		ENGINEERING CHANGE ORDER		<u>ECO No.</u> <u>36-1042</u>
CENTER FOR SPACE RESEARCH MASSACHUSETTS INSTITUTE OF TECHNOLOGY				
DWG. NO.	NEW REV.	DRAWING TITLE		
36-58010	E	Flight Software Standard Patch Release E, Optional Release E		
REASON FOR CHANGE: Update to standard patche <i>buscrash2</i> to operate correctly with the optional <i>untricklebias</i> patch.				
DESCRIPTION OF CHANGE: The new set of standard—release E—patches is compiled and loaded into a common address space so that each optional patch can be loaded independently of the others, provided the load order defined in <i>PatchRelease.spec</i> is maintained. No change is made to the optional patches, except for their release level, since the standard updates do not change the BEP load map. Patches <i>eventhist</i> , <i>cc3x3</i> , <i>ctireport1</i> , and <i>ctireport2</i> require that <i>smtimedlookup</i> is also loaded; similarly, the engineering patches <i>deaeng</i> , <i>dearepl</i> , and <i>printswhouse</i> require the <i>tlmio</i> patch. <i>deaeng</i> and <i>dearepl</i> must not be loaded at the same time. The standard patches must be loaded before <i>untricklebias</i> .				
	SIGNATURE	DATE	REMARKS:	
ORIGINATOR	RFG	01/06/10	Released	
MECHANICAL				
ELECTRICAL				
SOFTWARE				
STRUCTURE				
FABRICATION				
SCIENCE				
SYSTEMS ENG.				
QUALITY				
PROJ. ENGINEER				
DEPUTY PM				
PROJ. MANAGER				

Existing ACIS Flight Software Patches

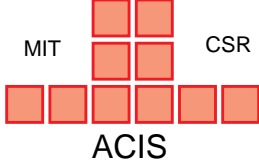
	Name	Rev	Size	Part	ECO	SPR
Standard Release E						
1	corruptblock	A	16	36-58030.01	994	113
2	digestbiaserror	A	64	36-58030.02	995	116
3	histogramvar	A	16	36-58030.03	999	115
4	biastiming	A	112	36-58030.04	993	117
5	rquad	A	16	36-58030.14	1000	121
6	histogrammean	A	156	36-58030.15	996	123
7	zaplexpo	A	64	36-58030.16	997	122
8	condoclk	A	640	36-58030.17	1012	127
9	fepbiasparity2	A	504	36-58030.19	1015	130
10	cornermean	A	32	36-58030.21	1017	128
11	tlmbusy	A	344	36-58030.29	1033	138
12	buscrash	A	296	36-58030.30	1034	140
13	badpix	A	60	36-58030.31	1037	141
14	buscrash2	B	428	36-58030.32	1041	142
Optional Release E						
1	eventhist	B	5908	36-58030.05	1025	N/A
2	cc3x3	B	4636	36-58030.06	1018	120,124,126
3	teignore	A	36	36-58030.09	1003	N/A
4	ccignore	A	36	36-58030.10	1004	N/A
5	smtimedlookup	A	3712	36-58030.24	1025	N/A
6	ctireport1	A	5452	36-58030.25	1026	N/A
7	ctireport2	A	2784	36-58030.26	1026	N/A
8	compressall	A	2368	36-58030.27	1027	134
9	untricklebias	B	1740	36-58030.28	1028	133
10	reportgrade1	A	816	36-58030.22	1021	131,132
Under Development						
1	hybrid	03	6104	36-58030.13	1010	N/A
2	fepbiasparity1	02		36-58030.18	1014	N/A
3	squeegy	06	4412	36-58030.23	1023	N/A
4	forcebiastrickle	01	N/A	36-58030.29	1024	133
Engineering Unit Utility Patches						
1	tlmio	02	10312	36-58030.07	1010	N/A
2	printshouse	01	7224	36-58030.08	986	N/A
3	deaeng	02	2604	36-58030.11	1010	N/A
4	dearepl	02	556	36-58030.12	1010	N/A

Status of Patch Release E, Optional Revision E

Name	Part Number	Description	Typos ^a	RIDs ^b	Status
<i>buscrash2</i>	36-58030.32 (ECO 36-1041)	Prevent BEP bus crash on FEP powerdown			
S/W Review	36-58020 (ECO 36-1042)	Documentation accompanying the individual patch ECOs			
Certification	36-58021.04 (ECO 36-1043)	Documentation describing the multi-patch certification tests			

a. typographical errors in the documentation

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

		ENGINEERING CHANGE ORDER		<u>ECO No.</u> <u>36-1041</u>
CENTER FOR SPACE RESEARCH MASSACHUSETTS INSTITUTE OF TECHNOLOGY				
DWG. NO.	NEW REV.	DRAWING TITLE		
36-58030.32	B	Flight S/W patch to prevent BEP bus crash on FEP power-down		
REASON FOR CHANGE: Testing multiple OBSIDs with the <i>buscrash2</i> and <i>untricklebias</i> patches installed has resulted in premature termination in ~10% of the runs. The fault has been traced to incorrect register use in <i>buscrash2</i>				
DESCRIPTION OF CHANGE: In the <code>BiasThief</code> class methods <code>trickleTeBias()</code> and <code>trickleCcBias()</code> , save <code>fepId</code> in register R6 before calling <code>getBuffer()</code> . Within <code>getBuffer()</code> , save R6 in an unused word in the stack, and load it back into R5 before calling <code>checkMonitor()</code> . This ECO begins with a repetition of the original <i>buscrash2</i> ECO 36-1038, and then describes the problem with that patch (Section 4) and its correction (Section 5).				
	SIGNATURE	DATE	REMARKS:	
ORIGINATOR	RFG	01/06/10	Released	
MECHANICAL				
ELECTRICAL				
SOFTWARE				
STRUCTURE				
FABRICATION				
SCIENCE				
SYSTEMS ENG.				
QUALITY				
PROJ. ENGINEER				
DEPUTY PM				
PROJ. MANAGER				

1. REASONS FOR THE PATCH

When the observatory's active CTU reset on 08/12/2008, ACIS was creating bias maps. The telemetry format changed to FMT4, preventing ACIS from sending further science telemetry to the RCTU. About 40 minutes later, SCS 107 was executed, in the course of which the FEPs were powered down. When the telemetry format was changed back to FMT2, the BEP suffered a bus error and watchdog reboot.

Subsequent tests and code analysis confirm that whenever ACIS is commanded to power-down its FEPs while copying bias maps to telemetry, it will cause a reset on the interface bus between the back-end processor (BEP) and the FEPs, causing the BEP to reboot without flight software patches. Normal operations must be restored via ground command. Since the observatory is usually in safe mode for several hours following the SCS 107, there is generally sufficient time to establish a real-time contact, set the BEP's warm-boot flag, and restart it. However, this takes time and manpower.

The bus crash has been traced to a flaw in the `BiasThief::trickleTeBias()` and companion `BiasThief::trickleCcBias()` methods. These routines are executed after the FEP bias maps have been created in order to copy the maps to telemetry packets and enqueue them to be written to the SI-RCTU. However, unlike other operations involving FEPs, `trickleTeBias()` and `trickleCcBias()` do not confirm that a FEP is powered up before reading from its bias memory. This causes the bus crash when the FEP isn't powered or is in a reset state.

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 `FepManager::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.

As the 08/12/2008 anomaly showed, the bias copying routines can be held in code loops for many hours waiting for telemetry buffers to become available and during that time they must not be prevented from responding to "heartbeat" interrupts from the task manager. The bus crash has been traced to a flaw in the `BiasThief::trickleTeBias()` and companion `BiasThief::trickleCcBias()` methods. These routines are executed after the FEP bias maps have been created in order to copy the maps to telemetry packets and enqueue them to be written to the SI-RCTU. However, unlike other operations involving FEPs, `trickleTeBias()` and `trickleCcBias()` do not confirm that a FEP is powered up before reading

from its bias memory. This causes the bus crash when the FEP isn't powered, or is in a reset state.

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 is more economical to replace the minimum number of methods, and use inline patches where possible, although this makes the job of preparing and reviewing the changes more difficult. Sorry.

Somewhat simplified, the `trickleTeBias()` method looks as follows:

```

Boolean BiasThief::trickleTeBias(FepId fepid)
{
    RowPacker packer;           // Bias row packing object
    Tf_Data_Te_Bias_Map form;   // Telemetry packet object
    unsigned rownum = fepinfo[fepnum].rowcnt;
    while (rownum > 0) {
        // --- Yield and check monitor queries ---
        yield ();
        if (checkMonitor() == BoolFalse) return BoolFalse;

        // --- Ensure packet has buffer ---
        if (form.hasBuffer() == BoolFalse) {
            if (getBuffer() == BoolFalse) return BoolFalse;
            setupTeForm();
            packer.setOutputBuffer();
        }

        Boolean postit = BoolTrue; // Assume the packet is full

        // --- Append bias pixels to telemetry packet ---
        if (packer.startRow() == BoolTrue) {
            if (packer.packSegment() == BoolTrue) {
                loadRowBuffer();
                postit = BoolFalse;
            }
        }

        // --- If full, release the packet to telemetry ---
        if (postit == BoolTrue) form.post();
    }

    // --- Flush out any partially filled packet ---
    if (form.hasBuffer() == BoolTrue) form.post();
    return BoolTrue;
}

```

The procedure is called for each FEP whose bias map is to be copied to telemetry. It creates two temporary objects, `packer` to assist in compressing the maps, and `form` to control the output packet format. Since we want to change this code so that it breaks out of the `while` loop when a FEP is unpowered, we must make the test (using the `isEnabled(fepid)` method of *FepManager*) and, if this returns `BoolFalse`, we must return `BoolFalse` from `trickleTeBias()` and, in the process, invoke the destructors of `packer` and `form`. The obvious replacement candidate is `checkMonitor()`, but there are two problems: it isn't passed the current `fepid` value, and this routine is also called from `getBuffer()`, which isn't passed `fepid` either.

```

Boolean BiasThief::getBuffer(TlmForm&form)
{
    // ---- Wait for form to get a telemetry buffer ----

```

```

while (form.waitForBuffer() == BoolFalse) {
    // --- Every so often, check monitor or aborts ---
    if (checkMonitor() == BoolFalse) return BoolFalse;
}
// ---- Got telemetry packet buffer ----
return BoolTrue;
}

```

Nevertheless, it is possible to patch both `getBuffer()` and `trickleTeBias()` (and the companion `trickleCcBias()`) so that `fepid` is passed as an argument to `checkMonitor()`, which must itself be changed from its original version:

```

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;
}

```

to

```

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 BoolFalse if abort, else BoolTrue ----
    return retval;
}

```

While this change prevents `checkMonitor()` from responding to `EV_TASKQUERY` events if the “if” clause is true, this will inevitably result in control passing to the outer loop of the `goTaskEntry()` procedure, which will always respond to these events.

The `Test2_BiasThief` class (not `Test_BiasThief` since this has already been defined in the `untricklebias` patch) is declared as a public child of the existing `BiasThief` class and we use a naughty `#define` trick to force the `FepManager::isEnabled()` to be publicly callable from “outside” its class. This is done to avoid proliferating the original “*biasThief.H*”.

```

#include "filesscience/biasthief.H"
#define private public
#include "filesprotocols/fepmanager.H"
#undef private
#include "filesswhouse/swhousekeeper.H"

class Test2_BiasThief : public BiasThief
{

```

```
public:
    Boolean checkMonitor(FepId fepid);
};
```

We have left the hard part until last: patching the binary instructions in `getBuffer()`, `trickleTeBias()` and `trickleCcBias()` so that they pass the `fepid` argument to the new version of `checkMonitor()`. To understand how this is done requires a short tutorial in some features of the architecture of the BEP's CPU, an R3000 RISC processor using MIPS standards, and of register usage by the GNU C++ compiler.

1. Function values are returned in hardware registers \$2 and \$3.
2. Function arguments are passed in registers \$4 through \$7. (In class methods, \$4 points to the object, \$5 contains the first argument, \$6 the second, etc.)
3. Registers \$16 through \$23 are preserved within functions: on entry, they are saved in the execution stack, and restored on exit. \$29 (aka \$sp) points to the stack.
4. On entry to a function, register \$31 points to the return address.
5. After executing a branch or function call instruction, the CPU *also* executes the instruction that immediately follows it, *before* taking the branch.
6. When data is loaded into a register from memory (but not from another register), it is not available for further computations until the *next* instruction has executed.

Because of this last feature, the compiler is forced to insert rather frequent “nop” (no operation) instructions, especially when loading the arguments of function calls into registers. We can replace these nops with instructions that pass the `fepid` value. From the source listing of the new `checkMonitor()` routine above, it is clear that \$5 must be made to contain the value of `fepid`. The way this is done differs between the cases in which the routine is called by `trickleTeBias()` and `trickleCcBias()` on the one hand, and by `getBuffer()` on the other. In the first case, `fepid` is declared in the caller, and is readily loaded into \$5. For instance, in the unpatched `trickleTeBias()`, `checkMonitor()` is called as follows:

```
lw      $2,8($16) // load $2 with BiasThief method table
nop     // wait for address to load into $2
lw      $2,64($2) // load address of checkMonitor method
nop     // wait for address to load into $2
jal     $31,$2   // call checkMonitor, return address in $31
move    $4,$16   // load address of BiasThief object into $4
```

To patch this, we replace one of the nop instructions with the following:

```
lw      $5,104($sp) // load $5 with the value of fepid
```

The corresponding change to `trickleCcBias()` is even simpler since, by inspecting the assembler listings, the compiler keeps the `fepid` value in register \$18:

```
move    $5,$18    // load $5 with the value of fepid
```

Passing `fepid` to the `checkMonitor()` call within `getBuffer()` is trickier since it must be passed through one routine into the other. However, `getBuffer()` is a very simple procedure. Before calling `checkMonitor()`, it calls only `form.waitForBuffer()`, which itself only calls the nucleus function `NU_Alloc_Partition()` that preserves all registers:

```
void* MemoryPool::waitForBuffer(unsigned timeout)
{
```



```

int      result;// RTX Result
unsigned* memPtr;// Obtained memory pointer
void*    retval;// Return Value

result = NU_Alloc_Partition(rtxPoolId, &memPtr, timeout);
if (result == NU_SUCCESS) {
    ASSERT(memPtr != 0);// Ensure valid ptr
    retval = memPtr;// Copy to return variable
} else {
    retval = 0; // Indicate nothing left
}
return (retval);// Return 0 (timeout) or memory pointer
}

```

Since `NU_Alloc_Partition()` is a simple function, not a class method, its three calling arguments use registers \$4, \$5, and \$6. An inspection of the assembler code shows that the remaining registers \$7 through \$29 are unused. If \$7 is loaded with the `fepid` value by `trickleTeBias()` or `trickleCcBias()` before they call `getBuffer()`, it will still be there when `getBuffer()` calls `checkMonitor()` (as verified by inspecting the assembler listings) so we insert the following into a convenient `nop` preceding the `getBuffer()` call in `trickleTeBias()`:

```
lw      $7,104($sp) // load $7 with the value of fepid
```

and make the corresponding update to `trickleCcBias()`:

```
move    $7,$18      // load $7 with the value of fepid
```

We're almost there. It remains only to replace a `nop` in `getBuffer()` just before it calls `checkMonitor()` with

```
move    $5,$7      // load $5 with the value of fepid
```

The full text of the assembler patch reads as follows. Global names refer to the 4-byte instructions in assembler listings. The syntax is “*module_lst_start_stop*”, where *module* refers to the assembler listing file “*module.lst*”, and “*start*” and “*stop*” are the beginning and ending hex addresses of the code to be replaced, as relative offsets to the start of the text segment in that listing.

```

.set noreorder
.set nomacro
.set noat
.text

# Pass fepid for call to checkMonitor() from getBuffer().
# This relies on $7 not changing during the waitForBuffer() call.

.globl biasthief_lst_0360_0360
.ent biasthief_lst_0360_0360
biasthief_lst_0360_0360:
    move    $5,$7      # $5 = fepid
.end biasthief_lst_0360_0360

# Load fepid for call to checkMonitor() from trickleTeBias().

.globl biasthief_lst_04d4_04d4
.ent biasthief_lst_04d4_04d4
biasthief_lst_04d4_04d4:
    lw      $5,104($sp) # $5 = fepid
.end biasthief_lst_04d4_04d4

# Load fepid for call to getBuffer() from trickleTeBias().

```

```

        .globl  biasthief_lst_050c_050c
        .ent   biasthief_lst_050c_050c
biasthief_lst_050c_050c:
        lw     $7,104($sp) # $7 = fepid
        .end   biasthief_lst_050c_050c

# Load fepid for call to checkMonitor() from trickleCcBias().
        .globl  biasthief_lst_07b0_07b0
        .ent   biasthief_lst_07b0_07b0
biasthief_lst_07b0_07b0:
        move   $5,$18      # $5 = fepid
        .end   biasthief_lst_07b0_07b0

# Load fepid for call to getBuffer() from trickleCcBias().
        .globl  biasthief_lst_07f4_07f4
        .ent   biasthief_lst_07f4_07f4
biasthief_lst_07f4_07f4:
        move   $7,$18      # $7 = fepid
        .end   biasthief_lst_07f4_07f4

```

3. CHANGE TO THE UNTRICKLEBIAS PATCH

Since the *untricklebias* patch also replaces the `BiasThief::checkMonitor()` method, this too must be updated to be compatible with *buscrash2*. The updated routine is as follows:

```

Boolean Test_BiasThief::checkMonitor(FepId fepid)
{
    Boolean retval = BoolTrue;

    // test if FEP powered up
    if (fepid >= FEP_COUNT ||
        fepManager.isEnabled(fepid)==BoolFalse) {
        swHousekeeper.report(SWSTAT_FEPREC_POWEROFF, fepid);
        return BoolFalse; // FEP not available or powered
    }

    // get science task event mask
    unsigned mask = EV_TASKQUERY | EV_SM_BIAS_ABORT_RUN;
    unsigned caught =
        taskManager.queryCurrentTask()->requestEvent(mask);

    // respond to task poll
    if (caught & EV_TASKQUERY) taskMonitor.respond();

    // abort sent by stopScience or RADMON inhibit
    if (caught & EV_SM_BIAS_ABORT_RUN) retval = BoolFalse;

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

```

Comparing this with the original `checkMonitor()` method on page 4, note the 2 changes:

1. The call to `fepManager.isEnabled()` and return if the FEP isn't accessible.
2. The call to `requestEvent()` from the current (*ScienceManager*) task, rather than from the *BiasThief* task.

4. PROBLEM WITH THE ORIGINAL PATCH

Although the patch was thoroughly tested, reviewed and certified by the ACIS instrument team, it failed two tests performed by the Chandra Science Operations Team. In both instances, the *untricklebias* batch was also loaded. Subsequent testing shows that runs that call for bias maps to be copied to telemetry and which use both the *buscrash2* and the *untricklebias* patches result in premature termination in ~10% of the runs. When *buscrash2* was used alone, no error was seen in tests using a total of 150 science runs.

In each case, the errors occurred when a bias map was about to be copied from a FEP to the BEP's output buffer. The error condition would strike at random: if multiple maps were to be copied, it would strike the last as often as the first, but always at the start of the map, never in the middle. In each case, software housekeeping reported the following:

```
swStatisticId      = 84 # SWSTAT_FEPREC_POWEROFF
count              = 1
value              = nnnn
swStatisticId      = 71 # SWSTAT_SCI_BIASFAILED
count              = 1
value              = 0
```

where “*nnnn*” varied from run to run. This pointed to the `BiasThief::checkMonitor()` method, which was replaced both in *buscrash2* and also in *untricklebias*. The major change in both versions described in Sections 2 and 3, respectively, was to pass the value of `fepId` to `checkMonitor()` which tests it and, if found invalid (*i.e.*, outside the range 0 through 5), reports the bad value in a `SWSTAT_FEPREC_POWEROFF` message and then halts the bias thief and the science run.

Since `checkMonitor()` is called before each bias packet is formatted, it is unexpected that the error should occur only on the first packet of a map. However, `checkMonitor()` is also called when a new packet buffer is needed, and this call is made from the `getBuffer()` method which is itself called by `trickleTeBias()` and `trickleCcBias()`. The original inline *buscrash2* patches pass `fepId` through `getBuffer()` in register R7. The contents of this register appeared to be preserved across the call to `TlmForm::waitForBuffer()` in `getBuffer()` (see Section 2) but it appears that this is not always the case, at least when the `BiasThief` methods are executed in the science task, which is the case when the *untricklebias* patches are installed.

5. UPDATE TO THE BUSCRASH2 PATCH

The full assembler language text of the new patch reads as follows. Changes to the original in Section 2 are colored red. The first change is on entry into `getBuffer()`, when the contents of register R6 are stored in the 10th fullword in that routine's execution stack, a location that is unused within that routine (and its callers). The MIPS compiler always allocates stack space in 8-byte segments, but this particular routine only uses 9 fullwords of stack, leaving 4 bytes available at offset 36 bytes from `$sp`, the stack pointer register. The remaining changes ensure that R6 contains the `fepId` value on entry to `getBuffer()` and, within that routine, reload `fepId` from the stack into R5 for the call to `checkMonitor()`.

```

.set noreorder
.set nomacro
.set noat
.text

# Save fepid in stack on entry to getBuffer()
.globl biasthief_lst_0340_0340
.ent biasthief_lst_0340_0340
biasthief_lst_0340_0340:
sw $6,36($sp) # 36($sp) = fepid
.end biasthief_lst_0340_0340

# Pass fepid for call to checkMonitor() from getBuffer().
.globl biasthief_lst_0360_0360
.ent biasthief_lst_0360_0360
biasthief_lst_0360_0360:
lw $5,36($sp) # $5 = fepid
.end biasthief_lst_0360_0360

# Load fepid for call to checkMonitor() from trickleTeBias().
.globl biasthief_lst_04d4_04d4
.ent biasthief_lst_04d4_04d4
biasthief_lst_04d4_04d4:
lw $5,104($sp) # $5 = fepid
.end biasthief_lst_04d4_04d4

# Load fepid for call to getBuffer() from trickleTeBias().
.globl biasthief_lst_050c_050c
.ent biasthief_lst_050c_050c
biasthief_lst_050c_050c:
lw $6,104($sp) # $6 = fepid
.end biasthief_lst_050c_050c

# Load fepid for call to checkMonitor() from trickleCcBias().
.globl biasthief_lst_07b0_07b0
.ent biasthief_lst_07b0_07b0
biasthief_lst_07b0_07b0:
move $5,$18 # $5 = fepid
.end biasthief_lst_07b0_07b0

# Load fepid for call to getBuffer() from trickleCcBias().
.globl biasthief_lst_07f4_07f4
.ent biasthief_lst_07f4_07f4
biasthief_lst_07f4_07f4:
move $6,$18 # $6 = fepid
.end biasthief_lst_07f4_07f4

```

No change is necessary for the replacement `checkMonitor()` method, nor to any part of the *untricklebias* patch.

6. CONTROLLED SOURCES

buscrash2	
<i>Makefile</i>	Generate a stand-alone <i>buscrash2.bcnd</i> file
<i>buscrash2.C</i>	Source code for the Test2_BiasThief class
<i>buscrash2.mak</i>	Makefile script to generate test patch
<i>buscrash2.pkg</i>	Script to generate patch release
<i>buscrash2inline.S</i>	Assembler code to generate inline patches
<i>eco-1041.doc</i>	Engineering change order describing the <i>buscrash2</i> patch
<i>spr142.pdf</i>	Originating software problem report
buscrash2/testsuite	
<i>makebias</i>	Generate a bias image and copy it to the image loader
buscrash2/testsuite/bug-hw	
<i>Makefile</i>	Run a test without the <i>buscrash2</i> patch
<i>runtest.tcl</i>	<i>expect</i> script to demonstrate a BEP bus crash
buscrash2/testsuite/fix-hw	
<i>Makefile</i>	Run a test with the <i>buscrash2</i> patch
<i>buscrash2.bcnd</i>	Stand-alone <i>buscrash2</i> patch
<i>runtest.tcl</i>	<i>expect</i> script to demonstrate prevention of BEP bus crash in TE mode
<i>runtest2.tcl</i>	<i>expect</i> script to demonstrate prevention of BEP bus crash in CC mode

7. TESTING

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

Because of the relatively low probability of occurrence of the problem described in Section 4, these tests have not been changed. Instead, additional “stress” tests have been added during the patch load certification stage.

7.1. Reproduce Test

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 down which ACIS commands will be written.
2. A telemetry pipe is spawned, terminating in the “*psci -m -u*” packet monitoring function with *expect* examining the standard output.
3. ACIS is cold-booted.
4. Software housekeeping, DEA replacement, and standard flight patches are applied.
5. ACIS is warm-booted.
6. 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 *te_3x3* parameter block is sent to ACIS. It calls for 6 FEPs to be run in faint mode, calling for 3.3 second full-frame exposures.
9. A science run is started. Its telemetry is monitored by the *expect* script.
10. Once a *dataTeBiasMap* packet is 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.

7.2. Fix Test in TE Mode

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

1. In step 4, *buscrash2.bcnd* is added to the patch load.
2. In step 12, the test passes if case (2) occurs; otherwise it fails.

7.3. Fix Test in CC Mode

This test, controlled by the *expect* procedure “*fix-hw/runtestcc2.tcl*”. is identical to the Reproduce Test except in two respects:

1. In step 4, *buscrash2.bcnd* is added to the patch load.
2. In step 8, a *cc_1x3* parameter block is sent.
3. In step 10, the script waits for a *dataCcBiasMap* packet.
4. In step 12, the test passes if case (2) occurs; otherwise it fails.

TITLE: ACIS Flight Software Standard Patch Component Release Notes

DOCUMENT NUMBER: 36-58010 REVISION: E

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

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

Software Change Order: 36-1042

Build Date: Wed Nov 4 19:15:19 EST 2009
Part Number: 36-58010
Version: E
CVS Tag: release-E

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

Load Size: 2660 bytes

Description:

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

The purpose of this release is to update the buscrash2 optional patch.

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

biastiming	- Fixes SPR 117
corruptblock	- Fixes SPR 113
digestbiaserror	- Fixes SPR 116
histogramvar	- Fixes SPR 115
rquad	- Fixes SPR 121
histogrammean	- Fixes SPR 123
zaplexpo	- Addresses SPR 122
condock	- Addresses SPR 127
fepbiasparity2	- Addresses SPR 130
cornermean	- Fixes SPR 128
tlmbusy	- Fixes SPR 138
buscrash	- Fixes SPR 140
badpix	- Fixes SPR 141
* buscrash2	- Fixes SPR 142

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-142
SPR-128
SPR-123
SPR-127
SPR-130
SPR-138
SPR-122

SPR-141
SPR-115
SPR-113
SPR-140
SPR-117
SPR-116
SPR-121

Included Patches:

tlmbusy
fepbiasparity2
biastiming
histogramvar
badpix
zaplexpo
digestbiaserror
corruptblock
cornermean
buscrash
buscrash2
rquad
condock
histogrammean

Additional Release Level Tests:

=====
Patch Name: tlmbusy

Part Number: 36-58030.29

Version: A

SCO:

Description:

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

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

Applicable Reports/Requests:

SPR-138
SER-None

Test Results:

smoke --> PASS

Replaced Functions:

TlmManager::post

Command Impact:

None.

Telemetry Impact:

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

Science Impact:

None.

=====
Patch Name: fepbiasparity2

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

Description:

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

The patch has no effect on other FEPs.

Applicable Reports/Requests:
SPR-130

Test Results:

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

Replaced Functions:

Command Impact:

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

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

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

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

Telemetry Impact:

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

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

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

Science Impact:

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

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

=====
Patch Name: biastiming

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

Description:

Reason:

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

Symptom:

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

Symptom Impact:

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

Symptom Cause:

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

Fix Description:

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

Applicable Reports/Requests:

SPR-117

Test Results:

unit --> PASS
fix --> PASS

Replaced Functions:

BiasThief::biasReady

Command Impact:

None

Telemetry Impact:

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

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

Science Impact:

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

With this patch, this situation will not occur.

=====
Patch Name: histogramvar

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

Description:

This patch fixes a software problem, SPR-115.

Symptom:

The Raw Histogram Mode 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 overlocks.

Symptom Cause:

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

Fix Description:

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

Applicable Reports/Requests:

SPR-115

Test Results:

reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:

None

Telemetry Impact:

None

Science Impact:

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

=====
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:

reproduce --> PASS
fix --> PASS

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: zaplexpo

Part Number: 36-58030.16

Version: A

SCO: 36-997

Description:

Reason:

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

Fix Description:

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

```

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

```

becomes

```

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

```

```

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

```

and

```

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

```

becomes

```

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

```

```

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

```

Applicable Reports/Requests:
SPR-122

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

Replaced Functions:

Command Impact:
None

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

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

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

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

=====
Patch Name: digestbiaserror

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

Description:

This patch fixes software problem SPR-116.

Symptom:

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

Symptom Impact:

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

Symptom Cause:

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

Fix Description:

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

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

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

The format of these 4 bits are as follows:

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

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

Applicable Reports/Requests:

SPR-116

Test Results:

reproduce --> PASS
fix --> PASS

Replaced Functions:

Command Impact:
None

Telemetry Impact:

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

Science Impact:

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

=====
Patch Name: corruptblock

Part Number: 36-58030.01

Version: A

SCO: 36-994

Description:

Reason:

This patch fixes software problem report SPR-113.

Symptom:

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

Symptom Impact:

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

Symptom Cause:

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

Fix Description:

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

The original line from chstartscirun.C is:

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

The effect of the patch changes this to:

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

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

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

Applicable Reports/Requests:
SPR-113

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

Replaced Functions:

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

at worst, a reset of the instrument.

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

Telemetry Impact:

None.

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

Science Impact:

None

=====
Patch Name: cornermean

Part Number: 36-58030.21

Version: A

SCO: 36-1017

Description:

Reason:

This patch fixes software problem report SPR-128.

Symptom:

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

Symptom Impact:

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

Symptom Cause:

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

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

```
mean = (sum + (count/2))/count
```

should be:

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

Fix Description:

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

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

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

The coded formula is:

```
mean = (sum + (count/2))/count
```

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

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

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

...

```

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

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

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

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

```

Applicable Reports/Requests:
SPR-128

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

Replaced Functions:

Command Impact:
None.

Telemetry Impact:

None.

Science Impact:

Without this patch, the corner mean values in Graded Telemetry mode may occasionally be invalid. There is a deterministic ground algorithm which can detect and 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 of sign, without further action needed by the ground science software.

=====
Patch Name: buscrash

Part Number: 36-58030.30

Version: A

SCO:

Description:

Reason:

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

Symptom:

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

Symptom Impact:

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

Symptom Cause:

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

Fix Description:

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

Applicable Reports/Requests:

SPR-140

Test Results:

reproduce --> PASS

fix --> PASS

Replaced Functions:

FepManager::loadBadPixel

Command Impact:

01/14/10
14:44:30

Flight S/W Patches, Revision E-E-F
../dist/standard-release-E-opt-E.notes

26

None.

Telemetry Impact:

None.

Science Impact:

None.

=====
Patch Name: buscrash2

Part Number: 36-58030.30

Version: B

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 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 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.

Fix Description:

Call the FepManager::isEnabled() method to check if the FEP is powered up before reading from a FEP's bias memory. This is done by patching BiasThief::checkMonitor() as follows:

```
class Test2_BiasThief : public BiasThief
{
public:
    Boolean checkMonitor(FepId fepid);
};

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

    if (fepid >= FEP_COUNT ||
        fepManager.isEnabled (fepid) == BoolFalse) {
        swHousekeeper.report(SWSTAT_FEPREC_POWEROFF, fepid);
        retval = BoolFalse;    // FEP not available or powered
    } 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 ----
    return retval;
}
```

To pass the fepId as an argument to this version of checkMonitor(), other BiasThief methods are patched inline, as follows:

```
biasthief+0x0340:
    sw      $6,36($sp)

biasthief+0x0360:
    lw      $5,36($sp)

biasthief+0x04d4:
    lw      $5,104($sp)

biasthief+0x050c:
    lw      $6,104($sp)

biasthief+0x07b0:
    move    $5,$18

biasthief+0x07f4:
    move    $6,$18
```

Applicable Reports/Requests:
SPR-142

Test Results:
reproduce --> PASS
fixTe --> PASS
fixCc --> PASS

Replaced Functions:
BiasThief::checkMonitor

Command Impact:
None.

Telemetry Impact:
None.

Science Impact:
None.

=====
Patch Name: rquad

Part Number: 36-58030.14

Version: A

SCO: 36-1000

Description:

Reason:

This patch fixes software problem report SPR-121.

Symptom:

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

Symptom Impact:

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

Symptom Cause:

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

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

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

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

This should be:

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

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

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

Fix Description:

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

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

```

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

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

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

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

```

```

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

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

                          .end   Pixel3x3::attachData(FEEven

tRec3x3 const *, EventExposure *)

                          $LM91:

```

Applicable Reports/Requests:
SPR-121

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

Replaced Functions:

Command Impact:
None

Telemetry Impact:
See SCIENCE IMPACT.

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

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

=====
Patch Name: condock

Part Number: 36-58030.17
Version: A
SCO: 36-1012

Description:

Reason:

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

Fix Description:

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

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

with an inline patch that saves R9-R12:

```
condockCtl(fp);

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


and adding the condockCtl function:

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

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

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

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

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

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

Applicable Reports/Requests:
SPR-127

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

Replaced Functions:

Command Impact:
None

Telemetry Impact:
None

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

=====
Patch Name: histogrammean

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

Description:

Reason:

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

Fix Description:

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

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

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

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

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

(c) set mean and variance sums to zero

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

(d) increment mean sum

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

(e) save stack pointer in R9

```
                .globl  fepSciTimed_lst_lc38_lc38
                .ent    fepSciTimed_lst_lc38_lc38
fepSciTimed_lst_lc38_lc38:
lc34 1403028E
lc38 48050926      addu   $9,$16,1368-16
lcec 22004010
                .end    fepSciTimed_lst_lc38_lc38
```

(f) load overclock mean sum

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

(g) load overclock variance sum

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

(h) increment R9

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

SPR-123

Test Results:

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

Replaced Functions:

Command Impact:

None

Telemetry Impact:

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

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

Science Impact:

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

TITLE: ACIS Flight Software Optional Patch Component Release Notes

DOCUMENT NUMBER: 36-58020 REVISION: E

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

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

Software Change Order: 36-1042

Build Date: Thu Nov 5 01:08:57 EST 2009
Part Number: 36-58020
Version: E
CVS Tag: release-E-opt-E

Std Number: 36-58010
Std Version: E
Std Tag: release-E
Std SCO: 36-1042

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

Description:

This is the fifth letter release of the optional patch set for the ACIS Flight Software. The purpose of this release is test these patches with the updated 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
- reportgradel - Addresses SPR 132
- smtimedlookup - Supports eventhist and ctireport
- teignore - Ignore Timed Exposure data frames
- untricklebias - Fixes SPR 133

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

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

Addressed Problem Reports:

- SPR-134
- SPR-126
- SPR-132
- SPR-133

SPR-120
SPR-124

Included Patches:

cc3x3 (4636 bytes)
ccignore (36 bytes)
compressall (2368 bytes)
ctireport1 (5452 bytes, depends on smtimedlookup)
ctireport2 (2784 bytes, depends on smtimedlookup)
deaeng (2604 bytes, depends on tlmio, conflicts with dearepl)
dearepl (556 bytes, conflicts with deaeng)
eventhist (5908 bytes, depends on smtimedlookup)
printshouse (7224 bytes, depends on tlmio)
reportgradel (816 bytes)
smtimedlookup (3712 bytes)
teignore (36 bytes)
tlmio (10312 bytes)
untricklebias (1740 bytes, depends on buscrash2)

=====
Patch Name: reportgradel

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

Conflicts:
Depends On:
Size: 816 bytes

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

Description:

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

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

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

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

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

Applicable Reports/Requests:
SPR-132

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

Replaced Functions:

PmEvent::filterEvent

Command Impact:

None.

Telemetry Impact:

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

Science Impact:

None.

=====
Patch Name: untricklebias

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

Conflicts:
Depends On: buscrash2
Size: 1740 bytes

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

Description:

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

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

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

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

When used with Standard Patch Release D or higher, containing the buscrash2 patch, the BiasThief::checkMonitor() method has been updated to test whether the fepId is powered up. This method must therefore overrides both the original checkMonitor() and the updated version loaded by the buscrash2 patch.

Applicable Reports/Requests:
SPR-133

Test Results:
patchTe --> PASS
patchAll --> PASS
patchCc --> PASS

Replaced Functions:
BiasThief::abort

```
ScienceMode::waitForBiasTrickle  
BiasThief::goTaskEntry  
BiasThief::biasReady  
BiasThief::checkMonitor
```

Command Impact:
None.

Telemetry Impact:
None.

Science Impact:
None.

=====
Patch Name: deaeng

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

Conflicts: dearepl
Depends On: tlmio
Size: 2604 bytes

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

Description:

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

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

Applicable Reports/Requests:
TOOL-PENDING

Test Results:
No Tests Specified

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

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

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

Science Impact:
N/A

=====
Patch Name: cc3x3

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

Conflicts:
Depends On:
Size: 4636 bytes

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

Description:

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

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

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

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

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

Test Results:

```
unit --> PASS
smoke --> PASS
```

Replaced Functions:

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

Command Impact:

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

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

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

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

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

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

Telemetry Impact:

This mode defines 4 new telemetry packet types.

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

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

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

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

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

Science Impact:

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

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

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

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

=====
Patch Name: tlmio

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

Conflicts:
Depends On:
Size: 10312 bytes

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

Description:

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

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

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

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

Applicable Reports/Requests:
TOOL-PENDING

Test Results:
No Tests Specified

Replaced Functions:

Command Impact:
None

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

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

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

Science Impact:

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

=====
Patch Name: compressall

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

Conflicts:
Depends On:
Size: 2368 bytes

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

Description:

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

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

Applicable Reports/Requests:
 SPR-134

 SER-none

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

Replaced Functions:
 PmCcRaw::digestRawRecord
 PmTeRaw::digestRawRecord

Command Impact:
 None.

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

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

=====
Patch Name: ccignore

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

Conflicts:
Depends On:
Size: 36 bytes

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

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

Applicable Reports/Requests:
SER-PENDING

Test Results:
smoke --> PASS

Replaced Functions:

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

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

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

=====
Patch Name: eventhist

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

Conflicts:
Depends On: smtimedlookup
Size: 5908 bytes

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

Description:

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

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

Applicable Reports/Requests:

Test Results:

smoke --> PASS
smoke2 --> PASS

Replaced Functions:

smTimedLookup3x3[3]
smTimedLookup5x5[3]

Command Impact:

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

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

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

Telemetry Impact:

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

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

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

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

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

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

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

Science Impact:

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

=====
Patch Name: printswhouse

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

Conflicts:
Depends On: tlmio
Size: 7224 bytes

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

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

Applicable Reports/Requests:
TOOL-PENDING

Test Results:
No Tests Specified

Replaced Functions:
SwHousekeeper::report

Command Impact:
None

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

Science Impact:
See the tlmio patch, 36-58030.07

=====
Patch Name: dearepl

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

Conflicts: deaeng
Depends On:
Size: 556 bytes

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

Description:

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

Applicable Reports/Requests:
TOOL-PENDING

Test Results:
No Tests Specified

Replaced Functions:

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

Command Impact:

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

Telemetry Impact:

This will produce true fiction from the DEAs.

Science Impact:

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

=====
Patch Name: teignore

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

Conflicts:
Depends On:
Size: 36 bytes

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

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

Applicable Reports/Requests:
SER-PENDING

Test Results:
smoke --> PASS

Replaced Functions:

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

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

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

=====
Patch Name: smtimedlookup

Part Number: 36-58030.24

Version: A

SCO: 36-1025

Environment: flight

Conflicts:

Depends On:

Size: 3712 bytes

Bcmd File: opt_smtimedlookup.bcnd

Pkts File: opt_smtimedlookup.pkts

Description:

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

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

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

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

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

Applicable Reports/Requests:

Test Results:

smoke --> PASS

Replaced Functions:

SmTimedExposure::terminate

SmTimedExposure::setupProcess

SmTimedExposure::setupFepBlock

Command Impact:

None.

Telemetry Impact:

None.

Science Impact:

None.

=====
Patch Name: ctireport1

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

Conflicts:
Depends On: smtimedlookup
Size: 5452 bytes

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

Description:

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

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

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

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

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

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

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

Applicable Reports/Requests:

Test Results:
smoke --> PASS

Replaced Functions:
smTimedLookupMode[4]

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

Command Impact:

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

Telemetry Impact:

The downlinked exposure and event data packets are identical in format to `exposureTeFaint` and `dataTeVeryFaint` except that their `formatTag` fields contain `TTAG_SCI_TE_REC_CTII1` and `TTAG_SCI_TE_DAT_CTII1`, respectively. When a `TTAG_SCI_TE_DAT_CTII1` is received, precursor charge data will be located in the `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: ctireport2

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

Conflicts:
Depends On: smtimedlookup
Size: 2784 bytes

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

Description:

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

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

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

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

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

Applicable Reports/Requests:

Test Results:
smoke --> PASS

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

Command Impact:

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

Telemetry Impact:

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

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

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

Science Impact:

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

```
/* =====
*
* $$Source: /acis/h3/acisfs/confignt1/patches/buscrash2/buscrash2.C,v $$
*
* Patch Name: Bus Crash Prevention, Part II
*
* Description:
* This defines C++ replacement functions for
* BiasThief::checkMonitor() and BiasThief::getBuffer()
*
* References:
*
* $$Log: buscrash2.C,v $
* $Revision 1.2 2008/08/27 18:48:53 pgf
* $Rename Test_BiasThief class to Test2_BiasThief.
* $
* $Revision 1.1 2008/08/27 17:25:39 pgf
* $Initial release.
* $$
* ===== */

#include "filesscience/biasthief.H"
#define private public
#include "filesprotocols/fepmanager.H"
#undef private
#include "filesswhouse/swhousekeeper.H"

class Test2_BiasThief : public BiasThief
{
public:
    Boolean checkMonitor(FepId fepid);
};

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

    if (fepid >= FEP_COUNT || fepManager.isEnabled (fepid) == BoolFalse) {
        swHousekeeper.report(SWSTAT_FEPREC_POWEROFF, fepid);
        retval = BoolFalse;            // FEP not available or powered
    } 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 ----
    return retval;
}
```



```
/*=====
//
//      $Source: /acis/h3/acisfs/configcntl/patches/buscrash2/buscrash2inline.S,v $
//
//      MODULE NAME: Patch to filesscience/biasthief.C
//
//      PURPOSE: Prevent bus crash on FEP powerdown
//
//      REFERENCES:
//
//      $Log: buscrash2inline.S,v $
//      Revision 1.3  2009/11/03 14:32:42  pgf
//      Preserve R6 (fepId) through call to TlmForm::waitForBuffer()
//
//      Revision 1.2  2009/10/01 15:21:24  pgf
//      Update for Standard-D Release
//
//      Revision 1.1  2008/08/27 17:25:40  pgf
//      Initial release.
//
//      COPYRIGHT: Massachusetts Institute of Technology 2008
//
=====*/

        .set noreorder
        .set nomacro
        .set noat
        .text

#####
#
# save fepid in stack on entry to getBuffer()
#
#####

        .globl biasthief_lst_0340_0340
        .ent biasthief_lst_0340_0340
biasthief_lst_0340_0340:
        sw      $6,36($sp) # 36($sp) = fepid
        .end biasthief_lst_0340_0340

#####
#
# pass fepid for call to checkMonitor() from getBuffer()
#
#####

        .globl biasthief_lst_0360_0360
        .ent biasthief_lst_0360_0360
biasthief_lst_0360_0360:
        lw      $5,36($sp) # $5 = fepid
        .end biasthief_lst_0360_0360

#####
#
# load fepid for call to checkMonitor() from trickleTeBias()
#
#####

        .globl biasthief_lst_04d4_04d4
        .ent biasthief_lst_04d4_04d4
biasthief_lst_04d4_04d4:
        lw      $5,104($sp) # $5 = fepid
        .end biasthief_lst_04d4_04d4
```

```
#####  
#  
# load fepid for call to getBuffer() from trickleTeBias()  
#  
#####  
  
        .globl biasthief_lst_050c_050c  
        .ent   biasthief_lst_050c_050c  
biasthief_lst_050c_050c:  
        lw     $6,104($sp)    # $6 = fepid  
        .end   biasthief_lst_050c_050c  
  
#####  
#  
# load fepid for call to checkMonitor() from trickleCcBias()  
#  
#####  
  
        .globl biasthief_lst_07b0_07b0  
        .ent   biasthief_lst_07b0_07b0  
biasthief_lst_07b0_07b0:  
        move  $5,$18         # $5 = fepid  
        .end   biasthief_lst_07b0_07b0  
  
#####  
#  
# load fepid for call to getBuffer() from trickleCcBias()  
#  
#####  
  
        .globl biasthief_lst_07f4_07f4  
        .ent   biasthief_lst_07f4_07f4  
biasthief_lst_07f4_07f4:  
        move  $6,$18         # $6 = fepid  
        .end   biasthief_lst_07f4_07f4
```

```
#!/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.bcml\
                 $basedir/$tools/share/opt_printswhouse.bcml\
                 $basedir/$tools/share/opt_dearepl.bcml"
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.bcml\
                 $basedir/$tools/share/opt_printshouse.bcml\
                 $basedir/$tools/share/opt_dearepl.bcml\
                 buscrash2.bcml"
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]" {
        fail "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.bcml\
                 $basedir/$tools/share/opt_printshouse.bcml\
                 $basedir/$tools/share/opt_dearepl.bcml\
                 buscrash2.bcml"

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 bias

puts ""
puts "# Starting test 1"
puts ""
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" }
}

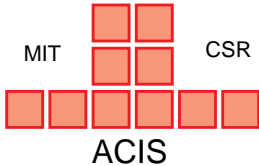
puts "# powering boards off"
send -i $cmd_id "change 0 systemConfig {
    entries = {
        itemId = 0
        itemValue = 0x0
    }
    entries = {
        itemId = 1
        itemValue = 0x0
    }
}
"
set timeout 60
expect {
    -re "bepStartupMessage.*\[\r\n]" {
        fail "Unexpected bus crash"
    }
}
```

```
-re "SWSTAT_FEPMAN_POWEROFF.*\[\r\n]" {
}
timeout {
    fail "Power-down Failure"
}
}
puts "# Powered off"

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

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"
```

		ENGINEERING CHANGE ORDER		<u>ECO No.</u> <u>36-1043</u>
CENTER FOR SPACE RESEARCH MASSACHUSETTS INSTITUTE OF TECHNOLOGY				
DWG. NO.	NEW REV.	DRAWING TITLE		
36-58021.04	F	Flight Software Patch Release E-E-F Certification		
REASON FOR CHANGE: Certification of standard patch release E, which includes the updated <i>buscrash2</i> patches, along with the same optional patches that were certified in release D-D-E, <i>i.e.</i> , <i>smtimedlookup</i> , <i>compressall</i> , <i>eventhist</i> , <i>cc3x3</i> , and <i>untricklebias</i> .				
DESCRIPTION OF CHANGE: Three optional patch combinations are certified as release E-E-F: (a) <i>cc3x3</i> , <i>eventhist</i> , and <i>smtimedlookup</i> . (b) <i>cc3x3</i> , <i>eventhist</i> , <i>compressall</i> , and <i>smtimedlookup</i> . (c) <i>cc3x3</i> , <i>eventhist</i> , <i>compressall</i> , <i>untricklebias</i> , and <i>smtimedlookup</i> . The certification tests are taken from these specific combinations of the optional release E patches, with the full set of standard patches, release E.				
	SIGNATURE	DATE	REMARKS:	
ORIGINATOR	RFG	01/06/10	Released	
MECHANICAL				
ELECTRICAL				
SOFTWARE				
STRUCTURE				
FABRICATION				
SCIENCE				
SYSTEMS ENG.				
QUALITY				
PROJ. ENGINEER				
DEPUTY PM				
PROJ. MANAGER				

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

DOCUMENT NUMBER: 36-58021.03 REVISION: F

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

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
D	36-1036	Certify CC3x3/EventHist/smTimedL	RFG	08/09/2007
E	36-1039	Certify Rev. D Standard and Rev.	RFG	09/29/2009
F	36-1043	Certify Rev-E-Opt-E patches	RFG	01/06/2010

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

Software Change Order: 36-1043

Build Date: Fri Nov 6 14:53:35 EST 2009
Part Number: 36-58021.03
Version: F
CVS Tag: cc3x3+eventhist-E-E-F

Std Number: 36-58010
Std Version: E
Std Tag: release-E
Std SCO: 36-1042

Opt Number: 36-58020
Opt Version: E
Opt Tag: release-E-opt-E
Opt SCO: 36-1042

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

Description:

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

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

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

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

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

comparison may be developed to eliminate this manual step.

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

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

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

Included Patches:

eventhist
cc3x3
smtimedlookup

Test Support Patches:

printswhouse
dearepl
tlmio

Test Results:

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

01/14/10
14:37:15

Flight S/W Patches, Revision E-E-F
../../certsrc/cc3x3+eventhist+compressall.notes

1

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

DOCUMENT NUMBER: 36-58021.03 REVISION: F

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

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
D	36-1036	Certify CC3x3/EventHist/smTimedL	RFG	08/09/2007
E	36-1039	Certify Rev. D Standard and Rev.	RFG	09/29/2009
F	36-1043	Certify Rev-E-Opt-E patches	RFG	01/06/2010

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

Software Change Order: 36-1043

Build Date: Fri Nov 6 16:37:18 EST 2009
Part Number: 36-58021.03
Version: F
CVS Tag: cc3x3+eventhist+compressall-E-E-F

Std Number: 36-58010
Std Version: E
Std Tag: release-E
Std SCO: 36-1042

Opt Number: 36-58020
Opt Version: E
Opt Tag: release-E-opt-E
Opt SCO: 36-1042

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

Description:

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

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

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

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

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

comparison may be developed to eliminate this manual step.

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

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

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

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

Included Patches:

eventhist
cc3x3
compressall
smtimedlookup

Test Support Patches:

printswhouse
dearepl
tlmio

Test Results:

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

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

DOCUMENT NUMBER: 36-58021.03 REVISION: F

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

LETTER	SCO NO.	DESCRIPTION	APPROVED	DATE
D	36-1036	Certify CC3x3/EventHist/smTimedL	RFG	08/09/2007
E	36-1039	Certify Rev. D Standard and Rev.	RFG	09/29/2009
F	36-1043	Certify Rev-E-Opt-E patches	RFG	01/06/2010

=====
Title: ACIS untricklebias, eventhist, cc3x3, compressall, smtimedlookup Patch Certification
Release Notes for Version F

Software Change Order: 36-1039

Build Date: Fri Nov 6 20:26:58 EST 2009
Part Number: 36-58021.03
Version: F
CVS Tag: cc3x3+eventhist+compressall+untricklebias-E-E-F

Std Number: 36-58010
Std Version: E
Std Tag: release-E
Std SCO: 36-1042

Opt Number: 36-58020
Opt Version: E
Opt Tag: release-E-opt-E
Opt SCO: 36-1042

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

Description:

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

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

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

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

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

comparison may be developed to eliminate this manual step.

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

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

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

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

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

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

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

Included Patches:

untricklebias
eventhist
cc3x3
compressall
smtimedlookup

Test Support Patches:

```
printswhouse  
dearepl  
tlmio
```

Test Results:

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