\r\n]+" {
        puts "Trickle Bias Anomaly rate=$rate"
        incr anom
        exp_continue -continue_timer
        }
        -re "scienceReport\[^\r]*\[\r\n]+" {
        }
        -re "\[^\r\n]*\[\r\n]+" {
        exp_continue -continue_timer
        }
        timeout {
```

```
        puts "Timeout waiting for scienceReport"
        }
    }

    # ---- Enable the FEP and disable DEA housekeeping ----
    enableFep $last_fep
    stopDeaHkp
    }
}

if {$anom > 0} {
    system "perl buscrash2-list.pl pkts.raw"
    system "mv pkts.raw pkts-$run-$anom.raw"
    pass "$anom Trickle-Bias Anomalies"
} else {
    system "mv pkts.raw pkts-$run-0.raw"
    fail "No Trickle-Bias Anomaly"
}

exit 0
```

```
#! /bin/env expect

set run "11"

puts "Welcome to buscrash2/testsuite/smoke/runtest$run.tcl"

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

# ---- Launch the command and telemetry server processes ----
set first_fep 0         ; # first FEP under test
set last_fep 0          ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD"  ; # desired outputRegisterMode
set ccd_list "7 10 10 10 10 10" ; # desired fepCcdSelect

set patch_list "./buscrash2nofix.bcmd"

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

# ---- Sleep while reporting packets ----
proc gotosleep { secs } {
    set timeout $secs
    expect { timeout { } }
}
# ---- Send commands slowly ----
proc send {arg1 arg2 arg3} {
    global cmd_id send_slow
    set send_slow {10 0.001}
    exp_send -s $arg1 $arg2 $arg3
}

# ---- Start DEA housekeeping at specified rate ----
proc startDeaHkp {rate} {
    global cmd_id

    # ---- Load DEA Housekeeping Block ----
    set str "
    deaBlockId   = $rate
    sampleRate   = $rate
    "
    for {set id 0} {$id <= 40} {incr id} {
        if {$id != 13 && $id != 14 && ($id < 21 || $id > 24)} {
        append str "queries = {\n\tccdId = 10\n\tqueryId = $id\n    }\n"
        }
    }
    send -i $cmd_id "load 0 dea 4 {$str}\r"
    command_echo 1 13 "load dea"
    gotosleep 2

    # ---- Start DEA Housekeeping ----
    send -i $cmd_id "start 0 dea 4\r"
    command_echo 1 18 "start dea housekeeping"
}

# ---- Stop DEA Housekeeping ----
proc stopDeaHkp {} {
    global cmd_id
    send -i $cmd_id "stop 0 dea\r"
    command_echo 1 20 "stop dea housekeeping"
    gotosleep 2
}
```

```
# ---- Indicate FEP disabled ----
proc disableFep {nfep} {
    global cmd_id
    send -i $cmd_id "exec 0 0x80089964 {\r0x80003da4\r$nfep\r}\r"
}

# ---- Enable FEP ----
proc enableFep {nfep} {
    global cmd_id
    send -i $cmd_id "exec 0 0x8008a0c8 {\r0x80003da4\r$nfep\r}\r"
    command_echo 1 195 "Enable FEP $nfep"
}

# ---- 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)
log_user 0
match_max 32768
gotosleep 2

# ---- Select Input from DEA ----
system make deaselect

# ---- Apply patches ----
cold_boot
load_patch_list $patch_list
warm_boot

# ---- Power on the FEPs and CCDs ----
power_on_boards "$ccd_list"
gotosleep 15

# ---- Load pBlock for Science Run ----
send -i $cmd_id "load 0 te 4 {
    parameterBlockId             = 0x00000014
    fepCcdSelect                 = $ccd_list
    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           = $quad_mode
    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
    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     =     1     1     1     1     1     1
    rawCompressionSlotIndex      = 0
    ignoreInitialFrames          = 0
    biasAlgorithmId              =     1     1     1     1     1     1
    biasArg0                     =     2     2     2     2     2     2
    biasArg1                     =     2     2     2     2     2     2
    biasArg2                     =    20    20    20    20    20    20
    biasArg3                     =     0     0     0     0     0     0
    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"
set anom 0

# ---- Loop over runs ----
for {set ntest 1} {$ntest <= 100} {incr ntest} {

    for {set rate 17} {$rate >= 1} {incr rate -1} {
    set timeout 120

    system "echo Starting runtest$run test $ntest rate $rate on ‘date‘"

    # ---- Set DEA Housekeeping ----
    startDeaHkp $rate

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

    # ---- Wait for bias readout or TBA ----
    expect {
        -re "dataTeBiasMap\[^\r]*dataPacketNumber=10 \[^\r]*\[\r\n]+" {
        disableFep $last_fep
        exp_continue -continue_timer
        }
        -re "bepReadReply\[^\r]*commandId=1\[^\r]*\[\r\n]+" {
        puts "Trickle Bias Anomaly rate=$rate"
        incr anom
        exp_continue -continue_timer
        }
        -re "scienceReport\[^\r]*\[\r\n]+" {
        }
        -re "\[^\r\n]*\[\r\n]+" {
        exp_continue -continue_timer
        }
        timeout {
```

```
        puts "Timeout waiting for scienceReport"
        }
    }

    # ---- Enable the FEP and disable DEA housekeeping ----
    enableFep $last_fep
    stopDeaHkp
    }
}

if {$anom > 0} {
    system "perl buscrash2-list.pl pkts.raw"
    system "mv pkts.raw pkts-$run-$anom.raw"
    pass "$anom Trickle-Bias Anomalies"
} else {
    system "mv pkts.raw pkts-$run-0.raw"
    fail "No Trickle-Bias Anomaly"
}

exit 0
```

```
#! /bin/env expect

set run "12"

puts "Welcome to buscrash2/testsuite/smoke/runtest$run.tcl"

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

# ---- Launch the command and telemetry server processes ----
set first_fep 0         ; # first FEP under test
set last_fep 0          ; # last FEP under test
set quad_mode "0 \# QUAD_ABCD"  ; # desired outputRegisterMode
set ccd_list "7 10 10 10 10 10" ; # desired fepCcdSelect

set patch_list "./buscrash2.bcmd"

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

# ---- Sleep while reporting packets ----
proc gotosleep { secs } {
    set timeout $secs
    expect { timeout { } }
}
# ---- Send commands slowly ----
proc send {arg1 arg2 arg3} {
    global cmd_id send_slow
    set send_slow {10 0.001}
    exp_send -s $arg1 $arg2 $arg3
}

# ---- Start DEA housekeeping at specified rate ----
proc startDeaHkp {rate} {
    global cmd_id

    # ---- Load DEA Housekeeping Block ----
    set str "
    deaBlockId   = $rate
    sampleRate   = $rate
    "
    for {set id 0} {$id <= 40} {incr id} {
        if {$id != 13 && $id != 14 && ($id < 21 || $id > 24)} {
        append str "queries = {\n\tccdId = 10\n\tqueryId = $id\n    }\n"
        }
    }
    send -i $cmd_id "load 0 dea 4 {$str}\r"
    command_echo 1 13 "load dea"
    gotosleep 2

    # ---- Start DEA Housekeeping ----
    send -i $cmd_id "start 0 dea 4\r"
    command_echo 1 18 "start dea housekeeping"
}

# ---- Stop DEA Housekeeping ----
proc stopDeaHkp {} {
    global cmd_id
    send -i $cmd_id "stop 0 dea\r"
    command_echo 1 20 "stop dea housekeeping"
    gotosleep 2
}
```

```
# ---- Indicate FEP disabled ----
proc disableFep {nfep} {
    global cmd_id
    send -i $cmd_id "exec 0 0x80089964 {\r0x80003da4\r$nfep\r}\r"
}

# ---- Enable FEP ----
proc enableFep {nfep} {
    global cmd_id
    send -i $cmd_id "exec 0 0x8008a0c8 {\r0x80003da4\r$nfep\r}\r"
    command_echo 1 195 "Enable FEP $nfep"
}

# ---- 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)
log_user 0
match_max 32768
gotosleep 2

# ---- Select Input from DEA ----
system make deaselect

# ---- Apply patches ----
cold_boot
load_patch_list $patch_list
warm_boot

# ---- Power on the FEPs and CCDs ----
power_on_boards "$ccd_list"
gotosleep 15

# ---- Load pBlock for Science Run ----
send -i $cmd_id "load 0 te 4 {
    parameterBlockId             = 0x00000014
    fepCcdSelect                 = $ccd_list
    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           = $quad_mode
    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
    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     =     1     1     1     1     1     1
    rawCompressionSlotIndex      = 0
    ignoreInitialFrames          = 0
    biasAlgorithmId              =     1     1     1     1     1     1
    biasArg0                     =     2     2     2     2     2     2
    biasArg1                     =     2     2     2     2     2     2
    biasArg2                     =    20    20    20    20    20    20
    biasArg3                     =     0     0     0     0     0     0
    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"
set anom 0

# ---- Loop over runs ----
for {set ntest 1} {$ntest <= 100} {incr ntest} {

    for {set rate 17} {$rate >= 1} {incr rate -1} {
    set timeout 120

    system "echo Starting runtest$run test $ntest rate $rate on ‘date‘"

    # ---- Set DEA Housekeeping ----
    startDeaHkp $rate

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

    # ---- Wait for bias readout or TBA ----
    expect {
        -re "dataTeBiasMap\[^\r]*dataPacketNumber=10 \[^\r]*\[\r\n]+" {
        disableFep $last_fep
        exp_continue -continue_timer
        }
        -re "bepReadReply\[^\r]*commandId=1\[^\r]*\[\r\n]+" {
        puts "Trickle Bias Anomaly rate=$rate"
        incr anom
        exp_continue -continue_timer
        }
        -re "scienceReport\[^\r]*\[\r\n]+" {
        }
        -re "\[^\r\n]*\[\r\n]+" {
        exp_continue -continue_timer
        }
        timeout {
```

```
        puts "Timeout waiting for scienceReport"
        }
    }

    # ---- Enable the FEP and disable DEA housekeeping ----
    enableFep $last_fep
    stopDeaHkp
    }
}

if {$anom > 0} {
    system "perl buscrash2-list.pl pkts.raw"
    system "mv pkts.raw pkts-$run-$anom.raw"
    fail "$anom Trickle-Bias Anomalies"
} else {
    system "mv pkts.raw pkts-$run-0.raw"
    pass "No Trickle-Bias Anomaly"
}

exit 0
```

TITLE: ACIS Flight Software Standard Patch Component Release Notes

DOCUMENT NUMBER: 36-58010 REVISION: F

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

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

Title: ACIS Patch Release Notes for Version F

Software Change Order: 36-1048

Build Date:   Wed Dec 18 19:08:13 EST 2013
Part Number:  36-58010
Version:      F
CVS Tag:      release-F

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

Load Size:    3672 bytes

----------------------------------------------------------------------
Description:
        This is the sixth letter release of the standard patch set for the
        ACIS Flight Software.

        The purpose of this release is to update the buscrash2 patch, removing
        biastiming and untricklebias.

        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
                  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-140
        SPR-121
        SPR-122
        SPR-148
        SPR-115
        SPR-123
        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-
>getQuadrant (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-
>getOverclockDelta (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)
                                                  $LM86:
                                ../filesscience/pixel3x3.C ****     split[2] = exposure-
>getSplitThreshold (quad);
                                                         .stabn 68,0,213,$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)
                                                  $LM87:
                                ../filesscience/pixel3x3.C ****
                                ../filesscience/pixel3x3.C ****     computePhGrade (docl
k, split);
                                                         .stabn 68,0,215,$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(FE
PeventRec3x3 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.thres
h[iquad]));
         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.thres
h[iquad]));
 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 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:     A
SCO:

Description:

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

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

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

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

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


Applicable Reports/Requests:
        SPR-140


Test Results:


Replaced Functions:
        FepManager::loadBadPixel


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: 36-58030.30
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 t
o
        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
-cycled.

        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: G

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

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

Title: ACIS Optional Patch Release Notes for Version G

Software Change Order: 36-1048

Build Date:   Wed Dec 18 23:00:38 EST 2013
Part Number:  36-58020
Version:      G
CVS Tag:      release-F-opt-G

Std Number:   36-58010
Std Version:  F
Std Tag:      release-F
Std SCO:      36-1048

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

----------------------------------------------------------------------
Description:
        This is the sixth letter release of the optional patch set for the ACIS
        Flight Software. The purpose of this release is to add the txings and
        fepthrottle patches and test them with the Rev. E Standard Patch 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

        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)
        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 (3160 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:        3160 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: 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 | FepMode        |
        |                                | smTimedLookup3x3  | BepPackingMode |
        |                                | smTimedLookup5x5  | 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.

MIT CSR ACIS	ENGINEERING CHANGE ORDER	ECO No. 36-1049

KAVLI INSTITUTE FOR ASTROPHYSICS AND SPACE RESEARCH
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

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

REASON FOR CHANGE:

Certification of standard patch release F, which includes the updated *buscrash2* patch and excludes *biastiming*, along with an optional patch set that was certified in release E-F-G, *i.e.*, *smtimedlookup, compressall, eventhist, cc3x3,* and *untricklebias*, but without the set that included *untricklebias*, which has been replaced by the new version of *buscrash2*.

DESCRIPTION OF CHANGE:

A single optional patch combination is certified as release F-G-H:
smtimedlookup, eventhist, cc3x3, compressall and *txings*.
The certification tests are made with this combination of the optional release G patches, with the full set of standard patches, release F.

	SIGNATURE	DATE	REMARKS
ORIGINATOR	RFG	12/16/13	Signature on file
MECHANICAL			
ELECTRICAL			
SOFTWARE			
STRUCTURE			
FABRICATION			
SCIENCE			
SYSTEMS ENG.			
QUALITY			
PROJ. ENGINEER			
DEPUTY PM			
PROJ. MANAGER			
APM RELEASE			

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

DOCUMENT NUMBER: 36-58021.03 REVISION: H

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

==========

Title: ACIS cc3x3, eventhist, txings, compressall, smtimedlookup Patch Certification Release Notes for Version H

Software Change Order: 36-1049

Build Date: Thu Dec 19 14:44:06 EST 2013
Part Number: 36-58021.03
Version: H
CVS Tag: cc3x3+eventhist+compressall+txings-F-G-H

Std Number: 36-58010
Std Version: F
Std Tag: release-F
Std SCO: 36-1048

Opt Number: 36-58020
Opt Version: G
Opt Tag: release-F-opt-G
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
- eventhist --> PASS
- compressall --> PASS
- txings --> PASS