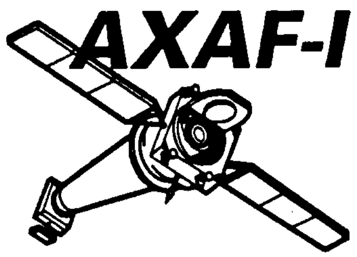
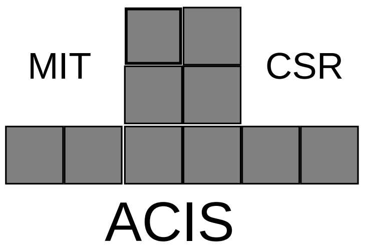
36-54003 Rev. A
July 21, 1999
NAS8-37716
DR SDM05





Advanced X-ray Astrophysics Facility

AXAF - I CCD Imaging Spectrometer

ACIS Science Instrument Software User's Guide

Submitted to:

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

Submitted by:

Center for Space Research
Massachusetts Institute of Technology
Cambridge, MA 02139

AXAF-I CCD Imaging Spectrometer
(ACIS)

ACIS Science Instrument Software User's Guide

36-54003 Rev. A

DR SDM05

Contract # NAS8-37716

July 21, 1999

Submitted to:

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

Submitted By:

Massachusetts Institute of Technology
Center for Space Research
77 Massachusetts Avenue
Cambridge, MA 02139

Approvals:

Dr. Peter Ford	Dr. William Mayer
Software Project Manager	Project Manager
Massachusetts Institute of Technology	Massachusetts Institute of Technology

CSR	**MASSACHUSETTS INSTITUTE OF TECHNOLOGY** **CENTER FOR SPACE RESEARCH** **CAMBRIDGE, MASSACHUSETTS 02139**	
REVISION LOG	**TITLE:** **ACIS Science Instrument Software User's Guide**	**DOC. NO.** **36-54003 Rev. A**

Revision	Date (mm/dd/yy)	ECO No.	Page(s) Affected	Reason	Approval
01	5/30/97	36-923	All	Initial Skeleton	
A	9/11/97	36-957	All	Initial Version	

Table of Contents

List of Figures

List of Tables

ACIS Science Instrument Software User's Guide

MIT Center for Space Research

36-54003 Rev. A

July 21, 1999

1.0 Introduction

The AXAF-I CCD Imaging Spectrometer (ACIS) Science Instrument Software (SIS) is being developed by the Massachusetts Institute of Technology, Center for Space Research (MIT-CSR) as part of the ACIS Digital Processor Assembly (DPA). The DPA resides on-board the Advanced X-ray Astrophysics Facility - Imaging (AXAF-I). The DPA Science Instrument Software is responsible for acquiring and processing image data from the ACIS CCD Imaging Spectrometer and transferring the processed data to the AXAF-I Command and Telemetry Unit (CTU), which is then responsible for sending the information to the ground.

1.1 Purpose

The ACIS Science Instrument Software User's Guide describes the operation of the key functions of the instrument software.

1.2 Scope

This document applies to the detailed design of the ACIS DPA Science Instrument Software. It does not provide information for the Ground Support Software (GSS), which is maintained separately as part of the Electronic Ground Support Equipment (EGSE).

This document supplies information applicable to SDM05 from the original contract, and to DM29 from MM8075.1.

By mutual agreement, MSFC Software Management and Development Requirements Manual MM8075.1, which supersedes MA-001-006-2H, forms the basis for this document.

1.3 References

This specification relies on a set of existing documentation. The following table lists these documents.

TABLE 1. Reference Documents

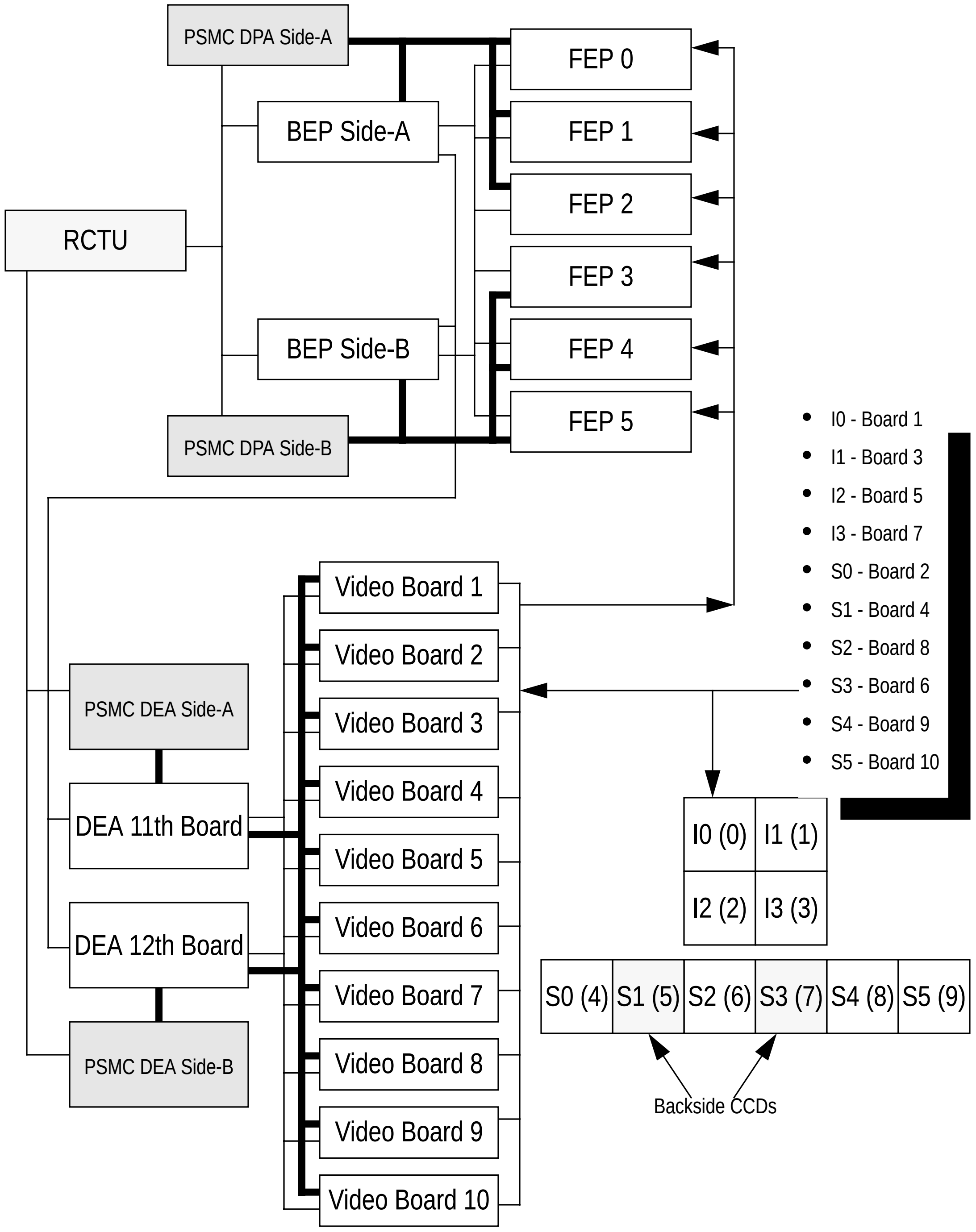
Part Number	Version	Title
MSFC MM 8075.1	January 22, 1991	MSFC Software Management and Development Requirements Manual
MIT-CSR 36-01103	I	ACIS Science Instrument Software Requirements Specification
MIT-CSR 36-01502	C	ACIS Technical Analyses and Models: ACIS Hardware Specification and System Description
NU910701	1991	Nucleus RTX Reference Manual from Accelerated Technology, Inc.
NU910702	1991	Nucleus RTX Internals Manual from Accelerated Technology, Inc.
MIT-CSR 36-01410	January 15, 1997	ACIS Instrument Protocol and Command List
MIT-CSR 36-53204.0204	K	ACIS IP&CL Software Structure Definitions
MIT-CSR 36-02205	C	DPA/DEA Interface Control Document
MIT-CSR 36-55001	Rev 3.1	ACIS Test Tools
MIT-CSR 36-54002.08	A	ACIS Flight Software Release 1.5
MIT-CSR 36-56102	01	Huffman Coding of ACIS Pixel Data

2.0 Overview

2.1 Instrument Hardware Overview

Figure 1 illustrates the overall hardware architecture of ACIS.

FIGURE 1. Hardware Architecture



- RCTU - Remote Command and Telemetry Unit

  This provides the command and telemetry interface between ACIS and the spacecraft.

- PSMC - Power Supply and Mechanism Controller

  This provides power to the ACIS Digital Processor Assembly (DPA), which contains the Back End Processors and Front End Processors, and to the Digital Electronics Assembly (DEA), which contains the CCD Video Controller Boards and two interface boards. It is also responsible for controlling the ACIS doors, vent valves, etc. (not shown).

- BEP - Back End Processor

  ACIS has two redundant Back End Processors, responsible for the overall control of the instrument, including command and data handling. Although both BEPs are usually powered at the same time, only one BEP is active at a time[1]. Each BEP is powered by one side of the PSMC DPA Power Supply.

- FEP - Front End Processors

  ACIS has 6 Front End Processors which are responsible for acquiring digitized image data from the CCD Video Controllers, and for detecting candidate X-ray events from these images. For a normal science observation, data from each CCD used for the observation is processed by any one of the 6 FEPs (NOTE: it is possible, however, to have more than one FEP process data from the same CCD). FEPs 0 - 2 are powered by A-side of the PSMC DPA Power Supply, and FEPs 3 - 5 are powered by the B-side of the supply. In order to process data from 6 CCDs simultaneously, both sides of the PSMC DPA Power Supply must be on.

- DEA 11th Board

  This is the primary interface board for the Detector Electronics Assembly, It contains power switching logic for the DEA Video Boards, a Focal Plane temperature controller, and some housekeeping logic. These components are powered by the primary side of the PSMC DEA Power Supply. This board also contains a set of relays used to switch power on the video boards to the current PSMC DEA Supply. Unlike the DPA, only one side of the PMSC DEA Power Supply may be on at a time.

- DEA 12th Board

  This board is a backup interface board for the DEA. It is identical to the 11th board except that it does not have the power-switching relays, nor does it have the housekeeping logic. This board is powered by the redundant side of the PSMC DEA Power Supply.

- Video Boards

  ACIS has 10 DEA Video Boards. Each board is hard-wired to a particular CCD (see diagram for the mapping). Each video board contains a sequencer (loaded by the BEP) which is used to transfer charge within the CCDs. Each also contains digitization circuits to convert the analog CCD data into a digital form, which is then fed to the FEPs.

1. It is possible to logically activate both BEPs at the same time, but this causes non-damaging contention on some of the interfaces, such as the telemetry interface.

The ACIS Mongoose processors are all configured by hardware to use “little-endian” byte ordering. For example, the 32-bit value 0x12345678 is stored as bytes in RAM as follows:

Virtual Address	**Byte Value**
0	0x78
1	0x56
2	0x34
3	0x12

By convention, bits within a word are numbered with the least-significant bit as bit number 0. This is consistent with the “MIPS Programmers Reference Guide.”

Unless otherwise specified, all signed values use two’s complement representation.

2.2 Instrument Software Overview

The ACIS Instrument Software runs on two types of processors within the system: the Back End Processor (BEP) and the set of Front End Processors (FEP). The BEP loads its software either from EEPROM or from the uplink command channel. Once up and running, the BEP can enable power to the FEPs, and load software into the FEPs using a shared-memory interface.

The BEPs run a single-processor preemptive, multi-tasking kernel which allows the BEP to be performing more than one task at a time. For example, a BEP can be processing science data, while acquiring DEA housekeeping data, while performing a memory dump (BTW: This is a very useful feature when trying to diagnose certain classes of problems). The telemetry produced by the BEP is organized into data packets, which appear in the Science Data portion of the telemetry stream. Each packet is preceded by a 32-bit synchronization pattern, and contains a length field indicating the number of words in the packet. This allows telemetry produced by the several tasks running on the BEP to be merged into the telemetry stream (for example, memory dump telemetry packets may be mixed in with science event data packets).

The following lists the BEP's tasks, grouped according to their priority (highest priority, 51, is first, and lowest priority listed, 55, is last. Tasks with the lowest priority number have the highest run-time priority):

TABLE 2. BEP Tasks

Task Name	Pri.	Command/Telemetry Types	Role
`TaskMonitor`	51	*`Not commandable`*	Perform aliveness tests of the other tasks. Allows the watchdog timer to reset the BEP if a task fails to respond to a query within 8 minutes.
`CmdManager`	52	*`All software commands executed or routed by this task`*	Interpret and dispatch uplinked commands
`SystemConfiguration`	53	*`Radiation Flag and Change System Configuration`*	Respond to changes in configuration table and monitor the radiation flag
`SwHousekeeper`	53	*`Not commandable. Produce software housekeeping packets`*	Collect and periodically report software statistics. Update LED bi-levels to reflect instrument's operating state.
`DeaHousekeeper`	53	*`Start/Stop DEA Housekeeping`*	Periodically collect and report DEA housekeeping values.
`MemoryServer`	54	*`Dump commands, Read, Write BEP/FEP/PRAM/SRAM commands, Execute BEP/FEP commands`*	Handle read (including dump), write, and execute memory commands
`BiasThief`	55	*`Affected by start/stop science/bias-only commands, via ScienceManager`*	Trickle the contents of the computed CCD bias maps to telemetry.
`ScienceManager`	55	*`Execute start/stop science runs and bias-only science runs.`*	Perform science run, including hardware setup, parameter dumps, bias computation and data processing

Each FEP, on the other hand, runs a single main thread, with a single interrupt handler to count the arrival of images produced by the DEA. The FEPs are only commanded by the software running on the active BEP. This single thread performs the high-performance portion of the science data processing, and polls its shared memory interface for requests from the BEP. In general, the FEPs are only asked to do one thing at a time by the BEPs.

2.3 Commands and Telemetry

ACIS is commanded via its Remote Command and Telemetry Unit (RCTU). This unit provides various types of commands and telemetry. The ACIS hardware is commanded via high-level pulse commands to the ACIS Power Supply and Mechanism Controller (PSMC) and 16-bit serial digital commands to the ACIS hardware serial port. Commands to the ACIS software are contained within a series of 16-bit serial command words, known as command packets, which are addressed to the ACIS software serial port. For the purposes of this document, “hardware commands” will refer to either high-level pulse commands addressed to the PSMC, or serial digital commands addressed to the ACIS hardware serial port. “Software commands” will always refer to command packets addressed to the ACIS software serial port.

The ACIS hardware provides engineering telemetry which is read by the spacecraft and placed in fixed places within the spacecraft telemetry stream. The ACIS software produces telemetry as a stream of 32-bit words, packed and placed into the spacecraft telemetry stream allocated for ACIS science telemetry. These packets may or may not be separated by a variable number of fill bytes (0xb7), placed into the stream by the ACIS hardware when a packet is not being transferred.

For details on the content and format of the ACIS hardware commands and telemetry, refer to the ACIS Instrument Protocol and Command List (MIT-CSR 36-01410). For details on the content and format of the ACIS software command and telemetry, refer to the ACIS IP&CL Software Structure Definitions (MIT-CSR 36-53204.0204).

3.0 Instrument Maintenance Activities

This section describes the various activities used to maintain the ACIS instrument software.

3.1 Starting the Software

This section describes how the ACIS instrument software is started.

3.1.1 Back End Processor Power-On

This section describes how to power-on the ACIS Digital Processor Assembly (DPA) Back End Processors (BEP). An ACIS BEP must be powered for the instrument software to run.

Description

On ACIS, there are two Back End Processors, a Side-A processor and a Side-B processor. The Side-A processor is powered by the Side-A DPA Power Supply and Mechanism Controller (PSMC), whereas the Side-B Back End Processor is powered by the Side-B DPA PSMC. Each BEP is powered by issuing an enable PSMC command to the appropriate side, followed by a power-on command to that side. When powered, the BEP hardware and software will perform a power-on boot (see Section 3.1.3.1).

Since the Side-A PSMC also supplies power to three of the Front End Processors (FEPs 0,1 and 2) and the Side-B PSMC supplies power to the remaining three FEPs (3, 4 and 5), both BEP processors are normally powered at the same time. However, only one BEP may be "active" at a time. At power-on, the Side-A BEP is the "active" BEP, while the Side-B BEP's processor is held in a reset state and its external I/O logic is inactive. See Section 3.1.2 for a description on how to switch which is the active BEP and for warnings concerning FEP power when switching between BEPs.

NOTE: If BEP B is powered up, and BEP A is not already powered, BEP B will not be the selected BEP, and the serial digital telemetry and bi-level telemetry from the DPA will float. Lab experience has shown that the floating lines tend to a logical '1'. See Section 3.1.2 for the procedure to activate BEP B.

Commands

The following are the commands issued to power-on the BEPs. See Section 3.2 for commands to power-off the DPA's BEPs. Also, see Section 3.3.3 for commands needed to enable and power on and off the Detector Electronics Assembly.

TABLE 3. Power On Command Sequence

Command[a] Mnemonic	Command Type	Description
Power Up Side-A BEP		
1DPPSAEN	Pulse to PSMC Side-A	Enable (but don't power on) the PSMC DPA Side-A
1DPPSAON	Pulse to PSMC Side-A	Power-On the PSMC DPA Side-A Power
Power Up Side-B BEP		
1DPPSBEN	Pulse to PSMC Side-B	Enable (but don't power on) the PSMC DPA Side-B
1DPPSBON	Pulse to PSMC Side-B	Power-On the PSMC DPA Side-B Power

a. Hardware command mnemonics are described in the ACIS Instrument Protocol and Command List (MIT-CSR 36-01410)

Engineering Telemetry

The following telemetry items in the engineering portion of the telemetry stream indicate when power is applied to the BEPs. The following assume that BEP A is being powered first.

TABLE 4. Power On Engineering Telemetry

Telemetry[a] Mnemonic	Telemetry Type	Value	Description
While the BEPs are off and PSMC disabled			
1DPPSAEN	Serial Digital	0	Verifies that the DPA A-side Power Supply is disabled.
1DPPSBEN	Serial Digital	0	Verifies that the DPA B-side Power Supply is disabled.
1DPPSAON	Serial Digital	0	Verifies that the DPA A-side Power Supply is off.
1DPPSBON	Serial Digital	0	Verifies that the DPA B-side Power Supply is off.
1TSERXT	SW Serial Digital	undefined	The software serial digital telemetry out of a BEP while powered off is undefined, however lab experience shows the interface almost always floats to 1's, giving 8-bit data values of 0xff
1STAT[0..7]ST	HW Bi-level	undefined	The bi-level telemetry of the BEP when powered off is undefined, however, lab experience shows the interface almost always floats to 1's.
After a pair of PSMC DPA Enable commands have been issued			
1DPPSAEN	Serial Digital	1	Verifies that the DPA A-side Power Supply has been enabled

TABLE 4. Power On Engineering Telemetry

Telemetry[a] Mnemonic	Telemetry Type	Value	Description
1DPPSBEN	Serial Digital	1	Verifies that the DPA B-side Power Supply has been enabled
After a pair of PSMC DPA Power-On commands have been issued			
1DPPSAON	Serial Digital	1	Verifies that the DPA A-side Power Supply has been turned on.
1DPPSBON	Serial Digital	1	Verifies that the DPA B-side Power Supply has been turned on.
1TSERXT	SW Serial Digital	fill = 0xb7	Once a BEP is powered and enabled (side-A defaults to enabled), the hardware places a fill pattern into the software serial telemetry stream. This fill pattern appears whenever the software is not transmitting telemetry packets.
1STAT0ST	Bi-level:	Vary	When the BEP is first powered, the software sets the LED bi-levels to 0xf. As the system boots, it walks the LEDs through a series of values so that if the instrument gets “stuck”, the ground can tell where. The LED value sequence is described in more detail in Section 3.1.3 and Section 3.1.4 . Once running, the Flight Software’s Housekeeping Task updates these bits to indicate the current state of the instrument.
1STAT1ST	“LED” bits 0-3	Vary	
1STAT2ST		Vary	
1STAT3ST		Vary	
1STAT4ST	Bi-level: BEP Id	0 (BEP A)	If BEP A is powered on first, it will be, by default, the selected BEP and drive the bi-level to 0.
1STAT5ST	Bi-level: CPU Not Reset	1	Since, by default, BEP A is not held in a reset state when first powered, the BEP will drive this bi-level to a 1.
1STAT6ST	Bi-level: FIFO Not Full	1	After a power-on, the BEP’s command FIFO is reset by the hardware, and therefore will not be full, but instead is empty.
1STAT7ST	Bi-level: FIFO Not Empty	0	

a. Engineering telemetry mnemonics are described in the ACIS Instrument Protocol and Command List (MIT-CSR 36-01410)

Science Telemetry

When a BEP has power, and the software is not transmitting any telemetry packets, the hardware supplies a single-byte fill pattern, 0xb7 (hexadecimal). Once the software is initialized after a power-on, it outputs a BEP Startup Message telemetry packet, and then approximately once every 64 seconds outputs a BEP Software Housekeeping telemetry packet. The Startup Message packet is described in more detail in Section 3.1.3 . The Software Housekeeping packet is described in more detail in Section 3.4.3 .

TABLE 5. Power On Science Telemetry

Tag	Field	Value	Description
After the PSMC DPA Power-On command has been issued			
(NOTE: If BEP B, a select command must also be issued (see Section 3.1.2)			
TTAG_STARTUP	-	-	After the BEP software is initialized, it issues a BEP Startup Message packet.
TTAG_SW_HOUSE	-	-	Once running, the flight software emits 1 software housekeeping packet every 64 seconds or so.

Warnings

1. Power Reset of Tables

   A power-on will reset the state of the all hardware within ACIS, and reset all internal software structures. Any code, tables and data loaded into RAM will be lost.

2. Power Reset of Uplink Flag

   If the instrument was in a “load-from-uplink” state prior to being powered-off, the state of “load-from-uplink” flag will be lost. When the instrument is subsequently powered on, it will load code from its EEPROMs and execute the loaded code.

3. BEP Select Confusion

   If BEP B is powered on and selected while BEP A is off, and then BEP A is powered on, both BEPs will be selected and will attempt to drive the telemetry interface. This will not hurt the hardware, but will make software telemetry impossible to understand. A “Select BEP” command to choose either A or B at this point will solve the problem.

3.1.2 Selecting which BEP is active

This section describes how to select which BEP should be active, and drive the interface hardware.

Description

By default, after a power-on, both BEP A and BEP B will assume that the active BEP is BEP A. To select BEP B, one must explicitly issue a hardware serial digital command to select it. In order to avoid confusion, and reduce the possibility of problems, it is recommended, that a select command be issued whenever one or both BEPs are first powered on (NOTE: see Warnings concerning FEP power when selecting BEPs). Whenever a BEP is de-selected, its processor is held in a reset state and the software on the BEP is halted. If the BEP is then selected, the reset is released, and the BEP software will proceed to boot and run.

Commands

The following are the commands issued to select a particular BEPs.

TABLE 6. BEP Selection Commands

Command Mnemonic	**Command Type**	**Value**	**Description**
Select BEP A			
1BSELICL	HW Serial Digital	v=0	Select BEP A by issuing the Select BEP command with data bit 0 (v) set to 0.
Select BEP B			
1BSELICL	HW Serial Digital	v=1	Select BEP B by issuing the Select BEP command with data bit 0 (v) set to 1.

Engineering Telemetry

The currently selected BEP is indicated by a bi-level in the engineering telemetry.

TABLE 7. Selected BEP in Engineering Telemetry

Telemetry Mnemonic	**Telemetry Type**	**Value**	**Description**
When BEP A is the selected BEP:			
1STAT4ST	Bi-level: BEP Id	0	If BEP A is selected, this bi-level will have a value of 0.
When BEP B is the selected BEP:			
1STAT4ST	Bi-level: BEP Id	1	If BEP B is selected, this bi-level will have a value of 1. NOTE: If the selected BEP is powered off, then the bi-level will float, usually to a ‘1’. This may lead to confusion under some conditions

Science Telemetry

If the chosen BEP is not already selected, then the select command releases the BEP’s reset line, causing the instrument software to load and run. Depending on the state of the BEP’s Boot Modifier Flag at the time of the select command is received, the instrument software’s bootstrap loader will either (a) load code from EEPROM (see Section 3.1.3) and will produce its standard Startup Message telemetry packet, and subsequent Software Housekeeping telemetry packets, or (b) will load code from a series of software serial command packets (Section 3.1.4) which may or may not produce any science telemetry, depending on what code is loaded and run.

Warnings

1. BEP already selected

   If the BEP is already selected, it will not be re-booted.

2. BEP not already selected

   If the BEP is not currently the selected BEP, it will be held in a reset state. Once selected, the BEP will re-boot.

3. Both BEP A and B are selected

   If BEP A is powered on, or power-cycled while BEP B is selected, BEP A will assume that it is selected. This will cause contention and confusion on the telemetry interface. To correct the problem, re-issue the most recent select command after powering on BEP A. To avoid the problem, issue a Select BEP A command prior to powering on BEP A, or ensure that BEP B is un-powered when BEP A is turned on.

4. Neither BEP A nor B are selected

   If BEP A is powered on while BEP B is powered off, and is then told that BEP B is selected, no BEP will be driving the interfaces.

5. FEPs may be held "on"

   The Side-A and Side-B BEPs can both be commanded to enable and disable power to the FEPs. If both BEPs are powered and the previously selected BEP has power enabled to some of the FEPs, the currently selected BEP will not be able to power-off those FEPs. See Section 3.3.2 for more detail.

3.1.3 Loading from EEPROM

Each ACIS Back End Processor is capable of loading its software from its Read-Only-Memory, or from its uplink channel. This section describes the former.

3.1.3.1 Power-On Boot

Description

When a Back-End Processor (BEP) is powered on and selected, it executes a bootstrap loader residing within its Read-Only Memory (ROM) (prior to launch, this is Electrically Erasable and Programmable Read-Only Memory, EEPROM). This loader copies the bulk of the instrument software from the ROM into the BEPs RAM, and transfers control to the loaded code. The loaded code detects that the instrument has been started by a power on and as a result takes the following actions:

1. Copies the bulk of code and initialized data from ROM into I-cache and D-cache RAM
2. Resets the Patch List to "empty"
3. Copies Parameter Blocks (Te, Cc, 2d, 1d, Dea) from ROM into RAM
4. Copies Bad Pixel and Column Maps from ROM into RAM

5. Copies Huffman Tables from ROM into RAM
6. Copies System Configuration Settings from ROM into RAM
7. Issues a Startup Message telemetry packet
8. All DEA Video boards will be powered off (as per the default ROM System Configuration Table settings)
9. All Front End Processors (FEP) will be powered off, unless held on by the other BEP (as per the default ROM System Configuration Table settings)
10. Focal Plane temperature will be reset to its default ROM System Configuration Table value

Commands

To power on and run Side-A BEP:

```
1DPPSAEN     Enable Side-A DPA Power Supply
1DPPSAON     Power-On Side-A DPA Power Supply
1BSELICL(v=0)Select Side-A BEP to ensure no contention with BEP B.
```

To power on and run Side-B BEP:

```
1DPPSBEN     Enable Side-B DPA Power Supply
1DPPSBON     Power-On Side-B DPA Power Supply
1BSELICL(v=1)Select Side-B BEP
```

Engineering Telemetry

Command Verifiers

```
1DPPSAEN  Verifies DPA Power Supply A is enabled
1DPPSAON  Verifies DPA Power Supply A is on
1DPPSBEN  Verifies DPA Power Supply B is enabled
1DPPSBON  Verifies DPA Power Supply B is on
1STAT4ST  0 - indicates BEP A is selected,
          1 - indicates BEP B is selected
(NOTE: See Warnings in Section 3.1.2 )
```

Boot LED Sequence

```
LED values are listed as: (1STAT3ST, 1STAT2ST, 1STAT1ST, 1STAT0ST)
LED_BOOT_RESET   (1,1,1,1) - BEP has just reset
LED_RUN_PATCH    (1,0,0,1) - Resetting patch list
LED_RUN_STARTUP  (1,0,0,0) - Starting the multi-tasking executive
(see Warnings)

After an initial 64 second wait, the BEP’s Software Housekeeping task
will alternate the LEDs between two values every 64 seconds indicating
whether or not a science run is in progress, and that the BEP was not
restarted due to a watchdog reset. If a science run is not in progress,
the LEDs will alternate between:
LED_RUN_IDLE_A   (0,1,1,0)
LED_RUN_IDLE_B   (0,1,1,1)

If a science run is activated, the LEDs will change to:
```

```
LED_RUN_SCIENCE_A(0,1,0,0)
LED_RUN_SCIENCE_B(0,1,0,1)
```

Science Telemetry

Startup Message from Power On Boot

```
bepStartupMessage: TTAG_STARTUP
{
   bepTickCounter   < 10 (if not, SEU or hardware error)
   version          = 11 (if not, SEU or hardware error)
   lastFatalCode    = random
   lastFatalValue   = random
   watchdogFlag     = 0 (if 1, SEU or hardware error)
   patchValidFlag   = 0 (if 1, SEU or hardware error)
   configFlag       = 0 (if 1, SEU or hardware error)
   parametersFlag   = 0 (if 1, SEU or hardware error)
   warmBootFlag     = 0 (if 1, was a Warm-Boot, SEU or hardware error)
}
```

Software Housekeeping

```
swHousekeeping: TTAG_SW_HOUSE
{
   startingBepTickCounter  = varying
   endingBepTickCounter    = startingBepTickCounter + 640
   statistics[] =
   {
      swStatisticId= SWSTAT_VERSION
      count        = 1
      value        = 11
   }
   {
      swStatisticId= SWSTAT_TIMERCB_INVOKE
      count        = 1
      value        ~= 640 (~10 timer ticks/second)
   }
}
```

Warnings

1. Power-On Boot is always a Cold-Boot

   A power-on boot resets the BEPs hardware flags, such as the Warm Boot flag, and the Load-from-Uplink flag, making it impossible to perform a Warm Boot or Load-from-Uplink Boot using just a power-on command sequence. See Section 3.1.4 for a description of how to perform a Load-from-Uplink Boot, and Section 3.1.3.3 for a description of a Warm-Boot.

2. “As-launched” settings (including temperature control)

   All Patches are lost and all Parameter Blocks, System Configuration Settings, and Huffman Compression Tables will be reset to their “as-launched” values. Note: This affects power-consumption of the instrument and the focal-plane temperature control set-point.

3. Memory Decay

Don’t count on any unused portions of BEP or FEP memory remaining intact. The powered-off memory will “decay” over time.

4. See Warnings in Section 3.1.1 and Section 3.1.2
5. “Stuck” LED Values

   The initial boot LED sequence will change faster than the sample rate of the telemetry system. Unless something goes wrong, don’t expect to see every code in the sequence as the instrument comes up. However, if the LED values “stick” to the following values for more than one science telemetry frame, or codes persist which are not listed, there is probably a problem with the instrument or command sequence to the instrument:

```
       LED_BOOT_RESET   (1,1,1,1) - BEP has just reset
       LED_RUN_PATCH    (1,0,0,1) - Copying patches or resetting patch list
```

   If the following LED code “sticks” for more than 64 seconds, there may also be a problem:

```
       LED_RUN_STARTUP  (1,0,0,0) - Starting the multi-tasking executive
```

3.1.3.2 Cold Boot

Description

Cold-Boots are performed in order to reset and re-initialize a BEP into an “as-launched” state, without resorting to cycling the power. In order to perform a cold-boot, the desired BEP must be powered-on, and selected, with its Load-from-Uplink flag cleared (default condition after power-on), and its Warmboot flag cleared (default condition after power-on). It then must receive a BEP Halt command, followed by a BEP Run command. This will cause the BEP CPU to reset and start executing the loader in its ROM. From then on, the boot process of the BEP appears just as a power-on boot. The loader and startup software will reset the Patch List, and reload the default tables from ROM.

Like a Power-On boot, a cold-boot of the BEP:

1. Copies the bulk of code and initialized data from ROM into I-cache and D-cache RAM
2. Resets the Patch List to “empty”
3. Copies Parameter Blocks (Te, Cc, 2d, 1d, Dea) from ROM into RAM
4. Copies Bad Pixel and Column Maps from ROM into RAM
5. Copies Huffman Tables from ROM into RAM
6. Copies System Configuration Settings from ROM into RAM
7. Issues a Startup Message telemetry packet
8. All DEA Video boards and Front End Processors will be powered off (as per the default ROM System Configuration Table settings)
9. All Front End Processors (FEP) will be powered off, as per the default ROM System Configuration Table settings, unless held on by the other BEP (see Section 3.3.2)

10. Focal Plane temperature will be reset to its default ROM System Configuration Table value

Commands

```
1BMODIBM (v=0)Set BootModifier Off
1WRMBTSB (v=0)Set Warmboot Off
1RSETIRT (v=1)Halt BEP
1RSETIRT (v=0)Run BEP
```

Engineering Telemetry

Boot LED Sequence (for detail, refer to Engineering Telemetry in Section 3.1.3.1)

```
LED_BOOT_RESET
LED_RUN_PATCH
LED_RUN_STARTUP
LED_RUN_IDLE_A
LED_RUN_IDLE_B
LED_RUN_SCIENCE_A
LED_RUN_SCIENCE_B
```

Science Telemetry

Startup Message

```
bepStartupMessage: TTAG_STARTUP
{
   bepTickCounter   < 10 (if not, SEU or hardware error)
   version          = 11 (if not, SEU or hardware error)
   lastFatalCode    = random
   lastFatalValue   = random
   watchdogFlag     = 0 (if 1, SEU or hardware error)
   patchValidFlag   = 0 (if 1, SEU or hardware error)
   configFlag       = 0 (if 1, SEU or hardware error)
   parametersFlag   = 0 (if 1, SEU or hardware error)
   warmBootFlag     = 0 (if 1, was a Warm-Boot, SEU or hardware error)
}
```

Software Housekeeping

```
swHousekeeping: TTAG_SW_HOUSE
{
   startingBepTickCounter  = varying
   endingBepTickCounter    = startingBepTickCounter + 640
   statistics[] =
   {
      swStatisticId= SWSTAT_VERSION
      count        = 1
      value        = 11
   }
   {
      swStatisticId= SWSTAT_TIMERCB_INVOKE
      count        = 1
      value        ~= 640 (~10 timer ticks/second)
   }
}
```

Warnings

1. A Cold Reset is initiated upon receipt of a BEP Select Command

   If an un-selected BEP is selected, and its Load-from-Uplink and Warmboot flags are de-asserted, the BEP will perform a cold-boot, losing any previously stored Patch List, Parameter Blocks, etc.

2. "As-launched" settings (including temperature control)

   All Patches, Parameter Blocks, System Configuration Settings, and Huffman Compression Tables will be reset to their "as-launched" values. Note: This affects power-consumption of the instrument and the focal-plane temperature control set-point.

3. Preserved Memory

   Unused portions of BEP memory will remain intact.

4. "Stuck" LED Values

   The initial boot LED sequence will change faster than the sample rate of the telemetry system. Unless something goes wrong, don't expect to see every code in the sequence as the instrument comes up. However, if the LED values "stick" to the following values for more than one science telemetry frame, or codes persist which are not listed, there is probably a problem with the instrument or command sequence to the instrument:

   ```
   LED_BOOT_RESET   (1,1,1,1) - BEP has just reset
   LED_RUN_PATCH    (1,0,0,1) - Copying patches or resetting patch list
   ```

   If the following LED code "sticks" for more than 64 seconds, there may also be a problem:

   ```
   LED_RUN_STARTUP  (1,0,0,0) - Starting the multi-tasking executive
   ```

3.1.3.3 Warm Boot

Description

A Warm-Boot is used to reset a BEP's hardware, re-load its code and data from ROM and install its patch list while attempting to maintain the already loaded configuration tables and parameter blocks. A Warm-Boot retains the BEPs Patch List, Parameter Blocks, etc. To issue a Warm-Boot, the BEP must be powered on and selected, with its Load-from-Uplink flag de-asserted, but its Warmboot flag in an asserted state. Upon receipt of a Halt BEP command followed by a Run BEP command, the BEP hardware will reset and invoke the loader software in its ROM. This loader copies the bulk of the instrument software from the ROM into the BEPs RAM, and transfers control to the loaded code. The loaded code detects that the instrument Warmboot flag is asserted, and installs the Patch List nodes, overwriting the code and data areas specified by the nodes with the data stored in the patch nodes. The startup software then issues a Startup Message telemetry packet.

A Warmboot of the BEP:

1. Copies the bulk of code and initialized data from ROM into I-cache and D-cache RAM.

2. Installs the patches, overwriting code and data specified by the nodes in the Patch List with the data contained in the Patch List nodes.
3. Issues a Startup Message telemetry packet, indicating the integrity of the patch list, parameter blocks and system configuration table. NOTE: Since the system configuration table controls the focal-plane temperature and FEP and DEA power settings, if the table has been corrupted, the startup code will restore the default system configuration table from ROM into RAM, overwriting the corrupted copy.
4. The initialization code for the DEA will reset the DEA Interface Controller board. This will have the effect of powering off the video boards. Within a few seconds, however, the System Configuration Task will restore power to those boards indicated in the preserved system configuration table (i.e. those boards which were on prior to the warm-boot, will be re-powered).
5. The hardware reset of the BEP will cause a reset of the Front End Processors (FEP), but won't reset the FEP power settings. FEPs which were powered prior to the reset will remain powered, but will be held in reset state until a science run is started, or until they are powered off via subsequent cold boots or "Change System Configuration" commands (see Section 3.3.2).

Commands

```
1BMODIBM (v=0) Set BootModifier Off
1WRMBTSB (v=1) Set Warmboot On
1RSETIRT (v=1) Halt BEP
1RSETIRT (v=0) Run BEP
```

Engineering Telemetry

Boot LED Sequence (for detail, refer to Engineering Telemetry in Section 3.1.3.1)

```
LED_BOOT_RESET
LED_RUN_PATCH
LED_RUN_STARTUP
LED_RUN_IDLE_A
LED_RUN_IDLE_B
LED_RUN_SCIENCE_A
LED_RUN_SCIENCE_B
```

Science Telemetry

Startup Message

```
bepStartupMessage: TTAG_STARTUP
{
   bepTickCounter   < 10 (if not, SEU or hardware error)
   version          = 11 (if not, System Patched, SEU or hardware)
   lastFatalCode    = random
   lastFatalValue   = random
   watchdogFlag     = 0 (if 1, Watchdog boot, SEU or hardware error)
   patchValidFlag   = 0 (if 1, Patch List corrupted)
   configFlag       = 0 (if 1, System Configuration corrupted)
   parametersFlag   = 0 (if 1, Parameter Blocks corrupted)
   warmBootFlag     = 1 (if 0, Cold Boot, SEU or hardware error)
}
```

Software Housekeeping

```
swHousekeeping: TTAG_SW_HOUSE
{
   startingBepTickCounter  = varying
   endingBepTickCounter    = startingBepTickCounter + 640
   statistics[] =
   {
      swStatisticId= SWSTAT_VERSION
      count        = 1
      value        = 11 (if not, System Patched, SEU or hardware)
   }
   {
      swStatisticId= SWSTAT_TIMERCB_INVOKE
      count        = 1
      value        ~= 640 (~10 timer ticks/second)
   }

NOTE: if an error is encountered restoring DEA video board power then
add:
   {
      swStatisticId = SWSTAT_DEABOARD_ERROR
      count        = variable
      value        = (deaslot << 16) | internal DEA error code
   }
}
```

Warnings

1. Warm Reset via the BEP Select command

   If an un-selected BEP is selected, its Load-from-Uplink is de-asserted, but its Warm-boot flag is asserted, the BEP will perform a warm-boot.

2. Retained Parameter Blocks

   All Patches, Parameter Blocks, and Huffman Compression Tables are retained.

3. Retained/Corrupted System Configuration

   The System Configuration Table will be retained if its checksum is intact. If corrupted, however, the System Configuration Table will be overwritten by the default contained in ROM.

4. Power-cycled DEA Video Boards

   If the System Configuration Table is intact, the DEA Video boards which were powered prior to the reset, will be power-cycled.

5. Cycled Focal Plane Temperature Control

   If the System Configuration Table is intact, the DEA focal-plane temperature control set-point will be set to 0 (due to the interface board reset), and then to the value it had prior to the reset.

6. Reset FEPs

   If the System Configuration Table is intact, FEP boards which were powered prior to the reset will remain powered, but will be held in a reset state.

7. Preserved Memory

Unused portions of BEP memory will remain intact.

8. “Stuck” LED Values

   The initial boot LED sequence will change faster than the sample rate of the telemetry system. Unless something goes wrong, don’t expect to see every code in the sequence as the instrument comes up. However, if the LED values “stick” to the following values for more than one science telemetry frame, or codes persist which are not listed, there is probably a problem with the instrument or command sequence to the instrument:

   ```
       LED_BOOT_RESET   (1,1,1,1) - BEP has just reset
       LED_RUN_PATCH    (1,0,0,1) - Copying patches or resetting patch list
   ```

   If the following LED code “sticks” for more than 64 seconds, there may also be a problem:

   ```
       LED_RUN_STARTUP  (1,0,0,0) - Starting the multi-tasking executive
   ```

3.1.3.4 Watchdog Reset

Description

When a BEP’s watchdog timer expires, the hardware sets a watchdog-reboot flag and issues a hardware reset. This may be caused by a task lockup in the BEP, or by a trapped fatal error condition, in which the recovery code uses the watchdog timer to reset the BEP. Once the BEP reboots, the loader in the BEPs ROM is invoked. If the Load-from-Uplink flag is de-asserted, the loader copies code and data from ROM into RAM, and invokes the loaded startup software. The startup software checks the state of the Warmboot flag, and if asserted, it performs a warm-reboot sequence, except that it skips the step which installs the patches. If the Warmboot flag is de-asserted, the boot-sequence is identical to that of a cold-reboot (see Section 3.1.3.2). The ground can detect the occurrence of a Watchdog reboot via the Startup Message telemetry packet, and via the bi-level “LED” telemetry items.

A Warm Watchdog Reset of the BEP:

1. Copies the bulk of code and initialized data from ROM into I-cache and D-cache RAM.
2. Skips the installation of the patches (NOTE: Although the patches aren’t applied to the code and data in RAM, the patch list will remain intact).
3. Issues a Startup Message telemetry packet, indicating that there was a watchdog reboot, and indicating the integrity of the patch list, parameter blocks and system configuration table. NOTE: Since the system configuration table controls the focal-plane temperature and FEP and DEA power settings, if the table has been corrupted, the startup code will restore the default system configuration table from ROM into RAM, overwriting the corrupted copy.
4. The initialization code for the DEA will reset the DEA Interface Controller board. This will have the effect of powering off the video boards. Within a few seconds, however, the System Configuration Task will restore power to those boards indicated in the preserved system configuration table (i.e. those boards which were on prior to the warm-boot, will be re-powered).

5. The hardware reset of the BEP will cause a reset of the of the Front End Processors (FEP), but won't reset the FEP power settings. FEPs which were powered prior to the reset will remain powered, but will be held in reset state until a science run is started, or until they are power-cycled via a pair of "Change System Configuration" commands (see Section 3.3.2).

<u>Commands</u>

None (although the crash may be as a result of an earlier command).

<u>Engineering Telemetry</u>

Boot LED Sequence

```
LED values are listed as: (1STAT3ST, 1STAT2ST, 1STAT1ST, 1STAT0ST)
LED_BOOT_RESET   (1,1,1,1) - BEP has just reset
LED_RUN_PATCH    (1,0,0,1) - Copying patches
LED_RUN_STARTUP  (1,0,0,0) - Starting the multi-tasking executive

After an initial 64 second wait, the BEP's Software Housekeeping task
will alternate the LEDs between two values every 64 seconds indicating
that no science runs are in progress, and that the BEP was restarted
due to a watchdog reset:
LED_WD_IDLE_A    (0,0,1,0)
LED_WD_IDLE_B    (0,0,1,1)

If a science run is activated, the LEDs will change to:
LED_WD_SCIENCE_A(0,0,0,0)
LED_WD_SCIENCE_B(0,0,0,1)
```

<u>Science Telemetry</u>

If the Watchdog expiration is due to a caught Fatal Error, the BEP will do a "best-effort" attempt to issue a Fatal Error Message telemetry packet, prior to forcing a watchdog timer expiration:

Fatal Message

```
fatalMessage: TTAG_FATAL
{
   bepTickCounter   = BEP tick on detection of error
   fatalCode        = enum FatalCode
   fatalValue       = depends on FatalCode
}
```

Startup Message

```
bepStartupMessage: TTAG_STARTUP
{
   bepTickCounter   < 10 (if not, SEU or hardware error)
   version          = 11 (if not, SEU or hardware error)
   lastFatalCode    = enum FatalCode (or random if uncontrolled)1
   lastFatalValue   = depends on FatalCode (or random)
   watchdogFlag     = 1 (if 0, SEU or hardware error, or not watchdog)
   patchValidFlag   = 0 (if 1, Patch List corrupted)
   configFlag       = 0 (if 1, System Configuration corrupted)
```

```
        parametersFlag   = 0 (if 1, Parameter Blocks corrupted)
        warmBootFlag     = 0/1
    }
```

Software Housekeeping

```
    swHousekeeping: TTAG_SW_HOUSE
    {
        startingBepTickCounter  = varying
        endingBepTickCounter    = startingBepTickCounter + 640
        statistics[] =
        {
            swStatisticId= SWSTAT_VERSION
            count        = 1
            value        = 11 (if not, System Patched, SEU or hardware)
        }
        {
            swStatisticId= SWSTAT_TIMERCB_INVOKE
            count        = 1
            value        ~= 640 (~10 timer ticks/second)
        }

    NOTE: if an error is encountered restoring DEA video board power then
    add:
        {
            swStatisticId = SWSTAT_DEABOARD_ERROR
            count        = variable
            value        = (deaslot << 16) | internal DEA error code
        }
    }
```

Warnings

1. Cold Watchdog Boot

   If the Warmboot flag is de-asserted when the watchdog timer expires, the system will perform a cold-boot, resetting the patch list, parameter blocks, bad pixel maps, etc. See Warnings in Section 3.1.3.2

2. Warm Watchdog Boot with No Patches

   If the Warmboot flag is asserted when the watchdog timer expires, the system will perform a warm-boot, except the patch list will not be installed. This could possibly raise compatibility issues with the retained tables, if format changes were introduced which relied on the patches being installed, or worse, a hardware problem work-around not being installed. Also, consider Warnings in Section 3.1.3.3

3. Power off in-use FEP

1. Due to a problem in the latest version of the instrument software, the lastFatalCode field in the startup message contains the BEP tick counter of the most recent Fatal Error message. The lastFatalValue field contains the fatal error code. This information is more useful from a diagnostic point of view, and is being left as-is in the instrument software.

The most common causes of Watchdog resets in the lab have been (a) bad patches, and (b) FATAL_INTR_FEP_BUS_ERROR fatal errors due to accessing a FEP when its power is off. This can happen if one powers off a FEP while it is being used in a science run.

4. Startup Message Info

   In the Startup Message of the current version of the instrument software, the lastFatalCode field contains the BEP tick counter of the most recent Fatal Error Message, and the lastFatalValue contains the corresponding fatal error message code.

5. Recovery from Watchdog Boot

   To recover from a Watchdog reset, assuming the Patch List is not the cause, issue a Halt BEP/Run BEP command sequence to cause a normal Warmboot of the BEP.

6. No looping Watchdogs

   The hardware prevents looping Watchdog resets from locking up the instrument. After an initial Watchdog reset, the hardware prevents subsequent watchdog timer expirations, including those caused by a software Fatal Error, from resetting the BEP. An intervening commanded reset (or power-on reset), is required to re-enable watchdog resets.

3.1.4 Loading from Uplink

Description

The Load-from-Uplink feature of ACIS allows a maintainer to load arbitrary software into a Back End Processor (BEP) using the software serial-digital command channel. In order to load the BEP, one must first assert the Load-from-Uplink flag, and then reset the BEP (Halt BEP/Run BEP). The hardware transfers control to the loader software in the BEP's ROM. The loader detects that the Load-from-Uplink flag is asserted, and waits for a “Start Upload” command packet followed by zero or more “Continue Upload” command packets. Once the entire load is copied from the uplinked packets into the BEPs RAM, the loader invokes the loaded software. Once running, the loaded code has control of the instrument, and is responsible for all subsequent software command processing (if any) and telemetry production (if any).

If the loader receives an unrecognized command packet, it will discard the packet, and restart the load, waiting for an initial “Start Upload” command packet.

If the loader receives a new “Start Upload” command packet in the middle of an existing load, the loader will stop the current load (leaving what was already copied into RAM intact) and start the new load. This behavior can be used to perform scatter loads into RAM (see Section 5.9).

Commands

```
        1BMODIBM (v=1) Set BootModifier On
        1RSETIRT (v=1) Halt BEP
```

```
1RSETIRT (v=0) Run BEP
```

Start Upload

```
startUpload: CMDOP_START_UPLOAD
{
   loadAddress      = user-defined- Starting address of load
   totalCount       = user-defined- Total 32-bit words in the load
   executeAddress   = user-defined- Execution address
   initialData[]    = user-defined - Array of 32-bit word data to load
}
```

Continue Upload

```
continueUpload: CMDOP_CONTINUE_UPLOAD
{
   continuationData[]= user-defined - More 32-bit data to load
}
```

Once the program has been loaded and started, it is responsible for processing subsequent software serial commands (if any).

Engineering Telemetry

Boot LED Sequence

```
LED values are listed as: (1STAT3ST, 1STAT2ST, 1STAT1ST, 1STAT0ST)
LED_BOOT_RESET         (1,1,1,1) - BEP has just reset
LED_BOOT_UPLINK_WAIT   (1,1,0,0) - Waiting for “Start Upload” packet
LED_BOOT_UPLINK_COPY   (1,0,1,1) - Waiting for “Continue Upload” pkts
LED_BOOT_UPLINK_EXECUTE(1,0,1,0) - Calling loaded program
It is up to the loaded program
```

Science Telemetry

Once the program has been loaded and started, it is responsible for all science telemetry (if any).

Warnings

1. Uses of Load-from-Uplink

   Although Load-from-Uplink is a useful diagnostic feature, building loads requires detailed knowledge of the instrument, DPA hardware, and the ACIS software development environment. Transmission of load-from-uplink commands does not require much knowledge beyond that described in this document, but act of building and understanding the effects of programs which use the Load-from-Uplink feature requires knowledge beyond the scope of this document. It is expected that only the ACIS software maintenance team will build programs which use this feature. However, it is possible that this team may request routine uplink loads to perform some types of ACIS maintenance activities.

2. State of Instrument after a Load-from-Uplink

The state of any on-board stored parameter blocks, tables, etc., is completely up to the loaded program(s). Some programs may corrupt the state of the instrument, whereas others may leave the instrument in the previous state.

3. Watchdog Timer Maintenance

   Once a program is loaded from the uplink channel and begins execution, it has up to 3 seconds to reset the BEP's watchdog timer. Once running, it is the loaded program's responsibility to maintain the timer (NOTE: If the watchdog timer expires, it will reset the BEP. It will not reset the BEP a second time until the BEP receives either a commanded reset (Halt BEP/Run BEP) or power-on reset).

3.2 Stopping the software

This section describes how the ACIS instrument software is halted.

3.2.1 BEP Power-Off

This section describes how to power-off the ACIS Digital Processor Assembly (DPA) Back End Processors (BEP).

<u>Description</u>

One method of stopping the ACIS software from running is to power-off the "active" BEP. This can be accomplished by issuing a power-off command or a disable command to the PSMC responsible for the active BEP (see Section 3.1.2 for a discussion on selecting a BEP). The recommended method is to first issue the power-off command, followed by the disable command, however, just issuing the disable command has the same affect.

<u>Commands</u>

The following are the commands issued to power-off the BEPs. See Section 3.1.1 for commands to power-on the DPA's BEPs. Also, see Section 3.3.3 for commands needed to enable and power on and off the Detector Electronics Assembly.

TABLE 8. Power Off Command Sequence

Command Mnemonic	**Command Type**	**Description**
Power Off Side-A BEP		
1DPPSAOF	Pulse to PSMC Side-A	Power-Off (but don't disable) the PSMC DPA Side-A
1DPPSADS	Pulse to PSMC Side-A	Disable the PSMC DPA Side-A
Power Off Side-B BEP		
1DPPSBOF	Pulse to PSMC Side-B	Power-Off (but don't disable) the PSMC DPA Side-B
1DPPSBDS	Pulse to PSMC Side-B	Disable the PSMC DPA Side-B

<u>Engineering Telemetry</u>

For a description of the state of the Engineering telemetry when power is off versus on, refer to Table 4 in Section 3.1.1 .

<u>Science Telemetry</u>

When the power is off on a BEP, the science telemetry is undefined, although experience in the lab indicates that it tends to float to logical '1'.

Warnings

1. Lose all loaded parameters and tables

   When a BEP is powered off, it loses all loaded parameter blocks, system configuration values, bad pixel and column maps, patches, Huffman tables and anything else that has been loaded into RAM. When the BEP is re-powered, the default parameter blocks, tables, etc. will be copied from ROM into RAM, and have their respective “as-launched” values (NOTE: This affects power-consumption and the state of the focal plane temperature controller).

2. DEA may still be running

   Since the DEA and DPA have separate power supplies, the DEA interface boards and video boards will remain powered when the DPA is powered off. Whatever state the DEA is in at the time the DPA is powered down will persist, specifically, the video board power (and clocking state), and the focal plane temperature control settings (including the focal plane bakeout heater state).

3.2.2 Halting the active BEP

Description

Another method of stopping the ACIS software is to halt the active Back End Processor. This is accomplished using a Halt BEP hardware command, or by selecting the other BEP, which throws the current BEP into a reset state (note, that if the other BEP is powered, it will run).

Commands

Halting the BEP
```
1RSETIRT (v=1)   Halt BEP
```

Selecting BEP B (causing BEP A to be held in a reset state)
```
1BSELICL (v=1)   Halt Side-A BEP and Select Side-B BEP
```

Selecting BEP A (causing BEP B to be held in a reset state)
```
1BSELICL (v=0)   Halt Side-B BEP and Select Side-A BEP
```

Engineering Telemetry

Selected BEP Bi-level
```
1STAT4ST  0 - indicates BEP A is selected,
          1 - indicates BEP B is selected
```

Reset Bi-level
```
1STAT5ST  0 - indicates selected BEP is not reset (i.e. is running)
          1 - indicates the currently selected BEP is held in reset
```

Science Telemetry

When the selected BEP is powered up and held in a reset state, it outputs a fill byte, 0xb7, to the software serial digital telemetry channel.

Warnings

1. Release of reset causes boot

   Refer to Section 3.1.3.2 , Section 3.1.3.3 , Section 3.1.4 for descriptions of the types of boots which can occur when the BEP's reset is released.

3.3 Changing the System Configuration

This section describes how to modify the ACIS software system configuration table, including controlling the power to the Front End Processors and the Detector Electronics Assembly Video Boards.

3.3.1 System Configuration Table

The System Configuration Table consists of a table of various settings. These settings are broken into the following components:

- FEP and DEA Power
- DEA Interface Board Settings
- DEA Video Board Settings

The values in the table are modified using a “Change System Configuration” command packet, whose contents identify a set of items and their values to load into the table.

The instrument’s System Configuration Task polls this table once per second to see if anything has changed in the FEP/DEA Power, or DEA Interface Board settings since the last time the table was polled. If anything has changed, the task updates the corresponding power item, or item in the DEA interface board.

The DEA Video board settings, on the other hand, are only used at the start of a Science Run. At that time, they are used to configure the various Digital-to-Analog settings in the video board.

3.3.2 Controlling FEP Power

<u>Description</u>

The Front End Processors (FEPs) are split into two groups, FEPs 0, 1, 2 are powered by the Side-A DPA Power Supply, and FEPs 3, 4 and 5 are powered by the Side-B DPA Power Supply (refer to Section 3.1.1 for a description of powering the DPA Power supplies). Prior to enabling the power to a specific FEP, the appropriate supply must be on. Once the FEPs supply is on, its power must be enabled by the BEP using an entry in the Change System Configuration software command.

The FEP Power entry of the Change System Configuration command consists of a bit-map, where bit 0 (lsb) corresponds to FEP 0, bit 1 corresponds to FEP 1, etc. If the bit in the map is 0, then the corresponding FEP is powered off. If the bit is a 1, then the FEP is powered on.

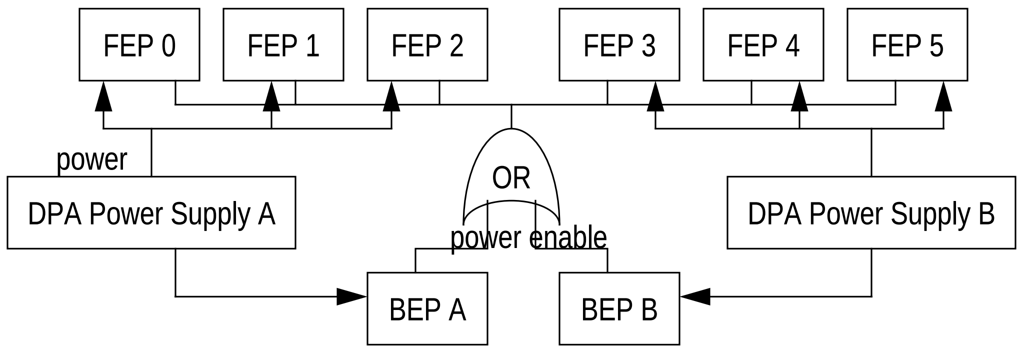
In order to avoid exceeding possible power limitations (there are currently no such limitations, but things change), the System Configuration task within ACIS always first powers off those FEPs which are to be off, and then powers on the FEPs which should be running. There is a 1 second delay between each power off command. There will never be less than a 1 second delay between each power on. However, it currently takes 7 to 10 seconds to load the current FEP software into each FEP, so it can take up to 1 minute to power on all 6 FEPs. If the FEPs are already in their desired states (i.e. off or on), the System Configuration task takes no action.

The Side-A and Side-B BEPs can both enable and disable power to the FEPs. In order to prevent a failure on one BEP from preventing the other being able to power on a FEP, the power enable outputs of the BEPs are logically OR'd. This can lead to confusion when selecting between BEPs. If both BEPs are powered (which they would have to be if you were using all 6 FEPs), and the previously selected BEP has power enabled to some of the FEPs, the currently selected BEP will not be able to power-off those FEPs. To avoid this condition, always issue a command to power off FEPs on the active BEP prior to selecting the other BEP.

An attempt to enable power to a FEP which does not have its corresponding power-supply on will generate a memory bus error exception on the BEP, which is trapped and handled by the BEP ONLY when the FEP is commanded on. In this situation, the System Configuration task will repeat its attempts to power on the FEP once a second. However, if a FEP is already on, and its power-supply is turned off, the next access to the FEP will cause a non-recovered bus error exception, resulting in a Fatal Error telemetry message, and watchdog reset of the BEP. Also note that, even when handled by the BEP, these bus errors will disrupt the transmission of a telemetry packet being sent to the RCTU, causing fill-pattern gaps in middle of the telemetry packets.

An un-handled memory bus exception (and resulting Fatal Error and watchdog reset) will also be caused if a FEP's power is disabled in the middle of a BEP to FEP memory access. In order to avoid this, disable FEP power only when there are no science runs active, and no commanded FEP memory loads or dumps in progress.

FIGURE 2. FEP Power Supplies and Enables



Commands

Change System Configuration

```
changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
{
   entries[] =
   {
      itemId    =  SYSSET_FEP_POWER
      itemValue =  FEP power bit-map, where bits 0 - 5 correspond to
                   FEPs 0 - 5, respectively, and bits 7 - 15 are
                   unused.
   }
}
```

For example, to power-on all 6 FEPs:

```
changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
{
   entries[] =
   {
      itemId    =  SYSSET_FEP_POWER
      itemValue =  0x3f
   }
}
```

To power on FEP 0 only (and power off FEPs 1 .. 5, if they were on):

```
changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
{
   entries[] =
   {
      itemId    =  SYSSET_FEP_POWER
      itemValue =  0x1
   }
}
```

Engineering Telemetry

The only engineering telemetry items which indicate whether or not FEPs are up and running are the electric current channels. These can give an indication as to how many FEPs are on and in what state they are in, but do not give very specific information beyond that. Refer to Appendix A for a table of power levels under different instrument configurations.

Science Telemetry

Command Echo

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

      changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
      {
```

```
                entries[] =
                {
                   itemId    =  SYSSET_FEP_POWER
                   itemValue =  FEP power bit-map
                }
             }
      }
```

Software Housekeeping Statistics (may be spread across two consecutive reports)

```
      swHousekeeping: TTAG_SW_HOUSE
      {
         startingBepTickCounter  = varying
         endingBepTickCounter    = startingBepTickCounter + 640
         statistics[] =
         {
            swStatisticId = SWSTAT_VERSION
            count         = 1
            value         = 11 (if not, System Patched, SEU or hardware)
         }
         {
            swStatisticId = SWSTAT_TIMERCB_INVOKE
            count         = 1
            value         ~= 640 (~10 timer ticks/second)
         }
         {
            swStatisticId =  SWSTAT_FEP_EXECMEM
            count         =  count of FEPs being powered on
            value         =  FEP Id of last FEP powered on
         }
         {
            swStatisticId =  SWSTAT_FEP_WRITEMEM
            count         =  count of FEPs being powered on
            value         =  FEP Id of last FEP powered on
         }
         {
            swStatisticId =  SWSTAT_FEPMAN_POWERON
            count         =  count of FEPs being powered on
            value         =  FEP Id of last FEP powered on
         }
         {
            swStatisticId =  SWSTAT_FEPMAN_POWEROFF
            count         =  count of FEPs being powered off
            value         =  FEP Id of last FEP powered off
         }
         {
            swStatisticId =  SWSTAT_FEPMAN_STARTLOAD
            count         =  count of FEPs being powered on
            value         =  FEP Id of last FEP powered on
         }
         {
            swStatisticId =  SWSTAT_FEPMAN_ENDLOAD
            count         =  count of FEPs being powered on
            value         =  FEP Id of last FEP powered on
         }
      }
```

If there were errors during the execution of the command, or an activity was using one of the FEPs which was being powered off, the Software Housekeeping statistics may also include entries:

```
SWSTAT_FEPLOCK_TIMEOUT     -  FEP Wait: Wait for lock timed out
SWSTAT_FEPLOCK_POWEROFF    -  FEP Wait: FEP has no power
SWSTAT_FEPLOCK_RESET       -  FEP Wait: FEP is reset
SWSTAT_FEPLOCK_NOIO        -  FEP Wait: FEP has no mailbox/ringbuffer

SWSTAT_FEPREPLY_TIMEOUT    -  FEP Reply: FEP timed out
SWSTAT_FEPREPLY_POWEROFF   -  FEP Reply: FEP has no power
SWSTAT_FEPREPLY_RESET      -  FEP Reply: FEP is reset
SWSTAT_FEPREPLY_NOIO       -  FEP Reply: FEP has no mailbox/ringbuffer

SWSTAT_FEPCMD_MBOXSTATE    -  FEP Mailbox not empty (protocol error)
```

If an attempt is made to enable FEP power, but its corresponding power supply is off, the following Software Housekeeping entry will be reported:

```
SWSTAT_INTR_FEPBUS     - FEP access causes bus error exception on BEP
```

If power is removed while the FEP is being used during a run or memory load, a Fatal Error and Watchdog Reset may result:

Fatal Error

```
fatalMessage: TTAG_FATAL
{
   bepTickCounter   =  BEP 10Hz Tick at time of report
   fatalCode        =  FATAL_INTR_FEP_BUS_ERROR
   fatalValue       =  Contents of R3000 Bad Virtual Address Register
}
```

<u>Warnings</u>

1. Power Levels

   Each FEP draws a certain amount of current. The current draw is different when the FEP is held in a reset state, versus when it is fully up and running. Refer to Appendix A for a table of power levels, given certain configurations.

2. Fatal Errors

   If a FEP is powered off while its memory is being accessed, it will generate an un-handled BEP memory bus exception, resulting in a Fatal Error message and BEP Watch-dog reset.

3. Power-Retries/Telemetry Disruption

   If a FEP's power-supply is off when a command is issued to enable the FEP's power, the hardware will generate BEP memory bus exceptions, which will interfere with telemetry packet transmission (fill-bytes, 0xb7, in the middle of telemetry packets). The System Configuration task will retry the power enable once per second, which may continue to interfere with telemetry.

4. FEP Timestamp after Power-On

   The FEP science 100KHz timestamp counters are synchronized to the BEPs 100KHz timestamp counter. They are reset to 0 whenever the BEPs counter's least significant 25 bits are 0. This occurs about once every 7 minutes. After a FEP is powered on, its counter is initially out-of-synch with the BEPs counter, and sometime within 7 minutes of being powered on, the FEP's counter will be set to 0. From then on, the reset of the FEP counter is synchronous with its rolling-over. This can lead to confusing initial FEP timestamps if a science run starts taking data within 7 minutes of a FEP being powered on.

3.3.3 Controlling the DEA Interface and Video Power

Description

The Detector Electronics Assembly (DEA) consists of two interface boards, and 10 video boards, one for each CCD. Each interface board is powered separately, one from the DEA Side-A Power Supply, and the other from the DEA Side-B Power Supply. Only one interface board may be powered at a time. The video boards are powered in pairs via a set of latching relays on the primary interface board, and enabled via the active interface board. These relays operate from the current active power supply, and, whenever the supplies are switched, must be commanded to switch to the active supply.

Like the DPA Power Supplies, the DEA Power Supplies are commanded using hardware pulse commands, consisting of enable, on, off and disable. The video board relays and power enables, however, are set using the BEP's System Configuration Table, which is modified with a "Change System Configuration" command packet.

The following table indicates the relationships between the CCDs, relays and video boards:

TABLE 9. CCD, Video Board and Power Relay Relationships

CCD	CcdId Value	Video Board	Relay Set
I0	CCD_I0 = 0	1	0
I1	CCD_I1 = 1	3	1
I2	CCD_I2 = 2	5	2
I3	CCD_I3 = 3	7	3
S0	CCD_S0 = 4	2	0
S1	CCD_S1 = 5	4	1
S2	CCD_S2 = 6	8	3
S3	CCD_S3 = 7	6	2
S4	CCD_S4 = 8	9	4
S5	CCD_S5 = 9	10	4

Because the ACIS software internally translates references to CCDs into DEA video board ids, all command and telemetry references identify video boards using the corresponding CCD identifier. The DEA Video Power entry of the Change System Configuration command consists of a bit-map, where bit 0 (lsb) corresponds to CCD I0, bit 1 corresponds to CCD I1, etc. If the bit in the map is 0, then the corresponding video board is powered off. If the bit is a 1, then the video board is powered on. Once the table entry is modified, the System Configuration Task first powers off all of the video boards which were indicated to be off, and then powers on the boards which were requested to be powered on. To avoid stressing the power supply, the System Configuration Task ensures at least a 1 second delay between each power command. Assuming that the power settings to all 10 boards are being modified, the entire power switch should take between 10 and 20 seconds.

The relays are set to point to the current power supply using the "Relay Set" entries of the Change System Configuration command. If an entry is set to a non-zero value, a command is issued to the DEA interface board to switch the corresponding relay, and the associated pair of video boards, to the currently active power supply. If an entry is set to 0, no action is taken. Since the relays are latching, they remain switched across power-cycles. The only way to "unswitch" a relay is to switch from one DEA Power Supply to the other.

Commands

To power on Side-A DEA Power Supply (and the primary DEA interface board) (NOTE: The Side-B DEA Power Supply must be off prior to issuing these commands):

```
1DEPSAEN  - High-level Pulse  - Enable DEA Side-A Power Supply
1DEPSAON  - High-level Pulse  - Power On DEA Side-A Power Supply
```

To power off the Side-A DEA Power Supply:

```
1DEPSAOF  - High-level Pulse  - Power Off DEA Side-A Power Supply
1DEPSADS  - High-level Pulse  - Disable DEA Side-A Power Supply
```

To power on the Side-B DEA Power Supply (and the redundant DEA interface board) (NOTE: The Side-A DEA Power Supply must be off prior to issuing these commands):

```
1DEPSBEN  - High-level Pulse  - Enable DEA Side-B Power Supply
1DEPSBON  - High-level Pulse  - Power On DEA Side-B Power Supply
```

To power off the Side-B DEA Power Supply:

```
1DEPSBOF  - High-level Pulse  - Power Off DEA Side-A Power Supply
1DEPSBDS  - High-level Pulse  - Disable DEA Side-A Power Supply
```

To switch all of the video relays to the currently active interface board:

```
changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
{
   entries[] =
   {
      itemId    =  SYSSET_CNTL_RELAY_SET_0 -   CCD I0/S0
      itemValue =  1                       -   Issue switch command
   }
   {
      itemId    =  SYSSET_CNTL_RELAY_SET_1 -   CCD I1/S1
      itemValue =  1                       -   Issue switch command
   }
   {
```

```
            itemId    =  SYSSET_CNTL_RELAY_SET_2 -   CCD I2/S3
            itemValue =  1                       -   Issue switch command
         }
         {
            itemId    =  SYSSET_CNTL_RELAY_SET_3 -   CCD I3/S2
            itemValue =  1                       -   Issue switch command
         }
         {
            itemId    =  SYSSET_CNTL_RELAY_SET_4 -   CCD S4/S5
            itemValue =  1                       -   Issue switch command
         }
      }
```

Note that if “itemValue” is 0, then the entry has no effect on the relay setting.

To power on and off the DEA Video Boards (NOTE: Either A or B side DEA Power must be on, but not both, and the relay corresponding to the desired CCDs must be switched to the active supply):

```
      changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
      {
         entries[] =
         {
            itemId    =  SYSSET_DEA_POWER
            itemValue =  DEA power bit-map, where bits 0 - 9 correspond to
                         CCDs I0 - S5, respectively, and bits 10 - 15 are
                         unused. See Table 9.
         }
      }
```

Engineering Telemetry

DEA Power Supply Verifiers

```
         1DEPSAEN  Verifies DEA Power Supply Side-A enabled state
                   0 - disabled
                   1 - enabled
         1DEPSAON  Verifies DEA Power Supply Side-A powered state
                   0 - not powered
                   1 - powered
         1DEPSBEN  Verifies DEA Power Supply Side-B enabled state
                   0 - disabled
                   1 - enabled
         1DEPSBON  Verifies DEA Power Supply Side-B powered state
                   0 - disabled
                   1 - enabled
```

The only engineering telemetry items which indicates whether or not DEA Video boards (and by induction, their relay states) are up and running are the electric current channels. These can give an indication as to how many video boards are on and in what state they are in, but do not give very specific information beyond that. Refer to Appendix A for a table of power levels under different instrument configurations.

Science Telemetry

Command Echo from Relay Command

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
   {
      entries[] =
      {
         itemId    =  SYSSET_CNTL_RELAY_SET_0 -  CCD I0/S0
         itemValue =  1                       -  Issue switch command
      }
      {
         itemId    =  SYSSET_CNTL_RELAY_SET_1 -  CCD I1/S1
         itemValue =  1                       -  Issue switch command
      }
      {
         itemId    =  SYSSET_CNTL_RELAY_SET_2 -  CCD I2/S3
         itemValue =  1                       -  Issue switch command
      }
      {
         itemId    =  SYSSET_CNTL_RELAY_SET_3 -  CCD I3/S2
         itemValue =  1                       -  Issue switch command
      }
      {
         itemId    =  SYSSET_CNTL_RELAY_SET_4 -  CCD S4/S5
         itemValue =  1                       -  Issue switch command
      }
   }
}
```

Command Echo from DEA Video Board Power Enable

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
   {
      entries[] =
      {
         itemId    =  SYSSET_DEA_POWER
         itemValue =  DEA Video Board power bit-map
      }
   }
}
```

Software Housekeeping Statistics

```
swHousekeeping: TTAG_SW_HOUSE
{
   startingBepTickCounter  = varying
   endingBepTickCounter    = startingBepTickCounter + 640
   statistics[] =
   {
      swStatisticId = SWSTAT_VERSION
```

```
            count        = 1
            value        = 11 (if not, System Patched, SEU or hardware)
         }
         {
            swStatisticId= SWSTAT_TIMERCB_INVOKE
            count        = 1
            value        ~= 640 (~10 timer ticks/second)
         }
         {
            swStatisticId=  SWSTAT_DEACCD_POWEROFF
            count        =  count of Video boards being powered off
            value        =  CCD Id associated with last Video turned off
         }
         {
            swStatisticId=  SWSTAT_DEACCD_POWERON
            count        =  count of Video boards being powered on
            value        =  CCD Id associated with last Video turned on
         }
      }
```

If an error is encountered while commanding the DEA, either to switch the relays, or control power to the video boards, the following housekeeping entry will be supplied:

```
      SWSTAT_DEABOARD_ERROR  - Error issuing command to DEA interface
      where the 16 least significant bits of reported “value” field contain
      an internal software error code, and the most significant 16 bits of
      the “value” field contain the video board slot index (as opposed to the
      CCD Id), or “11” if the command was directed to the interface board.
```

If DEA Housekeeping is running (see Section 3.4.6) and is querying the state of the relays, it will be indicated as follows:

```
      deaHousekeepingData: TTAG_DEA_HOUSE
      {
         deaBlockId      = Id of parameter block used for the DEA Run
         commandId       = Id of the command which started the DEA Run
         bepTickCounter  = BEP Interrupt Count (10Hz tick) at start of DEA
                           housekeeping acquisition
         entries[] =
         {
            query =
            {
               ccdId     = CCD_DESELECT (selects the interface board)
               queryId   = DEAHOUSE_CNTL_RELAY (reads state of relays)
            }
            value        = state of relays (where bit 0 corresponds to
                           relay set 0, bit 1 to relay set 1, etc. If a bit
                           is 0, the relay set is not switched to the active
                           supply. If a bit is 1, then the relay set is
                           switched, and the corresponding video boards can
                           be powered).
         }
      }
```

While the video boards are not powered (or if errors occur), any active DEA Housekeeping queries to the video boards will result in a “value” field of 0xffff.

If errors occur in queries to the interface board, the “value” field will contain 0xffff (for example, if there’s no DEA power supply turned on).

Warnings

1. Don’t turn on both DEA Power Supplies

   Although there are protections against damage to the system, the DEA is not designed to operate with both DEA Power Supplies powered.

2. The relays “remember”

   The DEA video relays stay switched to the most recent side across power-cycles. For example, if one powers DEA Side-B, switches the relays over, and then powers off the system, if one later powers DEA Side-A, all of the relays will remain switched to Side-B until a set of Change System Configuration entry commands are issued to switch them to the now active Side-A.

3. Video Board Power cycles may have to be routine

   The Analog-to-Digital converters on the DEA video boards may be subject to radiation-induced latch-up. The current limiter circuits on the video boards protect the boards from any damage, but the ADCs will not recover without intervention. To recover from a latch-up, the user must issue a Change System Configuration command to power off the video boards, wait, and then issue another to re-power the video boards (NOTE: Alternatively, a warm-boot of the BEP could be used). Currently, it is not expected to be a problem. If latch-ups become a problem in-flight, the ground may want to include power-cycles of the video boards as part of the science run start-up and shutdown procedure.

3.3.4 Controlling the DEA Video Board Settings

Description

The DEA Video boards contain a set of option settings and Digital-to-Analog Converter (DAC) voltage settings, for use when clocking the CCDs. All of these settings, with the exception of four Analog-to-Digital Converter (ADC) video offset values, are maintained using the System Configuration table, and are modified using “Change System Configuration” command packets. Since the values used for the video offsets may have to be varied to accommodate changes in Science Run parameters, the four ADC video offset parameters for each configured video board are supplied with the parameter blocks for a given science run (see Section 4.1). The setting values are loaded into their respective video boards during the setup stage of a science run (i.e. after a “Start Science” command is received). Refer to the MIT 36-02205 DPA/DEA Interface Control Document for detailed descriptions of the DEA register options and conversion formulae.

Each item has a limit value (see Table 10). If an entry within a “Change System Configuration” command attempts to set a value beyond its limit, the command echo will report the condition and the offending item will be set to its maximum allowed value. The limits values are listed in Section 3.3.5 and can only be changed via Write Memory or Patch commands.

NOTE: There is currently a reasonable set of defaults for all values, which was used during the High-Resolution Mirror X-ray Calibration and Flat-Field X-ray Calibration activities at Marshall Space Flight Center. It is expected that the DEA setting values will rarely be changed.

Commands

Change System Configuration Video Board Register Options:

```
X := (CCD Id * (SYSSET_CCD_END - SYSSET_CCD_BASE + 1);

changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
{
   entries[] =
   {
      itemId    = SYSSET_CCD_SEQ_OFFSET+X  - Video Sequencer Offset
      itemValue = 0..63                    - Sequencer Cycle Number
   },
   {
      itemId    = SYSSET_CCD_ADC_OFFSET+X  - Video ADC Offset
      itemValue = 0..63                    - Sequencer Cycle Number
   },
   {
      itemId    = SYSSET_CCD_VIDEO_ENABLE+X   - Video Channel Mask
      itemValue = selection bitmap            -
        Bit 0 corresponds to channel A, bit 1 to channel B, etc. If 0,
        output from the channel is enabled, if 1, then the output from
        the channel is driven to 0. This is intended as a diagnostic
        tool only.
   },
   {
      itemId    = SYSSET_CCD_BJD+X         - Back-Junction Diode
      itemValue = 0 or 1                   - O = Off, 1 = On
   }
}
```

Change System Configuration Video Board DAC Levels:

```
X := (CCD Id * (SYSSET_CCD_END - SYSSET_CCD_BASE + 1);

SYSSET_DAC_PIA_P+X  - Image Array Parallel High Level
SYSSET_DAC_PIA_MP+X - Image Array Parallel Low Level Positive
SYSSET_DAC_PIA_M+X  - Image Array Parallel Low Level Negative
(actual low-level is SYSSET_DAC_PIA_MP - SYSSET_DAC_PIA_M)

SYSSET_DAC_PFS_P+X  - Framestore Parallel High Level
SYSSET_DAC_PFS_MP+X - Framestore Parallel Low Level Positive
SYSSET_DAC_PFS_M+X  - Framestore Parallel Low Level Negative
(actual low-level is SYSSET_DAC_PFS_MP - SYSSET_DAC_PFS_M)

SYSSET_DAC_S_P+X    - Serial Register High Level
SYSSET_DAC_S_M+X    - Serial Register Low Level

SYSSET_DAC_R_P+X    - Reset Gate High Level
```

```
SYSSET_DAC_R_MP+X   - Reset Gate Low Level Positive
SYSSET_DAC_R_M+X    - Reset Gate Low Level Negative
(actual low-level is SYSSET_DAC_R_MP - SYSSET_DAC_R_M)

SYSSET_DAC_SCP+X    - Scupper

SYSSET_DAC_OG_P+X   - Output Gate High Level
SYSSET_DAC_OG_M+X   - Output Gate Low Level

SYSSET_DAC_RD+X     - Reset Diode

SYSSET_DAC_DR0+X    - Drain Output for Channel A
SYSSET_DAC_DR1+X    - Drain Output for Channel B
SYSSET_DAC_DR2+X    - Drain Output for Channel C
SYSSET_DAC_DR3+X    - Drain Output for Channel D
```

Engineering Telemetry

None

Science Telemetry

Command Echo if all settings within limits

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
   {
      entries[] =
      {
         itemId    = SYSSET_CCD/DAC_+X - Selected item
         itemValue = as sent           - Commanded value
      },
      ...
   }
}
```

Command Echo if one or more settings beyond limits

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_ITEM_CLIPPED

   changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
   {
      entries[] =
      {
         itemId    = SYSSET_CCD/DAC_+X - Selected item
         itemValue = as sent           - Commanded value
      },
      ...
   }
}
```

Software Housekeeping Statistics

```
swHousekeeping: TTAG_SW_HOUSE
{
   startingBepTickCounter= random
   endingBepTickCounter   = startingBepTickCounter + 640
   statistics[] =
   {
      swStatisticId= SWSTAT_VERSION
      count        = 1
      value        = 11 (if not, System Patched, SEU or hardware)
   }
   {
      swStatisticId= SWSTAT_TIMERCB_INVOKE
      count        = 1
      value        ~= 640 (~10 timer ticks/second)
   }
}
```

If an entry within the command exceeded its limit value, the following software housekeeping entry will be supplied:

```
SWSTAT_SYSCFG_IN_CLIP  - Configured setting beyond limit, where the
reported “value” field is the last offending entry “itemId” field in
the command packet.
```

NOTE: Since the CCD Video Board parameters are loaded into the DEA when a science run is started, a SWSTAT_DEABOARD_ERROR won’t be reported as an immediate result of the Change System Configuration command.

These parameters, once loaded into the DEA at the start of a science run, will affect the following DEA Housekeeping values:

```
DEAHOUSE_CCD_REG_0  - Register 0 Sequencer Control Contents
DEAHOUSE_CCD_REG_1  - Register 1 Video ADC Control Contents
DEAHOUSE_CCD_REG_2  - Register 2 Test Aid Contents
DEAHOUSE_CCD_REG_3  - Register 3 Miscellaneous Contents

DEAHOUSE_CCD_PIA_P  - Image Array Parallel Voltage + Level
DEAHOUSE_CCD_PIA_M  - Image Array Parallel Voltage - Level
DEAHOUSE_CCD_PFS_P  - Framestore Parallel Voltage + Level
DEAHOUSE_CCD_PFS_M  - Framestore Parallel Voltage - Level
DEAHOUSE_CCD_S_P    - Serial Register Voltage + Level
DEAHOUSE_CCD_S_M    - Serial Register Voltage - Level
DEAHOUSE_CCD_R_P    - Reset Gate Voltage + Level
DEAHOUSE_CCD_R_M    - Reset Gate Voltage - Level
DEAHOUSE_CCD_OG     - Output Gate Bias Level
DEAHOUSE_CCD_SCP    - Scupper Voltage Level
DEAHOUSE_CCD_RD     - Reset Diode Voltage
DEAHOUSE_CCD_DR0    - Drain Output Channel A Voltage
DEAHOUSE_CCD_DR1    - Drain Output Channel B Voltage
DEAHOUSE_CCD_DR2    - Drain Output Channel C Voltage
DEAHOUSE_CCD_DR3    - Drain Output Channel D Voltage
```

Warnings

1. Loaded into DEA upon Science Run

Although the settings are loaded into the instrument with a Change System Configuration command, the Video Board settings aren't actually loaded into the DEA hardware until a Start Science Run or Start Bias Only Run command is received, and the BEP starts setting up for the run.

2. Jitter DAC

   During X-ray Calibration testing, it was discovered that it is necessary to "jitter" some of the video board voltages prior to starting to take science data. This action causes the CCDs to flush out built-up residual charge more effectively, leading to shorter start-up times. The most recent version of the instrument software now contains code which, when setting up for a science run, sets the necessary video board DACs to their "jitter" value, parallel clocks the CCDs for up to 11 seconds, and then restores the DACs to the values configured in the System Configuration Table. For more detail of Jitter DAC, see TBD.

3. Video ADC Offsets in Science Parameter Block

   Although the Video Board ADC Offset parameters are listed in the System Configuration Table, they are overridden by the corresponding parameters in a science run's parameter block, and as such, the values stored in the System Configuration Table never make it into the DEA.

4. Clipped to upper limits

   Items whose values exceed their respective limits are clipped to the limit value.

5. See Warnings in Section 3.4.6.5

3.3.5 CCD Voltage Setting Limits

Description

All items in the System Configuration Table have upper limits (no lower limits were required). If a "Change System Configuration" command is received which contains one or more entries whose value exceeds its limit, the system stores the maximum values for these items in their corresponding locations, and reports the occurrence in the "result" field of the "Command Echo" telemetry packet, as well as in an entry of the next "Software Housekeeping" telemetry packet.

Table Values

The following contains the limit values for the System Configuration Table entries:

TABLE 10. System Configuration Table Value Limits

Change System Configuration: itemId	**Limit Value**	**Effect**
SYSSET_DEA_POWER	0xffff	no limit
SYSSET_FEP_POWER	0xffff	no limit
SYSSET_CNTL_MASTER_CLK	0xffff	no limit
SYSSET_CNTL_FOCAL_TEMP	0xffff	no limit
SYSSET_CNTL_BAKE_TEMP	0xffff	no limit
SYSSET_CNTL_BAKE_ENABLE	0	cannot enable bakeout heater without patch or write-memory command
SYSSET_CNTL_LED_ENABLE	0xffff	no limit
SYSSET_CNTL_HOUSE_HOLD	0xffff	no limit
SYSSET_CNTL_SIGNAL_PATH	0xffff	no limit
SYSSET_CNTL_CMDCLOCK_DISABLE	0xffff	no limit
SYSSET_CNTL_CMDDATA_DISABLE	0xffff	no limit
SYSSET_CNTL_RELAY_SET_0	0xffff	no limit
SYSSET_CNTL_RELAY_SET_1	0xffff	no limit
SYSSET_CNTL_RELAY_SET_2	0xffff	no limit
SYSSET_CNTL_RELAY_SET_3	0xffff	no limit
SYSSET_CNTL_RELAY_SET_4	0xffff	no limit
SYSSET_CCD_SEQ_OFFSET	0xffff	no limit
SYSSET_CCD_ADC_OFFSET	0xffff	no limit
SYSSET_CCD_VIDEO_ENABLE	0xffff	no limit
SYSSET_CCD_HOLD_HOUSE	0xffff	no limit
SYSSET_CCD_BJD	0xffff	no limit
SYSSET_CCD_HIGH_SPEED_TAP	0xffff	no limit
SYSSET_DAC_PIA_P	255	limited to 12.775V
SYSSET_DAC_PIA_MP	255	limited to 12.775V
SYSSET_DAC_PIA_M	140	limited to -7.025V
SYSSET_DAC_PFS_P	255	limited to 12.775V
SYSSET_DAC_PFS_MP	255	limited to 12.775V
SYSSET_DAC_PFS_M	140	limited to -7.025V
SYSSET_DAC_S_P	255	limited to 12.775V
SYSSET_DAC_S_M	140	limited to -7.025V
SYSSET_DAC_R_P	255	limited to 12.775V
SYSSET_DAC_R_MP	255	limited to 12.775V
SYSSET_DAC_R_M	140	limited to -7.025V
SYSSET_DAC_SCP	255	limited to 12.775V
SYSSET_DAC_OG_P	255	limited to 12.775V
SYSSET_DAC_OG_M	140	limited to -7.025V

TABLE 10. System Configuration Table Value Limits

Change System Configuration: itemId	Limit Value	Effect
SYSSET_DAC_RD	233	limited to 11.7V
SYSSET_DAC_DR0	177	limited to 20.6V
SYSSET_DAC_DR1	177	limited to 20.6V
SYSSET_DAC_DR2	177	limited to 20.6V
SYSSET_DAC_DR3	177	limited to 20.6V
SYSSET_DAC_A_OFF	0xffff	no limit
SYSSET_DAC_B_OFF	0xffff	no limit
SYSSET_DAC_C_OFF	0xffff	no limit
SYSSET_DAC_D_OFF	0xffff	no limit

Telemetry

If an entry in a "Change System Configuration Table" command exceeds its limit value, the result field of the corresponding Command Echo telemetry packet will be set to "CMDRESULT_ITEM_CLIPPED", and the next Software Housekeeping telemetry packet will contain a "SWSTAT_SYSCFG_IN_CLIP" entry, whose "count" field indicates the number of clipped entries, and whose "value" field indicates the last "SYSSTAT_" item which was clipped.

3.3.6 Controlling the Focal Plane Thermal Controller

Description

The ACIS Focal Plane thermal controller uses coarse and fine temperature set-points to control the focal plane temperature. These set-points are maintained in the System Configuration Table, and are modified using entries within a "Change System Configuration" command. Within about 1 second of being modified, the System Configuration task relays the change into the DEA interface control board.

In addition to the normal 4 watt (maximum) heater provided to control the focal plane temperature to the commanded set-point, a bake-out 40 watt (maximum) heater may be switched on to operate in parallel with the normal heater. The bake-out enable command provides this switching function. The circuitry controlling the focal plane temperature is entirely separate from the choice of maximum heater power available.

In order to heat the focal plane up enough to bake off contaminants, the bakeout heater must be enabled. The desired state of this heater is also maintained in the System Configuration Table. Currently, the limit for this setting is set to 0, preventing a "Change System Configuration" command entry from enabling the heater. If the system is patched or poked (write-memory) to change the setting limit, the heater can be enabled through a subsequent "Change System Configuration" command. Once enabled, when the desired focal plane temperature is greater than the actual temperature, the heater will be turned on.

Refer to TBD for the conversion formulae from set-point values to resulting temperatures.

Commands

Change System Configuration to modify temperature set-points:

```
changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
{
   entries[] =
   {
      itemId    = SYSSET_CNTL_BAKE_TEMP    - Coarse Temp. Set-Point
      itemValue = temperature setting      - Desired coarse temp.
   },
   {
      itemId    = SYSSET_CNTL_FOCAL_TEMP   - Fine Temp. Set-Point
      itemValue = temperature setting      - Desired fine temperature
   }
}
NOTE: For historical reasons, SYSSET_CNTL_BAKE_TEMP is badly named. It
is not the “bake-out” temperature, but rather the coarse temperature
set-point.
```

Change System Configuration to disable the bake-out heater

```
changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
{
   entries[] =
   {
      itemId    = SYSSET_CNTL_BAKE_ENABLE  - Bakeout Enable
      itemValue = 0                        - Disable heater
   },
}
```

Change System Configuration to enable bake-out heater

```
changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
{
   entries[] =
   {
      itemId    = SYSSET_CNTL_BAKE_ENABLE  - Bakeout Enable
      itemValue = 1                        - Enable heater
   },
}
NOTE: Unless the limit table is modified, this command will result in a
CMDRESULT_ITEM_CLIPPED, and the heater will not be enabled.
```

Engineering Telemetry

Temperature Measurements TBD

Commands to change the temperature set-points and enable or disable the Focal Plane bakeout heater will affect the overall power consumption of the system. Refer to Appendix A for a list of the various power configurations.

Science Telemetry

Command Echo for temperature set-points

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK

   changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
   {
      entries[] =
      {
         itemId    = SYSSET_CNTL_BAKE_TEMP
         itemValue = temperature setting
      },
      {
         itemId    = SYSSET_CNTL_FOCAL_TEMP
         itemValue = temperature setting
      }
   }
}
```

Command Echo for Bakeout Heater Disable

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK

   changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
   {
      entries[] =
      {
         itemId    = SYSSET_CNTL_BAKE_ENABLE
         itemValue = 0
      }
}
```

Command Echo for Bakeout Heater Enable if limit table is as-launched

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_ITEM_CLIPPED

   changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
   {
      entries[] =
      {
         itemId    = SYSSET_CNTL_BAKE_ENABLE
         itemValue = 1
      }
}
```

Command Echo for Bakeout Heater Enable if limit table modified to allow Bake-out Enables

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
```

```
      result    = CMDRESULT_OK

      changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
      {
         entries[] =
         {
            itemId    = SYSSET_CNTL_BAKE_ENABLE
            itemValue = 1
         }
   }
```

Software Housekeeping

```
   swHousekeeping: TTAG_SW_HOUSE
   {
      startingBepTickCounter  = varying
      endingBepTickCounter    = startingBepTickCounter + 640
      statistics[] =
      {
         swStatisticId = SWSTAT_VERSION
         count         = 1
         value         = 11 (if not, System Patched, SEU or hardware)
      }
      {
         swStatisticId = SWSTAT_TIMERCB_INVOKE
         count         = 1
         value         ~= 640 (~10 timer ticks/second)
      }
   }
```

If a Bakeout Enable entry is present, and the limit table had not been modified to allow it, the following software housekeeping entry will be supplied:

```
   SWSTAT_SYSCFG_IN_CLIP  - Configured setting beyond limit, where the
   reported “value” field is “SYSSET_CNTL_BAKE_ENABLE”.
```

If DEA Housekeeping is running (see Section 3.4.6) and is monitoring the Focal Plane temperature, the set-point changes and Bakeout Enable will affect the resulting temperature of the focal plane:

```
   deaHousekeepingData: TTAG_DEA_HOUSE
   {
      deaBlockId      = Id of parameter block used for the DEA Run
      commandId       = Id of the command which started the DEA Run
      bepTickCounter  = BEP 10Hz Tick at start of DEA housekeeping
                        acquisition
      entries[] =
      {
         query =
         {
            ccdId     = CCD_DESELECT (selects the interface board)
            queryId   = DEAHOUSE_CNTL_ADC_FPTEMP_12
         }
         value        = Focal Plane temperature (board 12)
      }
      {
         query =
         {
            ccdId     = CCD_DESELECT (selects the interface board)
            queryId   = DEAHOUSE_CNTL_ADC_FPTEMP_11
```

```
                }
                value        = Focal Plane temperature (board 11)
            }
        }
```

<u>Warnings</u>

1. Instrument Current Draw and Fusing

   At the time of this writing, it is uncertain if one can turn on the Bakeout heater while running 6 FEPs and 6 CCDs without incurring the risk of blowing the fuses to ACIS. To ensure that this doesn't happen, prior to patching or modifying the Bakeout Enable's limit value:

- First, power off the FEPs and Video Boards
- Check the DPA and DEA electrical current draw to ensure that the FEPs and Video boards are off (see Appendix A for a list of current draws under various conditions).
- Once the Bakeout Heater is enabled (NOTE: it may not actually be drawing current - this occurs only when the temperature set-point is greater than the current focal plane temperature), NEVER issue a command to power on the FEPs or Video Boards.

  Ensure that prior to removing the patch:

- Always disable the Bakeout Heater, restore the limit value to 0 and re-boot (warm or cold) the BEP prior to powering the FEPs and Video Boards. NOTE: Just restoring the limit table and performing a warm-boot is NOT SUFFICIENT to disable the Bakeout Heater. One must explicitly disable the Bakeout Heater (a cold-boot will disable the heater, however).

  Note that, since the DEA has an independent power supply, once the heater is enabled and on, it will remain on even if the DPA is powered off.

2. Dirt moves to coldest item

   Since contaminants tend to migrate to the coldest object, when baking out the focal plane, always heat the focal plane prior to enabling the Detector Housing heater.

3.3.7 Controlling other DEA Settings

<u>Description</u>

In addition to the settings described in the previous sections, the following DEA settings are maintained in the System Configuration table. These settings are intended for maintenance and diagnostic activities only, and should not be modified from their defaults without assistance from the ACIS software maintenance team. These settings will be loaded into the instrument with a "Change System Configuration" command, and loaded into the DEA by the System Configuration Task within 1 second of being modified in the System Configuration Table. They include:

```
        SYSSET_CNTL_MASTER_CLOCK      - Disable Signal clock to DEA during
                                        science
```

```
SYSSET_CNTL_LED_ENABLE        - LED to check for holes in filter
                                (requires careful ground analysis of
                                CCD images)
SYSSET_CNTL_HOUSE_HOLD        - Hold Housekeeping address control
SYSSET_CNTL_SIGNAL_PATH       - Select alternate signal path
SYSSET_CNTL_CMDCLOCK_DISABLE  - Disable Command Clock to video boards
                                during science
SYSSET_CNTL_CMDDATA_DISABLE   - Disable data line to video boards dur-
                                ing science
```

The following settings are loaded into the DEA's video boards during the setup stage of a science run:

```
SYSSET_CCD_HOLD_HOUSE         - Hold Video Housekeeping address con-
                                trol (overridden by instrument soft-
                                ware behavior)
SYSSET_CCD_HIGH_SPEED_TAP     - Enable high-speed tap (not useful in
                                flight)
```

Commands

These items are modified as described in earlier sections above using entries within a "Change System Configuration" command packet.

Engineering Telemetry

none

Science Telemetry

The commands to modify these items are echoed as described in earlier sections, and can produce similar Software Housekeeping reports. Only the SYSSET_CCD_HIGH_SPEED_TAP entry will affect the DEA Housekeeping entries, by controlling bit 3 of the DEAHOUSE_CCD_REG_3 register read-back.

3.4 Monitoring the Instrument

This section describes the facilities provided by ACIS for monitoring the health and safety of the instrument, and to verify the receipt and execution of its software commands.

3.4.1 Engineering Housekeeping

Engineering housekeeping information is polled by the spacecraft and inserted in the engineering portion of the telemetry stream, in fixed locations specified by the telemetry format descriptions provided by TRW. Refer to TBD for a description of these items.

3.4.1.1 DPA Voltages, Currents and Temperatures

TBD

3.4.1.2 DEA Voltages, Currents and Temperatures

TBD

3.4.2 System Startup Message

Description

Whenever the instrument software re-boots, either due to a power-on reset, cold reset, warm reset or watchdog reset, the software issues a “Startup Message” telemetry packet. (bepStartupMessage). The ground can determine the type of reset using the following:

TABLE 11. Startup Message Flags to Reset Type

watchdogFlag	warmBootFlag	Reset Type	Reference
0	0	Cold Boot or Power On Boot	Section 3.1.3.1 and Section 3.1.3.2
0	1	Warm Boot	Section 3.1.3.3
1	0	Cold Watchdog Boot	Section 3.1.3.4
1	1	Warm Watchdog Boot	Section 3.1.3.4

The message also indicates the results of the parameter block and table integrity checks. If any of the following message fields are ‘1’, then the associated table has been corrupted. Since these tables reside in SEU-immune RAM, the corruption is probably due to either a software problem, a misdirected write-memory command, or due to a ROM or RAM failure:

```
patchValidFlag      - Patch List has been corrupted.
configFlag          - System Configuration Table has been corrupted,
                      and is now re-loaded from ROM.
parametersFlag      - One or more parameter blocks have been cor-
                      rupted.
```

3.4.3 Software Housekeeping

Approximately once every 64 seconds, the Software Housekeeping task issues a “Software Housekeeping” telemetry packet and updates the engineering LED bi-level values.

3.4.3.1 Bi-Level Telemetry (LEDs)

Description

The engineering bi-level LED values indicate the state of the instrument during boot (see Section 3.1.3 and Section 3.1.4) and, once running under flight software control, indicate the overall state of the instrument. To indicate that the instrument is alive, the Software Housekeeping task toggles the level values. The following lists the bi-level mnemonics associated with each bit of the bi-level code:

LED Bi-level Telemetry Mnemonics

TABLE 12. LED Bi-level Telemetry Mnemonic

Mnemonic	LED Bit Position (lsb first)
1STAT0ST	0
1STAT1ST	1
1STAT2ST	2
1STAT3ST	3

The following lists the LED codes used by the Software Housekeeping task (to review those used by the boot process, refer to Section 3.1).

TABLE 13. Software Housekeeping LED States

Symbol	Bit-Pattern (3 2 1 0)	Instrument State
LED_RUN_IDLE_A	0 1 1 0	Most recent boot was commanded. Not performing science run.
LED_RUN_IDLE_B	0 1 1 1	
LED_RUN_SCIENCE_A	0 1 0 0	Most recent boot was commanded. Performing Science Run.
LED_RUN_SCIENCE_B	0 1 0 1	
LED_WD_IDLE_A	0 0 1 0	Most recent boot was from watchdog timer (patches not installed). Not performing science run.
LED_WD_IDLE_B	0 0 1 1	
LED_WD_SCIENCE_A	0 0 0 0	Most recent boot was from watchdog timer (patches not installed). Performing Science Run.
LED_WD_SCIENCE_B	0 0 0 1	

Warnings

1. TBD Double samples

3.4.3.2 Software Housekeeping Statistics

Description

Once every approximately 64 seconds, the Software Housekeeping task produces a "Software Housekeeping" telemetry packet. This packet consists of a list of entries, which were reported during since the previous housekeeping packet was sent. Each entry consists of an identifier (swStatisticId), a report counter (count), and an entry-specific value field (value). The identifier indicates what was reported, the count indicates how many times it was reported since the previous housekeeping packet (or since the instrument booted), and the associated value supplied with the most recent report. When the housekeeping task is started, and after it issues each telemetry packet, it reports SWSTAT_VERSION, using the "value" field to supply the software version number (this may be overwritten by software maintenance to indicate the current patch version).

If, due to telemetry saturation, a software housekeeping packet is held for additional 64 second periods, the occurrence will be indicated in the SWSTAT_SWHOUSE_SKIPPED, where the "count" indicates the number of 64 second periods skipped, and the "value" field is unused.

If a report is issued and the housekeeping identifier is out-of-range (most likely due to an improperly conceived patch or software error), the SWSTAT_SWHOUSE_RANGE item will be reported instead, where the "value" field is the most recent offending identifier.

3.4.4 Command Echoes

Description

The ACIS software attempts to write "Command Echo" telemetry packets to describe each software command packet it receives. The echo contains a copy of the received command packet, plus an error code, indicating the disposition of the command. If, due to telemetry saturation, a command echo is dropped, a SWSTAT_CMDECHO_DROPPED report will be issued to software housekeeping, where the "value" field contains a copy of the "commandId" of the most recent command whose echo was not telemetered. Although the echo isn't sent, the command is executed.

TBD - fill-in more detail

3.4.5 Fatal Error Messages

Description

If and when the instrument detects an error from which it cannot recover, or which indicates a serious problem with the integrity of the instrument software, the Flight Software issues a "Fatal Error" telemetry message packet, and resets the BEP, using the watchdog timer.

TBD - Fatal Error Detail

3.4.6 DEA Housekeeping Runs

The ACIS software supports acquisition and telemetry of hardware housekeeping information from the Detector Electronics Assembly. The system sends this information under two scenarios. In the first scenario, the ground loads a housekeeping parameter block and then starts a “DEA Housekeeping” run. In the second, the instrument acquires and telemeters DEA video board housekeeping information as part of the setup for a science run.

3.4.6.1 Loading DEA Housekeeping Parameter Block

Description

The DEA Housekeeping Parameter Block contains an ordered list of DEA Housekeeping items to query, and a rate at which to query the entire list (i.e. how often to produce 1 DEA Housekeeping telemetry packet containing the results of each query). A given item may appear more than once in the parameter block, although this is of marginal use. During the acquisition, there is a built-in 0.2 second delay between each item query (NOTE: Video Board ADC Read-Out delays are much longer. See Section 3.4.6.5).

The instrument has five slots in which it stores DEA Housekeeping Parameter blocks. The parameter blocks are loaded into a particular slot using a “Load DEA Housekeeping Parameter Block” command packet. When a slot is loaded, its prior contents are overwritten by the new block contained in the command. If the instrument is power-cycled or cold-booted (see Section 3.1.3.1 , Section 3.1.3.2), the contents of the slots will be restored to their “as-launched” values. A warm-boot (see Section 3.1.3.3), however, retains the contents of the most recently loaded blocks.

Each parameter block contains a checksum field. The checksum is the bitwise XOR of each 16-bit word in the command packet following the checksum field (i.e. starting with the least-significant 16-bit word of the deaBlockId field).

Commands

Load DEA Housekeeping Parameter Block

```
loadDeaBlock: CMDOP_LOAD_DEA
{
   deaBlockSlotIndex   = 0..5     - Specify which slot to load
   checksum            = checksum- 16-bit wide XOR of remainder of
                                   packet (illustrated in bold-type)
   deaBlockId          = id       - Ground selected Parameter Block Id
   sampleRate          = seconds - Frequency of housekeeping packet
   queries[]           =          - Array of items to query
```

```
      {
         ccdId           = I0..S5 or CCD_DESELECT
                                 - Select which Video board to query,
                                   or CCD_DESELECT to choose a DEA
                                   interface board item.
         queryId         = item    - Select item on the chose board to
                                   read
      }
   }
```

For example, the following shows a command which loads a parameter block into slot 3, which when used as part of a DEA Housekeeping run, will read both focal plane temperatures and the board temperature from CCD I3 (see Section 3.3.3 for a description of the relationship between a video board and its associated CCD) once every 10 seconds:

```
   loadDeaBlock: CMDOP_LOAD_DEA
   {
      deaBlockSlotIndex   = 3       - Load block into slot 3
      checksum            = 0x823d  - 16-bit wide XOR of remainder of pkt
      deaBlockId          = 0x1234  - Ground selected Parameter Block Id
      sampleRate          = 10      - Query the set once every 10 seconds
      queries[]           =         - Array of items to query
      {
         ccdId           = CCD_DESELECT- Select interface board
         queryId         = DEAHOUSE_CNTL_ADC_FPTEMP_12
                                 - Query FP temperature
      }
      {
         ccdId           = CCD_DESELECT- Select interface board
         queryId         = DEAHOUSE_CNTL_ADC_FPTEMP_11
                                 - Query FP temperature
      }
      {
         ccdId           = CCD_I3  - Select I3
         queryId         = DDEAHOUSE_CCD_TEMP_BOARD
                                 - Query board temperature
      }
   }
```

Science Telemetry

Command Echo on a successful load:

```
   commandEcho: TTAG_CMD_ECHO
   {
      arrival   = time command arrived (BEP 10Hz Ticks)
      result    = CMDRESULT_OK (other values indicate an error)

      loadDeaBlock: CMDOP_LOAD_DEA
      {
         deaBlockSlotIndex   = 0..5
         checksum            = checksum
         deaBlockId          = id
         sampleRate          = seconds
         queries[]           =
         {
            ccdId            = I0..S5 or CCD_DESELECT
         }
```

```
        }
    }
```

If the checksum of the packet doesn't match its contents, the block will not be loaded into the slot, and the Command echo will report the error by setting the result field to CMDRESULT_STORE_ERROR.

Warnings

1. Actual sample rate is longer by "0.2* # of queries" seconds (none for video ADC)

   The instrument software does not correct the requested sample rate by the inter-sample delay time, nor for jitter due to instrument processor loading and DEA command time-delays. Not counting processor loading and command delays, the actual sample rate is approximately:

   ```
       actual rate := sampleRate + (0.2 seconds * number of queries)
   ```

2. Sample Rate with Video Board ADCs is even longer

   In order to handle the rise-times of the Video Board ADC signal path, the read-time of a video board ADC channel is longer than normal. See Section 3.4.6.5 for a more detailed description.

3. Video Board ADC read-outs require all installed video boards to have power

   See Section 3.4.6.5

4. Power-cycle/Cold Boot resets slots

   A power-cycle or cold reboot of the instrument will reload the contents of the DEA Housekeeping parameter block slots with those provided in ROM.

3.4.6.2 Starting DEA Housekeeping

Description

A DEA Housekeeping Run is initiated using a "Start DEA Housekeeping" command. The slot field in the command specifies which slot contains the DEA Housekeeping parameter block to use for the run.

Once started, the DEA Housekeeping task will accumulate all of the entries specified in the parameter block, with at least 0.2 second delay between each query, into a DEA Housekeeping telemetry buffer and then send the buffer (NOTE: Video Board ADC Read-Out delays are much longer. See Section 3.4.6.5). The task will then sleep for the time requested in the "sampleRate" field of the parameter block, and then repeat the process. The task will continue in this fashion until a "Stop DEA Housekeeping" command is received.

Commands

Start DEA Housekeeping Run:

```
    startDea: CMDOP_START_DEA
```

```
{
   deaBlockSlotIndex   = 0..4    - Select parameter block slot
}
```

For example, the following would start a DEA Housekeeping run using parameters previously loaded into slot 3 (see example in Section 3.4.6.1 Commands):

```
startDea: CMDOP_START_DEA
{
   deaBlockSlotIndex   = 3    - Select parameters loaded into Slot 3
}
```

Science Telemetry

Command Echo on a successful start:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   startDea: CMDOP_START_DEA
   {
      deaBlockSlotIndex   = 0..5
   }
}
```

If the contents of the chosen parameter block slot have been corrupted, but the parameters in slot 0 are intact, the parameters in slot 0 will be used instead of the selected slot. In this case, the run will commence, and the result field of the command echo will be CMDRESULT_CORRUPT_DEFAULT.

If, however, both the contents of the selected slot, and the contents of slot 0 are corrupt, the system will not attempt to start the DEA Housekeeping run. The command echo result field will be CMDRESULT_CORRUPT_IDLE.

Once running, the DEA Housekeeping task will produce 1 DEA Housekeeping telemetry packet each sample period (extended as needed for any read-out delays, or telemetry saturation).

```
deaHousekeepingData: TTAG_DEA_HOUSE
{
   deaBlockId       = Id of DEA Housekeeping parameter block used for
                      the DEA Run (or, if part of a science setup, the
                      Id of the science parameter block. See
                      Section 3.4.6.6 )
   commandId        = Id of the command which started the DEA Run (or,
                      if part of a science setup, the Id of the command
                      which started the Science Run. See
                      Section 3.4.6.6 ).
   bepTickCounter   = BEP 10Hz Tick at start of DEA housekeeping
                      acquisition
   entries[] =      Array of queried housekeeping items
   {
```

```
            query =
            {
                ccdId     = Queried CCD or CCD_DESELECT for interface board
                queryId   = Requested item
            }
            value         = Value of the requested item, or 0xffff if there
                              was an error when the item was queried (NOTE:
                              Housekeeping time-outs may occur during science
                              setup. This is expected.).
        },
        ...
    }
```

Warnings

1. Possible Noise

   In order to reduce noise and improve science data quality, science runs typically disable the video board command clock and command data lines while taking data. In order to acquire housekeeping data from the video boards, the instrument software will temporarily re-enable the command clock and command data lines to the video boards during those queries. This may (this hasn't been quantified) introduce additional noise to the science data while DEA housekeeping is performed on the video boards. To avoid this, don't configure video board queries for housekeeping runs which are run concurrently with science runs.

2. Corrupted Slots

   If the selected slot is corrupt, is not slot 0, and slot 0 is also not corrupt, then slot 0 will be selected for the run. If both slots are corrupt, then the run will not be started.

3. Contention for DEA Interface

   When a DEA Housekeeping run is active, and a science run is in the process of setting up, the DEA Housekeeping task and the Science task arbitrate for access to the DEA command and status channels. Some actions of the Science task can cause the DEA Housekeeping task to time-out (but not vice-versa) waiting for the channel. This is reported as an 0xffff value for the affected query or queries.

3.4.6.3 Stopping DEA Housekeeping

Description

An active DEA Housekeeping run is stopped using a "Stop DEA Housekeeping" command packet. Once the instrument receives this command, it stops its periodic acquisition and telemetry of DEA Housekeeping information. Issuing this command packet when no DEA Housekeeping run is active has no ill effects.

Commands

Stop DEA Housekeeping Run:

```
stopDea: CMDOP_STOP_DEA
{
   No additional command-specific parameters
}
```

Science Telemetry

Command Echo as a result of the stop:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   stopDea: CMDOP_STOP_DEA
   {
   }
}
```

Warnings

None

3.4.6.4 DEA Interface Board Housekeeping

Description

The DEA Interface Board Housekeeping query items cover various thermistor, voltage and discrete items. Except for the relay position item and the focal plane temperature items, the values of interface board housekeeping items are loosely coupled to the software configuration and state of the instrument. The conversions for the voltage and thermistor readings are provided in TBD.

Video Board Relay State

```
DEAHOUSE_CNTL_RELAY               - Reports Video Board Power Relays
                                    (see Section 3.3.3 )
```

Board Thermistors

```
DEAHOUSE_CNTL_ADC_TMP_BEP_PCB     - DPA Thermistor 1 - BEP PC Board
DEAHOUSE_CNTL_ADC_TMP_BEP_OSC     - DPA Thermistor 2 - BEP Oscillator
DEAHOUSE_CNTL_ADC_TMP_FEP0_MONG   - DPA Thermistor 3 - FEP 0 Mongoose
DEAHOUSE_CNTL_ADC_TMP_FEP0_PCB    - DPA Thermistor 4 - FEP 0 PC Board
DEAHOUSE_CNTL_ADC_TMP_FEP0_ACTEL  - DPA Thermistor 5 - FEP 0 ACTEL
DEAHOUSE_CNTL_ADC_TMP_FEP0_RAM    - DPA Thermistor 6 - FEP 0 RAM
DEAHOUSE_CNTL_ADC_TMP_FEP0_FB     - DPA Thermistor 7 - FEP 0 Frame Buf.
DEAHOUSE_CNTL_ADC_TMP_FEP1_MONG   - DPA Thermistor 8 - FEP 1 Mongoose
DEAHOUSE_CNTL_ADC_TMP_FEP1_PCB    - DPA Thermistor 9 - FEP 1 PC Board
DEAHOUSE_CNTL_ADC_TMP_FEP1_ACTEL  - DPA Thermistor 10- FEP 1 ACTEL
DEAHOUSE_CNTL_ADC_TMP_FEP1_RAM    - DPA Thermistor 11- FEP 1 RAM
DEAHOUSE_CNTL_ADC_TMP_FEP1_FB     - DPA Thermistor 12- FEP 1 Frame Buf.
```

Video Board Sub-Housekeeping (TBD - Better description)

```
DEAHOUSE_CNTL_ADC_SUBAHK          - DEA Video Board ADC
```

Focal Plane Temperature

`DEAHOUSE_CNTL_ADC_FPTEMP_12`	- Focal Plane Temp. Board 12
`DEAHOUSE_CNTL_ADC_FPTEMP_11`	- Focal Plane Temp. Board 11

Ground References

`DEAHOUSE_CNTL_ADC_DPAGNDREF1`	- DPA Ground Reference 1
`DEAHOUSE_CNTL_ADC_DPAGNDREF2`	- DPA Ground Reference 2
`DEAHOUSE_CNTL_ADC_GND_1`	- Interface Ground Reference
`DEAHOUSE_CNTL_ADC_GND_2`	- Ground

Voltages

`DEAHOUSE_CNTL_ADC_DPA5VHKA`	- DPA 5V Housekeeping A
`DEAHOUSE_CNTL_ADC_DPA5VHKB`	- DPA 5V Housekeeping B
`DEAHOUSE_CNTL_ADC_DEA28VDCA`	- PSMC A DEA 28V DC
`DEAHOUSE_CNTL_ADC_DEA24VDCA`	- PSMC A DEA 24V DC
`DEAHOUSE_CNTL_ADC_DEAM15VDCA`	- PSMC A DEA -15.5V
`DEAHOUSE_CNTL_ADC_DEAP15VDCA`	- PSMC A DEA +15.5V
`DEAHOUSE_CNTL_ADC_DEAM6VDCA`	- PSMC A DEA -6V DC
`DEAHOUSE_CNTL_ADC_DEAP6VDCA`	- PSMC PSMC A +6V DC
`DEAHOUSE_CNTL_ADC_DEA28VDCB`	- PSMC B DEA 28V DC
`DEAHOUSE_CNTL_ADC_DEA24VDCB`	- PSMC B DEA 24V DC
`DEAHOUSE_CNTL_ADC_DEAM15VDCB`	- PSMC B DEA -15.5V DC
`DEAHOUSE_CNTL_ADC_DEAP15VDCB`	- PSMC B DEA +15.5V DC
`DEAHOUSE_CNTL_ADC_DEAM6VDCB`	- PSMC B DEA -6V DC
`DEAHOUSE_CNTL_ADC_DEAP6VDCB`	- PSMC B DEA +6V DC

Current Measurements (radiation damage monitoring TBD - Better description)

`DEAHOUSE_CNTL_ADC_RAD_PCB_A`	- Relative Dose Rad. Monitor Side A
`DEAHOUSE_CNTL_ADC_RAD_PCB_B`	- Relative Dose Rad. Monitor Side B

3.4.6.5 Video Board Housekeeping (all boards must be powered)

Description

The DEA Housekeeping items for the Video boards include the current values in the video board control registers, and some analog housekeeping, which include measurements of the CCD clocking voltage rails and some video board temperatures (NOTE: To obtain sensible values for the analog housekeeping items, all 10 video boards must be powered on. This is due to the analog multiplexer built into the video boards).

Due to the behavior of the Video Board ADC signal path, the read-out procedure of a video board ADC channel takes longer than all other read-outs. In order to “pre-charge” the capacitance of the signal path, the instrument software reads a queried video board ADC housekeeping value 5 times, with a 0.1 second delay between each read. This adds 1/2 second to the actual sample time for each video board ADC channel being read during a DEA Housekeeping run. The adjusted formula for the actual sample rate is:

```
actual rate :=sampleRate + (0.2 seconds * number of queries) + (0.5
             seconds * number of video board queries)
```

The register state values are directly coupled to the current state of the video boards, and the clocking voltage rails are directly coupled to the state of the DAC settings used to configure the boards. The board temperatures, however, are only loosely coupled to the configured state of the boards. The conversions for the voltage and thermistor readings are provided in TBD.

Video Board Registers

```
DEAHOUSE_CCD_REG_0  - Sequencer Control Register
{
   bit 0     - Sequencer Start (1 - Sequencer Started, 0 - N/A)
   bit 1     - Sequencer Stop (1 - Sequencer Stopped, 0 - N/A)
   bits 2..7 - Sequencer Offset (see Section 3.3.4
               SYSSET_CCD_SEQ_OFFSET)
}
DEAHOUSE_CCD_REG_1  - Video A/D Converter Register
{
   bit 0     - A/D-Cycle Start (1 - ADC Started, 0 - N/A)
   bit 1     - A/D-Cycle Stop (1 - ADC Stopped, 0 - N/A)
   bits 2..7 - A/D-Cycle Offset Delay Count (see Section 3.3.4
               SYSSET_CCD_ADC_OFFSET)
}
DEAHOSUE_CCD_REG_2  - Test Aid Register
{
   bit 0     - Video Output A (0 - disabled, 1 - enabled, see
               Section 3.3.4 SYSSET_CCD_VIDEO_ENABLE)

   bit 1     - Video Output B (0 - disabled, 1 - enabled, see
               Section 3.3.4 SYSSET_CCD_VIDEO_ENABLE)

   bit 2     - Video Output C (0 - disabled, 1 - enabled, see
               Section 3.3.4 SYSSET_CCD_VIDEO_ENABLE)

   bit 3     - Video Output D (0 - disabled, 1 - enabled, see
               Section 3.3.4 SYSSET_CCD_VIDEO_ENABLE)
   bit 4..5  - Unused
   bit 6     - Hold Video Housekeeping Address (0 - un-held, 1 - held)
               The instrument software holds this bit only when read-
               ing ADC values. Since the DEAHOUSE_CCD_REG_2 is NOT an
               ADC item, the read value of this bit should always be 0.
               Also, see Section 3.3.7 SYSSET_CCD_HOLD_HOUSE.
   bit 7     - Unused
}
DEAHOUSE_CCD_REG_3  - Miscellaneous Register
{
   bit 0     - Unused
   bit 1     - Back-Junction Diode (0 - off, 1 - on, see Section 3.3.4
               SYSSET_CCD_BJD)
   bit 2     - Clock-swap (0 - 4 node clocking, 1 - 2 node clocking,
               see Section 4.1.4 ). This field should be 0 when in FULL
               or DIAG clocking mode, and 1 when in AC or BD clocking
               mode.
   bit 3     - High-Speed Tap (0 - disabled, 1 - enabled, see
               Section 3.3.7 SYSSET_CCD_HIGH_SPEED_TAP)
   bit 4     - Status Enable (0 - disabled, 1 - enabled) This bit must
               be 1 in order for a video board to respond to any query,
               including the query to read the register, and therefore
               will always read as a 1.
   bit 5..7  - Unused
}
```

Video Board Clocking Signal Rail and Bias Voltages

```
DEAHOUSE_CCD_PIA_P      - Image Array Parallel +
DEAHOUSE_CCD_PIA_M      - Image Array Parallel -
DEAHOUSE_CCD_PFS_P      - Framestore Parallel +
DEAHOUSE_CCD_PFS_M      - Framestore Parallel -
DEAHOUSE_CCD_S_P        - Serial Register +
DEAHOUSE_CCD_S_M        - Serial Register -
DEAHOUSE_CCD_R_P        - Reset Gate +
DEAHOUSE_CCD_R_M        - Reset Gate -
DEAHOUSE_CCD_OG         - Output Gate Bias Level
DEAHOUSE_CCD_SCP        - Scupper
DEAHOUSE_CCD_RD         - Reset Diode
DEAHOUSE_CCD_DR0        - Drain Output Channel A
DEAHOUSE_CCD_DR1        - Drain Output Channel B
DEAHOUSE_CCD_DR2        - Drain Output Channel C
DEAHOUSE_CCD_DR3        - Drain Output Channel D
```

Video Board Temperatures

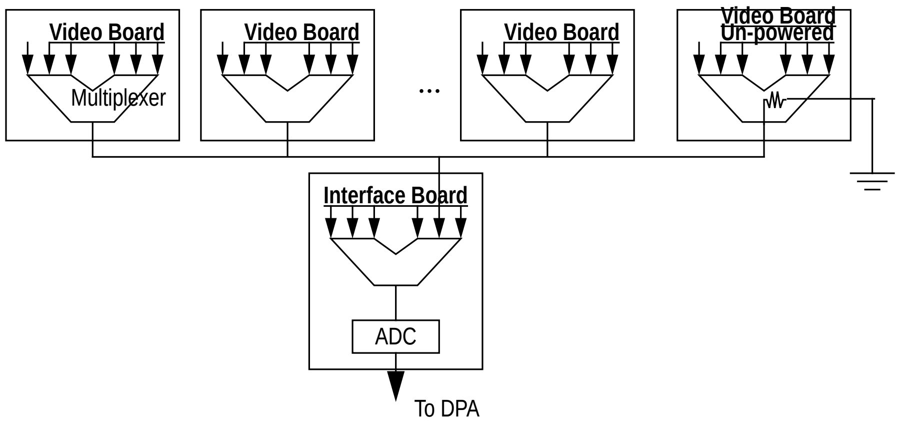
```
DEAHOUSE_CCD_TEMP_BOARD - Board Temperature (RTD4)
DEAHOUSE_CCD_TEMP_SRAM  - SRAM Temperature (RTD3)
DEAHOUSE_CCD_TEMP_ADC   - ADC Temperature (RTD2)
DEAHOUSE_CCD_TEMP_ACTEL - Gate Array Temperature (RTD1)
```

Warnings

1. Powered off boards drag down ADC read-outs

   The following roughly illustrates the architecture of the video board analog housekeeping:

FIGURE 3. Video Board ADC Signal Path



The analog multiplexers on the Video boards shunt to ground when powered off, and all outputs from these multiplexers are tied together before going into the interface board multiplexer and housekeeping ADC. This drags down the housekeeping readings from the powered video boards. Under these conditions, the read values can indicate that the read channel isn't dead, but not much more than that. However, if all of the video boards have power, then the read values reflect a close approximation to the actual voltage, current or temperature level.

2. ADC channels take time to read

   Due to the charging time of the video board ADC signal path, the instrument software reads a Video Board ADC channel 5 times, with a 0.1 second delay between each read. This leads to minimum of 0.5 seconds for each read of an ADC video board channel.

3. Some things should never change

   Due to the behavior of the instrument software, certain bits in the Video Board registers should always read the same value. These are:

```
DEAHOUSE_CCD_REG_2 bit 6 - Hold Housekeeping   - Always reads as 0
DEAHOUSE_CCD_REG_3 bit 4 - Status Enable       - Always reads as 1
```

4. Video Board Parameters aren't loaded until Science Run

   Since a Science run globally resets the video boards, the video board parameters are not loaded into the boards until the setup stage of a Science Run (see Section 3.3.4), and, therefore, the affected DEA Video Board housekeeping values will not reflect any System Configuration changes until a science run starts.

3.4.6.6 Standard DEA Housekeeping at Start of Science Run

Description

During the setup of a Science Run (see Section 4.1), the Science task (as opposed to the DEA Housekeeping task) resets all of the DEA video boards, loads the parameters into the boards, and loads the contents of the video boards Sequencer and Program RAM (SRAM/PRAM), and then reads all of the housekeeping channels on each video board used for the science run (it will not read housekeeping from boards not used for the run). The task then packages the read housekeeping values into a DEA Housekeeping Telemetry packet, setting the deaBlockId field of the packet to the parameter block id, parameterBlockId, of the Science Parameter block, loadTeBlock or loadCcBlock, used to configure the science run (as opposed to the deaBlockId field of a loadDeaBlock).

Commands

This housekeeping pass occurs as the result of a science run. See Section 4.1.5 .

Science Telemetry

Science Run DEA Housekeeping Telemetry Packet

```
deaHousekeepingData: TTAG_DEA_HOUSE
{
   deaBlockId       = Id of Science Parameter Block used to configure
                      the science run (either a loadTeBlock, or
                      loadCcBlock)
   commandId        = Id of the command which started the Science Run
   bepTickCounter   = BEP 10Hz Tick at start of Science Run DEA house-
                       keeping acquisition
   entries[] =      Array of queried housekeeping items
   {
      For each selected CCD, the following items will have the form:
      query =
      {
         ccdId     = Selected CCD
         queryId   = Requested item
      }
      value        = Read Value
   },
   ...
}
```

Since the science run is actively controlling the video boards during the setup, the read Video Board Registers should have the following values:

```
DEAHOUSE_CCD_REG_0  - Sequencer Control Register
{
   bit 0     := 0         - Sequencer Start (sequencer not started)
   bit 1     := 1         - Sequencer Stop (sequencer is stopped)
   bits 2..7 := value of SYSSET_CCD_SEQ_OFFSET
}
DEAHOUSE_CCD_REG_1  - Video A/D Converter Register
{
   bit 0     := 1         - A/D-Cycle Start (ADC started)
   bit 1     := 0         - A/D-Cycle Stop (ADC not stopped)
   bits 2..7 := value of SYSSET_CCD_ADC_OFFSET
}
DEAHOSUE_CCD_REG_2  - Test Aid Register
{
   bit 0     := bit 0 of SYSSET_CCD_VIDEO_ENABLE
   bit 1     := bit 1 of SYSSET_CCD_VIDEO_ENABLE
   bit 2     := bit 2 of SYSSET_CCD_VIDEO_ENABLE
   bit 3     := bit 3 of SYSSET_CCD_VIDEO_ENABLE
   bit 4..5  := undefined
   bit 6     := always 0
   bit 7     := undefined
}
DEAHOUSE_CCD_REG_3  - Miscellaneous Register
{
   bit 0     := undefined
   bit 1     := value of SYSSET_CCD_BJD
   bit 2     := 0 if in FULL/DIAG Mode
             := 1 if in AC or BD Mode
   bit 3     := value of SYSSET_CCD_HIGH_SPEED_TAP
   bit 4     := always 1
   bit 5..7  := undefined
}
```

In addition to the above, the housekeeping packet will report the following for each configured board:

```
DEAHOUSE_CCD_PIA_P      - Image Array Parallel +
DEAHOUSE_CCD_PIA_M      - Image Array Parallel -
DEAHOUSE_CCD_PFS_P      - Framestore Parallel +
DEAHOUSE_CCD_PFS_M      - Framestore Parallel -
DEAHOUSE_CCD_S_P        - Serial Register +
DEAHOUSE_CCD_S_M        - Serial Register -
DEAHOUSE_CCD_R_P        - Reset Gate +
DEAHOUSE_CCD_R_M        - Reset Gate -
DEAHOUSE_CCD_OG         - Output Gate Bias Level
DEAHOUSE_CCD_SCP        - Scupper
DEAHOUSE_CCD_RD         - Reset Diode
DEAHOUSE_CCD_DR0        - Drain Output Channel A
DEAHOUSE_CCD_DR1        - Drain Output Channel B
DEAHOUSE_CCD_DR2        - Drain Output Channel C
DEAHOUSE_CCD_DR3        - Drain Output Channel D
DEAHOUSE_CCD_TEMP_BOARD - Board Temperature (RTD4)
DEAHOUSE_CCD_TEMP_SRAM  - SRAM Temperature (RTD3)
DEAHOUSE_CCD_TEMP_ADC   - ADC Temperature (RTD2)
DEAHOUSE_CCD_TEMP_ACTEL - Gate Array Temperature (RTD1)
```

Warnings

1. ADC/Temperature readings are just flags unless all installed boards are powered

   See Section 3.4.6.5 Warnings

2. For science runs, deaBlockId is the parameterBlockId

   If the user assigns distinct block ids for DEA Housekeeping parameter blocks and Science Parameter blocks, the deaBlockId within housekeeping telemetry packets can be used to discriminate between housekeeping packets produced by a DEA Housekeeping run, v.s. packets produced during the setup for a science run.

3.5 Managing the Bad Pixel and Column Maps

The ACIS Instrument software maintains lists of bad pixels and vertical columns for each CCD. The instrument uses these lists to reduce the amount of processing time and telemetry consumed by “fake” events produced by pixels and columns which are known to be producing erroneous information. Over the life of the instrument, pixels and columns within the CCDs will erode due to radiation damage.

3.5.1 Adding Bad Pixels and Columns

Description

ACIS supports a single “Bad Pixel Map’, and two “Bad Column Maps”, one for Timed Exposure science runs, and one for Continuous Clocking science runs. The Bad Pixel Map can contain up to 10,000 entries, and each Bad Column Map can contain up to 10,240 entries each (i.e. you can eliminate every column if you want to).

The user can append a set of new bad pixels to the on-board bad pixel map using a “Add Bad Pixel” command packet, which contains a list of zero or more pixel entries to add. Each entry specifies which CCD contains the pixel, and the row and column position of the pixel within that CCD.

One uses the “Add Bad Te Column” or “Add Bad Cc Column” command packet to add a list of zero or more bad column entries to the Timed Exposure Bad Column Map and Continuous Clocking Bad Column map, respectively. Each entry specifies which CCD contains the column, and the column position within that CCD.

Once loaded, the bad pixel and column maps are only used after a bias-map has been recomputed. After a bias map for a CCD has been recomputed, the bad pixel and column maps are scanned. If an entry is found for the CCD, the corresponding pixel(s) bias map value is set to “PIXEL_BAD” (for columns, all 1024 pixels in the column are flagged). This prevents the system from reporting events centered on these pixels. For Timed Exposure science runs, the Bad Pixel map and Timed Exposure Bad Column map are used. For Continuous Clocking science runs, only the Continuous Clocking Bad Column Map is used.

Refer to Section 3.6.2.1 for a description of how to dump the contents of the Bad Pixel and Column maps.

NOTE: When the BEP is power-cycled or cold booted, the bad pixel map and bad column maps are reset and re-loaded with the default lists stored in ROM. All pixels and columns commanded prior to the reset are lost. A warn-boot, however, preserves the lists.

Commands

Adding one or more Bad Pixels

```
addBadPixel: CMDOP_ADD_BAD_PIXEL
{
   pixels[] =   Array of pixel entries
   {
      ccdId     = I0..S5  - Which CCD contains the pixel
      ccdRow    = 0..1023 - CCD row position of the pixel
      ccdColumn = 0..1023 - CCD column position of the pixel
   }
   ...
}
```

Adding one or more Bad Timed Exposure Columns

```
addBadColumn: CMDOP_ADD_BAD_TE_COL
{
   columns[] =  Array of column entries
   {
      ccdId     = I0..S5  - Which CCD contains the column
      ccdColumn = 0..1023 - CCD column position
   }
   ...
}
```

Adding one or more Bad Continuous Clocking Columns

```
addBadColumn: CMDOP_ADD_BAD_CC_COL
{
   columns[] =  Array of column entries
   {
      ccdId     = I0..S5  - Which CCD contains the column
      ccdColumn = 0..1023 - CCD column position
   }
   ...
}
```

For example, to load a bad pixel to the top right corner of I0 and S2 and flag column 511 on CCD S3 for Continuous Clocking mode, the following two commands are sent:

```
addBadPixel: CMDOP_ADD_BAD_PIXEL
{
   pixels[2] =
   {
      ccdId     = I0
      ccdRow    = 1023
      ccdColumn = 1023
   },
   {
      ccdId     = S2
      ccdRow    = 1023
      ccdColumn = 1023
   }
}
addBadColumn: CMDOP_ADD_BAD_CC_COL
{
   columns[1] =
   {
      ccdId     = S5
      ccdColumn = 511
   }
}
```

Science Telemetry

If a load bad pixel map command is successful, the command echo reports:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   addBadPixel: CMDOP_ADD_BAD_PIXEL
   {
      pixels[] =   Array of pixel entries
      {
         ccdId     = specified CCD
         ccdRow    = specified row
         ccdColumn = specified column
      }
      ...
   }
}
```

If a load bad te column map command is successful, the command echo reports:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   addBadColumn: CMDOP_ADD_BAD_TE_COL
   {
      columns[] =  Array of column entries
      {
         ccdId     = specified CCD
         ccdColumn = specified column
      }
      ...
   }
}
```

If a load bad cc column map command is successful, the command echo reports:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   addBadColumn: CMDOP_ADD_BAD_CC_COL
   {
      columns[] =  Array of column entries
      {
         ccdId     = specified CCD
         ccdColumn = specified column
      }
```

```
            ...
        }
    }
```

If one or more entries cannot be stored because the corresponding table is full, the command echo “result” field will report: CMDRESULT_TABLE_FULL.

If one or more entries contains an argument which is out of range, the command echo “result” field will report: CMDRESULT_BAD_ARGUMENT, and the offending and remaining entries in the command packet will not be added to the map.

Warnings

1. Power-Cycle or Cold Boot loads default lists

   If the BEP is power-cycled or performs a cold re-boot, all previously loaded bad pixels and columns are removed and overwritten with the default lists contained in ROM. See Section 3.5.2 to set the lists to an empty state.

2. Re-compute bias maps whenever the pixel/column maps change

   The implementation of the bad pixel and column filtering uses special codes poked into the FEP bias map to eliminate those pixels from data processing. These codes are written into the maps after the maps have been re-computed. If any pixels or columns have been added since the last bias-recomputation, new bias maps must be computed. If any pixels or columns are removed from the lists, the bias maps must be recomputed to produce valid bias values for the corresponding locations in the map.

3.5.2 Resetting the on-board maps

Description

To empty the contents of the Bad Pixel Map, the user issues a “Reset Bad Pixel Map” command, and to empty the contents of the Bad Column Maps, the user issues a “Rest Bad Column” command.

Since there are default lists loaded from ROM upon power-on and cold boot, the bad pixel and column maps are not empty upon start-up. The “reset” commands are required to empty the lists.

Commands

To empty the contents of the Bad Pixel Map,

```
    resetBadPixelMap: CMDOP_RESET_BAD_PIXEL
    {
        No additional parameters
    }
```

To empty the contents of the Timed Exposure Bad Column Map,

```
    resetBadColumnMap: CMDOP_RESET_BAD_TE_COL
```

```
{
   No additional parameters
}
```

To empty the contents of the Continuous Clocking Bad Column Map,

```
resetBadColumnMap: CMDOP_RESET_BAD_CC_COL
{
   No additional parameters
}
```

Science Telemetry

The command echo to the Reset Bad Pixel Map command reports:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   resetBadPixelMap: CMDOP_RESET_BAD_PIXEL
   {
   }
}
```

The command echo to the Timed Exposure and Continuous Clocking Reset Bad Column Map commands are, respectively:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   resetBadPixelMap: CMDOP_RESET_BAD_TE_COL
   {
   }
}
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   resetBadPixelMap: CMDOP_RESET_BAD_CC_COL
   {
   }
}
```

Warnings

1. Same as Section 3.5.1 Warnings

3.6 Memory Dumps

ACIS implements all of its memory reads using the Memory Server task. This task is capable of trickling out one contiguous region of memory at a time, using one or more telemetry packets, concurrently with science processing runs and DEA housekeeping runs. In order to prevent erroneously large memory dumps from preventing subsequent dump commands from being executed, all flavors of memory dump commands will abort any dump in progress.

3.6.1 General Memory Reads[1]

This section describes reads of general areas of memory. For these types of reads, the user must know the type of memory to read, the address of the region to read, and the number of words to read. To see special "tagged" memory reads, where the instrument determines the address and length of the read, see Section 3.6.2 .

NOTE: In general, any memory read may be aborted by a subsequent memory read, write or execute command.

3.6.1.1 BEP Memory Read

Description

The user issues a "Read Bep" command to perform a general read of BEP memory, specifying the starting address to read, and the number of 32-bit words to read. Any location in the BEP address space can be read using a "Read Bep" command, with the following constraints:

- The start address must be aligned to a 32-bit word boundary (i.e. evenly divisible by 4)

  If not, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo "result" field.
- The region being read must not cross an Instruction Cache address boundary

  These addresses are: 0x80080000 and 0x800fffff. If the region crosses one of these boundaries, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo "result" field.
- If the region is in a FEP shared memory area, the associated FEP must be powered on

  The FEP shared memory regions are:

```
        FEP 0  - 0xa8000000 0xa8ffffff
        FEP 1  - 0xa9000000 0xa9ffffff
```

1. acisBepUnix Users: Certain General Memory reads for the Unix simulation, acisBepUnix, won't work the same way. You need to locate the address of interest in acisBepUnix.map, and use that address. In some cases, certain regions, such as the ROM, don't exist on the simulation.

```
FEP 2  - 0xaa000000 0xaaffffff
FEP 3  - 0xab000000 0xabffffff
FEP 4  - 0xac000000 0xacffffff
FEP 5  - 0xad000000 0xadffffff
```

If the FEP does not have power, a BEP Bus Error Exception will be raised and will cause a Fatal Error and subsequent BEP re-boot.

Commands

General read of BEP Memory

```
readBep: CMDOP_READ_BEP
{
   readAddress  = start   - Starting Address (must be 32-bit aligned)
   wordCount    = # words - Number of 32-bit words to read
}
```

For example, to read the entire contents of the BEP's Data Cache:

```
readBep: CMDOP_READ_BEP
{
   readAddress  = 0x80000000 - Start of D-cache
   wordCount    = 0x10000    - Number of 32-bit words (256Kbytes)
}
```

Another example, to read the entire contents of the BEP's Instruction Cache:

```
readBep: CMDOP_READ_BEP
{
   readAddress  = 0x80080000 - Start of I-cache
   wordCount    = 0x20000    - Number of 32-bit words (512Kbytes)
}
```

Science Telemetry

Command Echo of a generic BEP read request:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   readBep: CMDOP_READ_BEP
   {
      readAddress  = specified start address
      wordCount    = specified word count
   }
}
```

If the read interrupts an earlier on-going read, the Command Echo “result” field will contain CMDRESULT_CLOBBERED.

If the read attempts to interrupt a Write-Memory or Execute-Memory activity, the read is not performed. In this case, the Command Echo “result” field will contain CMDRESULT_BUSY.

If the address region spans an I-cache boundary, or if the starting address is not aligned to a 32-bit word boundary, the command will be rejected, and the “Command Echo “result” field will contain CMDRESULT_ARGUMENT.

Once executing, the Memory Server task will produce a series of “BEP Read Reply” telemetry packets of the form:

```
bepReadReply: TTAG_READ_BEP
{
   commandId          = Command Id of read command
   bepTickCounter     = BEP 10Hz Tick counter at start of the read
   requestedAddress   = original specified address
   requestedWordCount = original specified word count
   readAddress        = start address of data in readData[]. For each
                        but the first packet in the series, this
                        should be the readAddress of the previous
                        telemetry packet plus the number of bytes
                        (i.e. words * 4) in the previous packet’s
                        readData[]
   readData[]         = array of read 32-bit words
}
```

The Memory Server uses telemetry packet buffers of 1023 32-bit words. The header portion of each “BEP Read Reply” telemetry packet is 7 words, leaving 1016 32-bit data words for each “BEP Read Reply” telemetry packet. Given a “wordCount” for any region of memory, the Memory Server will use the following number of telemetry packets to send that region:

```
wordCount / (1023 - 7) (+ 1 if wordCount % (1023 - 7) != 0)
```

Warnings

1. Aborts earlier read

   If, when the command is received, the Memory Server task is executing an earlier read command, such as a dump of the Huffman Tables, the earlier read will be aborted, possibly causing incomplete information to be telemetered. The new read will then commence.

2. Ignored if Write-Memory or Execute-Memory being performed

   If a Write-Memory command or Execute-Memory command is being executed by the Memory Server task when a read request is received (or even if a new Write-Memory or Execute-Memory command is received), the new command will NOT be executed.

3. Fatal Errors due to reads of un-powered FEP shared memory

   Attempts to read shared memory regions belonging to FEPs which do not have power may result in a BEP Bus Error exception, Fatal Error Message and BEP re-boot.

3.6.1.2 ROM Memory Read

Description

The ACIS ROM is memory-mapped into the BEP’s address space, and is located at memory location 0xbfc00000 and contains 0x40000 32-bit words. To dump the contents of the entire ROM, the user issues a “Read Bep” command, specifying the address and length listed above.

Commands

Command to Read all of ROM:

```
readBep: CMDOP_READ_BEP
{
   readAddress  = 0xbfc00000 - Start of BEP ROM
   wordCount    = 0x40000    - # 32-bit words in ROM (1Mbyte)
}
```

Science Telemetry

Command Echo of read request:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   readBep: CMDOP_READ_BEP
   {
      readAddress  = 0xbfc00000 - Start of BEP ROM
      wordCount    = 0x40000    - # 32-bit words in ROM (1Mbyte)
   }
}
```

If the read interrupts an earlier on-going read, the Command Echo “result” field will contain CMDRESULT_CLOBBERED.

If the read attempts to interrupt a Write-Memory or Execute-Memory activity, the read is not performed. In this case, the Command Echo “result” field will contain CMDRESULT_BUSY.

Once executing, the Memory Server task will produce a series of “BEP Read Reply” telemetry packets of the form:

```
bepReadReply: TTAG_READ_BEP
{
   commandId           = Command Id of read command
   bepTickCounter      = BEP 10Hz Tick counter at start of the read
   requestedAddress    = 0xbfc000000
   requestedWordCount  = 0x40000
   readAddress         = start address of data in readData[]. For each
                         but the first packet in the series, this
                         should be the readAddress of the previous
                         telemetry packet plus the number of bytes
                         (i.e. words * 4) in the previous packet’s
                         readData[]
   readData[]          = array of read 32-bit words
}
```

Warnings

1. See Section 3.6.1.1 Warnings

3.6.1.3 FEP Memory Read

Description

The user issues a “Read Fep” command to perform a general read of FEP memory, specifying the FEP, the starting address to read, and the number of 32-bit words to read. Any location in the FEP address space can be read using a “Read Fep” command, with the following constraints:

- The start address must be aligned to a 32-bit word boundary (i.e. evenly divisible by 4)

  If not, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo “result” field.

- The region being read must not cross an Instruction Cache address boundary

  These addresses are: 0x80080000 and 0x800fffff. If the region crosses one of these boundaries, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo “result” field.

- The FEP must be powered when the read command is received

  Attempts to read from a FEP when the FEP is not powered will cause the read to be rejected. The Command Echo result field will be set to CMDRESULT_BOARD_OFF.

- The FEP must remain powered during the read

  Attempts to read a FEPs memory when the FEP is not powered may cause the read to abort, or possibly cause a BEP Bus Error Exception, which will subsequently cause a Fatal Error and BEP re-boot.

Commands

General read of FEP Memory

```
readFep: CMDOP_READ_FEP
{
   fepId        = FEP_0..FEP_5   - FEP to read from
   readAddress  = start          - Starting Address (must be 32-bit
                                   aligned)
   wordCount    = # words        - Number of 32-bit words to read
}
```

For example, to read the entire contents of the FEP 3’s Data Cache:

```
readFep: CMDOP_READ_FEP
{
   fepId        = FEP_3      - Read from FEP 3
   readAddress  = 0x80000000 - Start of D-cache
   wordCount    = 0x10000    - Number of 32-bit words (256Kbytes)
}
```

Another example, to read the entire contents of the FEP 5's Instruction Cache:

```
readFep: CMDOP_READ_FEP
{
   fepId        = FEP_5      - Read from FEP 5
   readAddress  = 0x80080000 - Start of I-cache
   wordCount    = 0x20000    - Number of 32-bit words (512Kbytes)
}
```

Science Telemetry

Command Echo of a generic FEP read request:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   readFep: CMDOP_READ_FEP
   {
      fepId        = specified FEP
      readAddress  = specified start address
      wordCount    = specified word count
   }
}
```

If the read interrupts an earlier on-going read, the Command Echo "result" field will contain CMDRESULT_CLOBBERED.

If the read attempts to interrupt a Write-Memory or Execute-Memory activity, the read is not performed. In this case, the Command Echo "result" field will contain CMDRESULT_BUSY.

If the address region spans an I-cache boundary, or if the starting address is not aligned to a 32-bit word boundary, the command will be rejected, and the "Command Echo "result" field will contain CMDRESULT_ARGUMENT.

If an attempt is made to read from a FEP whose power is off, the command will be rejected and the Command Echo "result" field will contain CMDRESULT_BOARD_OFF.

Once executing, the Memory Server task will produce a series of "FEP Read Reply" telemetry packets of the form:

```
fepReadReply: TTAG_READ_FEP
{
   commandId           = Command Id of read command
   fepId               = original specified FEP
   bepTickCounter      = BEP 10Hz Tick counter at start of the read
   requestedAddress    = original specified address
   requestedWordCount  = original specified word count
   readAddress         = start address of data in readData[]. For each
                         but the first packet in the series, this
                         should be the readAddress of the previous
                         telemetry packet plus the number of bytes
```

```
                               (i.e. words * 4) in the previous packet’s
                               readData[]
        readData[]           = array of read 32-bit words
    }
```

The Memory Server uses telemetry packet buffers of 1023 32-bit words. The header portion of each “FEP Read Reply” telemetry packet is 7 words, leaving 1016 32-bit data words for each “FEP Read Reply” telemetry packet. Given a “wordCount” for any region of memory, the Memory Server will use the following number of telemetry packets to send that region:

```
    wordCount / (1023 - 7) (+ 1 if wordCount % (1023 - 7) != 0)
```

Warnings

1. See Section 3.6.1.1 Warnings
2. Reads from Powered Off FEPs may reset BEP

   Attempts to read FEP memory from FEPs which do not have power may result in a BEP Bus Error exception, Fatal Error Message and BEP re-boot.
3. Shared Memory Reads are direct

   The BEP implements FEP read requests to shared memory regions by just reading the memory directly. This may affect the performance of on-going science by stealing bus cycles away from the FEP.
4. Non-shared Memory Reads need help from FEP

   The BEP implements FEP read requests to non-shared memory regions by issuing requests to the FEP via the BEP-to-FEP software mailbox. This may interfere with the performance of on-going science by actively requesting interactions with the software running on the FEP. These types of reads also require that the FEP software be up-and running, and will time-out if the FEP software has crashed or if the FEP is held in a reset state.
5. Don’t read Bias/T-plane/P-plane during a Science Run

   The FEP hardware cannot reliably handle science data processing and BEP requests to read bias-memory and threshold or parity plane simultaneously. This may cause writes of the hardware to the threshold plane and parity planes to cease, causing loss of science data from the affected FEP. A reset of the FEP is required to recover from this condition, and FEPs are only routinely reset when they are power-cycled, are re-loaded at the start of a science run after a FEP crash, or when a science run is aborted due to the radiation monitor.

3.6.1.4 DEA Video Board Memory Read

Description

The DEA Video Boards contain two types of readable RAM: Sequencer RAM (SRAM) and Program RAM (PRAM). When the sequencers aren’t running, the user can issue a “Read SRAM” command to perform a general read of Video Board Sequencer memory, specifying the CCD associated with the board (see Section 3.3.4), the starting address to read, and the number of 16-bit words to read. To read a board’s Program RAM, the user can issue a “Read PRAM” command. These commands can only be issued under the following conditions:

- The selected Video Board must be powered

  Attempts to read from a Video Board when the board is not powered will cause the read to be rejected. The Command Echo result field will be set to CMDRESULT_BOARD_OFF.

- The sequencers must be idle

  If a science or bias run is in progress when a read is requested, or starts while a read request is active, the read will be aborted. This is reflected in Software Housekeeping with a SWSTAT_DEABOARD_ERROR entry.

Note: PRAM/SRAM contain trash until a science run. The contents of a video board’s SRAM and PRAM are undefined until the first science run after the video board has been powered on, or until a “Write PRAM” or “Write SRAM” command is received which loads their contents.

Commands

Read SRAM

```
readSram: CMDOP_READ_SRAM
{
   ccdId     = I0..S5      - Selects which video board to read from
   readIndex = 0..0x7fff   - Starting SRAM address
   wordCount = 0..0x8000   - Number of 16-bit words to read
}
```

Read PRAM

```
readPram: CMDOP_READ_PRAM
{
   ccdId     = I0..S5      - Selects which video board to read from
   readIndex = 0..0x7fff   - Starting PRAM address
   wordCount = 0..0x8000   - Number of 16-bit words to read
}
```

For example, the command to dump all of CCD I3’s PRAM is as follows:

```
readPram:CMDOP_READ_PRAM
{
   ccdId     = I3
   readIndex = 0
   wordCount = 0x8000
}
```

Science Telemetry

Command Echo of a generic SRAM read request:

```
commandEcho: TTAG_CMD_ECHO
```

```
        {
           arrival   = time command arrived (BEP 10Hz Ticks)
           result    = CMDRESULT_OK (other values indicate an error)

           readSram: CMDOP_READ_SRAM
           {
              ccdId        = specified Video Board
              readIndex    = specified start index
              wordCount    = specified word count
           }
        }
```

Command Echo of a generic PRAM read request:

```
        commandEcho: TTAG_CMD_ECHO
        {
           arrival   = time command arrived (BEP 10Hz Ticks)
           result    = CMDRESULT_OK (other values indicate an error)

           readFep: CMDOP_READ_PRAM
           {
              ccdId        = specified Video Board
              readIndex    = specified start index
              wordCount    = specified word count
           }
        }
```

If the read interrupts an earlier on-going read, the Command Echo “result” field will contain CMDRESULT_CLOBBERED.

If the read attempts to interrupt a Write-Memory or Execute-Memory activity, the read is not performed. In this case, the Command Echo “result” field will contain CMDRESULT_BUSY.

If the address region spans an I-cache boundary, or if the starting address is not aligned to a 32-bit word boundary, the command will be rejected, and the “Command Echo “result” field will contain CMDRESULT_ARGUMENT.

If an attempt is made to read from a Video Board whose power is off, the command will be rejected and the Command Echo “result” field will contain CMDRESULT_BOARD_OFF.

Once executing, the Memory Server task will produce a series of “SRAM Read Reply” or “PRAM Read Reply” telemetry packets of the form:

```
        sramReadReply: TTAG_READ_SRAM
        {
           commandId           = Command Id of read command
           ccdId               = original specified Video Board
           bepTickCounter      = BEP 10Hz Tick counter at start of the read
           requestedIndex      = original specified index
           requestedWordCount  = original specified word count
           readIndex           = start address of data in readData[]. For each
                                 but the first packet in the series, this
                                 should be the readIndex of the previous
                                 telemetry packet plus the number of bytes
```

```
                              (i.e. words * 2) in the previous packet’s
                              readData[]
    readData[]              = array of read 16-bit words
}
pramReadReply: TTAG_READ_PRAM
{
    commandId               = Command Id of read command
    ccdId                   = original specified Video Board
    bepTickCounter          = BEP 10Hz Tick counter at start of the read
    requestedIndex          = original specified index
    requestedWordCount      = original specified word count
    readIndex               = start address of data in readData[]. For each
                              but the first packet in the series, this
                              should be the readIndex of the previous
                              telemetry packet plus the number of bytes
                              (i.e. words * 2) in the previous packet’s
                              readData[]
    readData[]              = array of read 16-bit words
}
```

Since an odd number of 16-bit words may be requested, except for the last, all readData[] fields in a series of telemetry packets will be filled out to an even number of 16-bit words. The last packet may contain an odd number, although the length of the telemetry packet is always 32-bit aligned. The user determines the length of the last readData[] array by relying on the requestedWordCount.

The Memory Server uses telemetry packet buffers of 2046 16-bit words. The header portion of each “SRAM/PRAM Read Reply” telemetry packet is 11 16-bit words, leaving 2035 16-bit data words for each “SRAM/PRAM Read Reply” telemetry packet. Given a “wordCount” for any region of SRAM/PRAM, the Memory Server will use the following number of telemetry packets to send that region:

```
wordCount / (2046 - 11) (+ 1 if wordCount % (2046 - 11) != 0)
```

Warnings

1. See Section 3.6.1 Warnings

3.6.2 Memory Reads Tagged by Context

ACIS contains certain memory regions containing data structures which are necessary for interpreting the science data. All of this information can be determined by the commanding sequence. However, to ease the burden on the ground system, the ACIS software packages dumps some of these regions using distinct commands and telemetry format tags. The actual form of the telemetry packet data is the same as for the memory reads described in Section 3.6.1.1 .

3.6.2.1 Bad Pixel and Column Maps

Description

ACIS maintains one Bad Pixel Map, a Timed Exposure Bad Column Map and a

Continuous Clocking Bad Column Map (see Section 3.5). The user can telemeter the contents of these maps by issuing “Dump Bad Pixels”, “Dump Bad Te Column Map” and “Dump Bad Cc Column Map” command packets, respectively.

Commands

Dump Bad Pixel Map

```
dumpBadPixels: CMDOP_DUMP_BAD_PIXELS
{
   No additional Parameters
}
```

Dump Timed Exposure Bad Column Map

```
dumpBadTeColumnMap: CMDOP_DUMP_BAD_TE_COL
{
   No additional Parameters
}
```

Dump Continuous Clocking Bad Column Map

```
dumpBadCcColumnMap: CMDOP_DUMP_BAD_CC_COL
{
   No additional Parameters
}
```

Science Telemetry

Command Echo of a generic “Dump” request, where the “echoed command” is substituted for each type of dump.

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   dumpBadPixels: CMDOP_DUMP_BAD_PIXELS
   {
   }
}
```

The constraints and error codes for a dump command are the same as those for a general “Read Bep” command (see Section 3.6.1.1 Science Telemetry), although a result of CMDRESULT_BAD_ARGUMENT indicates a problem within the instrument, rather than a commanding error.

The overall form of the telemetry Bad Pixel Map and Bad Column Map packets are the same as those for a “Read Bep” command (see Section 3.6.1.1 Science Telemetry), except the telemetry format tags are unique for each type of table.

The format of the Bad Pixel Map (formatTag = TTAG_DUMP_BAD_PIXEL) data is an array of 32-bit entries, where each entry has the form:

```
bits 0..3    = I0..S5  - CCD containing the bad pixel
bits 4..13   = 0..1023 - CCD Row Location of the pixel
bits 14..23  = 0..1023 - CCD Column Location of the pixel
```

```
      bits 24..31  = unused
```

The format of the Bad Column Map (formatTag = TTAG_DUMP_BAD_TE_COL or TTAG_DUMP_BAD_CC_COL) data is an array of packed 16-bit words. Although each column is represented using 1 16-bit word, the read will always telemeter data in an array of 32-bit words. If there are an odd number of bad column entries, the last dumped item will contain zeros for all fields.

```
      bits 0..3    = I0..S5  - CCD containing the bad column
      bits 4..13   = 0..1023 - CCD Column Location
      bits 13..15  = unused
```

Warnings

1. See Section 3.6.1.1 Warnings
2. Odd number of columns confusing

   The number of columns is ambiguous when dumping a table containing an odd number of bad columns and when the last entry in the list has the values (CCD = I0 ccdColumn = 0).

3.6.2.2 Parameter Block Slots

Description

ACIS maintains its parameter blocks in sets of arrays of “slots.” Each type of parameter block has a distinct set of slots. Each set contains 5 slots, and each slot is 512 bytes in length (NOTE: All parameter block types are 512 bytes or less in length). In order to verify the contents of the parameters, independently of any science runs, ACIS provides distinct commands to dump the contents of each set of slots using “dumpParameterSlots” command packets, where the command opcode discriminates between the different slot types. Since the number of bytes in an entire set of slots (including overhead) is less in size than a single telemetry packet, each set of slots is telemetered in a single “BEP Read Reply” telemetry packet, where the slot type is indicated using different telemetry format tags:

```
      Timed Exposure Parameter Blocks        - CMDOP_DUMP_TE_SLOTS
                                             - TTAG_DUMP_TE_SLOTS
      Continuous Clocking Parameter Blocks   - CMDOP_DUMP_CC_SLOTS
                                             - TTAG_DUMP_CC_SLOTS
      2-D Window Parameter Blocks            - CMDOP_DUMP_2D_SLOTS
                                             - TTAG_DUMP_2D_SLOTS
      1-D Window Parameter Blocks            - CMDOP_DUMP_1D_SLOTS
                                             - TTAG_DUMP_1D_SLOTS
      DEA Housekeeping Parameter Blocks      - CMDOP_DUMP_DEA_SLOTS
                                             - TTAG_DUMP_DEA_SLOTS
```

NOTE: Be careful not to confuse memory slot dumps with parameter dumps automatically performed at the start of each science run. The purpose and formats of the packets are different (the later contains just the slots used for the run, not the entire set of 5 slots).

Commands

Command to dump all of the stored Timed Exposure Parameter Block slots:

```
dumpParameterSlots: CMDOP_DUMP_TE_SLOTS
{
   No additional parameters required
}
```

Command to dump all of the stored Continuous Clocking Parameter Block slots:

```
dumpParameterSlots: CMDOP_DUMP_CC_SLOTS
{
   No additional parameters required
}
```

The other types of slots are dumped in a similar fashion by substituting the CMDOP_* used to request the dump.

Science Telemetry

Command Echo of a generic “Dump” request, where the opcode of the “echoed command” is substituted for each type of dump.

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   dumpParameterSlots: CMDOP_DUMP_*_SLOTS
   {
   }
}
```

The overall format of the read-reply telemetry packet is identical to that in Section 3.6.1.1 . However, the telemetry format tag will reflect the type of slot set being telemetered. The resulting data array will consist of five 512-byte slots, with a parameter block of the specified type aligned to start at the beginning of each slot. The format of a given parameter block type is identical to the command format used to load that type of block.

Warnings

1. See Section 3.6.1.1 Warnings
2. Each type of parameter block has its own set of slots

   A common misconception about ACIS is that any type of parameter block can go into any slot. THIS IS FALSE. Each type of parameter block has its own distinct set of slots. One cannot store a Continuous Clocking Parameter Block (well, not by accident at least) in a Timed Exposure Parameter Block Slot.
3. Each slot is always 512 bytes in length

The parameter block slots are always 512-bytes in length, although the parameter blocks themselves may be shorter. This leaves gaps between the parameter blocks in the dumped telemetry data filled with random data.

3.6.2.3 Huffman Table

Description

ACIS uses a fixed-table Huffman compression scheme to compress telemetered bias maps and raw CCD images. ACIS maintains an array of selectable compression tables using a data structure. This structure consists of a header, which is an array of 32 table offset values, followed by a data array of 32-bit words. The data array contains the compression tables themselves. The offsets in the header are indexed by a “compression table slot”. Each offset identifies the start of each compression table within the data array, in terms of 32-bit words relative to the end of the header. If an offset is set to 0xffffffff, then there is no compression table associated with that compression table slot.

The software provides a command which dumps the contents of this data structure, “Dump Huffman”. When this command is received, the Memory Server tasks telemeters the raw contents of this entire data structure in a series of “Bep Read Reply” telemetry packets, tagged with the TTAG_DUMP_HUFFMAN format tag.

Commands

Command to dump the Huffman Table data structure

```
dumpHuffman: CMDOP_DUMP_HUFFMAN
{
   No additional parameters required
}
```

Science Telemetry

Command Echo of a “Dump Huffman” request

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   dumpHuffman: CMDOP_DUMP_HUFFMAN
   {
   }
}
```

The overall format of the read-reply telemetry packet is identical to that in Section 3.6.1.1 , except the telemetry format tag will be set to TTAG_DUMP_HUFFMAN. The resulting data array will consist of an array of 32 4-byte offset values, followed by a raw data array of 8197 32-bit data words. The bit-level format of the compression tables themselves is described in Huffman Coding of ACIS Pixel Data (MIT-CSR 36-56102)

.

The number of telemetry packets that will be sent is:

```
      (32 offsets + 8197 data words)/(1023 words/pktbuf-7 header words)
      = 9 packets
```

NOTE: A common error when running the instrument “by-hand” is to forget that the Huffman tables require more than 1 telemetry packet to send, and to issue a dump command for another item before the Huffman Table has been completely sent.

Warnings

1. See Section 3.6.1.1 Warnings
2. Huffman Tables take 9 packets to dump

   A common error when manually commanding ACIS is to not leave enough time for the Huffman Table dump to complete prior to issuing the next dump command. At 24Kbps (format 2), this will take about 13 seconds, and at 500 bps (all other formats), it will take about 10 minutes.

3.6.2.4 Patch List

Description

ACIS maintains a list of “patch nodes” in its I-cache memory, and provides a command to dump the raw contents of this list, using the “Dump Patchlist” command. The resulting series of one or more “BEP Read Reply” telemetry packets is tagged with a telemetry format tag, TTAG_DUMP_PATCHES. Since the list of patch nodes within ACIS is stored first in high-memory, and then grow downward in memory, the format of the dumped list is complex.

Commands

Command to dump the PatchList

```
      dumpPatchlist: CMDOP_DUMP_PATCHLIST
      {
         No additional parameters required
      }
```

Science Telemetry

Command Echo of a “Dump Patchlist” request

```
      commandEcho: TTAG_CMD_ECHO
      {
         arrival   = time command arrived (BEP 10Hz Ticks)
         result    = CMDRESULT_OK (other values indicate an error)

         dumpHuffman: CMDOP_DUMP_PATCHLIST
         {
    }
}
```

The overall format of the read-reply telemetry packet is identical to that in Section 3.6.1.1 , except the telemetry format tag will be set to TTAG_DUMP_PATCHES.

The patch nodes are stored in reverse order in memory, with the first word of the data structure at location 0x800ffffc. The last byte location in the dump will always be:

```
(requestedAddress + requestedWordCount) = 0x800ffffc
```

The easiest way to parse the patch list data is to start from the end of the dumped data, and work backwards towards the beginning. The sent data format, indexed as 32-bit words, is as follows, where RWC is the requestedWordCount field in the “BEP Read Reply” telemetry packets:

```
          etc.
/* END OF NODE 1 */
(RWC - 5 - wordCount0 - 3 - wordCount1 + 1)
   through
(RWC - 5 - wordCount0 - 3):
             data[wordCount1] - Patch Data
(RWC - 5 - wordCount0 - 2):
          wordCount1 - # 32-bit words in Patch Node
(RWC - 5 - wordCount0 - 1):
             dstAddress - Destination Address of Patch
(RWC - 5 - wordCount0):
             patchId - 32-bit Patch Identifier
/* START OF NODE 1 */

/* END OF NODE 0 */
(RWC - 5 - wordCount0 + 1)
   through
(RWC - 5):
             data[wordCount0] - Patch Data
(RWC - 4):   wordCount0 - # 32-bit words in Patch Node
(RWC - 3):   dstAddress - Destination Address of Patch
(RWC - 2):   patchId - 32-bit Patch Identifier
/* START OF NODE 0 */

/* CHECKSUM */
(RWC - 1):   32-bit wide XOR checksum of bolded region
```

If there are no nodes in the list, only the checksum is telemetered.

NOTE: The size of the dump depends on the number of nodes in the patchlist and the size of the data area in each node. Currently, the upper bound is the amount of space remaining in Instruction Cache after the Bad Pixel Map, which is currently TBD bytes. This upper bound may vary, however, if the system is patched to use some of the Bad-Pixel Map space for additional patches. For simplicity, one can always rely on the absolute upper bound, which is the size of Instruction Cache (512Kbytes).

NOTE: The stored on-board pointer to the end of the list (i.e. where to put the next patch node) is not telemetered in the dump. If needed, this field can be read using a “Read BEP” command, using 0x800ffffc as the readAddress, and 1 as the wordCount. The read should be 4-bytes less than the requestedAddress field produced by the dumped patchlist “BEP Read Reply” packets.

```
0x800ffffc: where to write the next patch node
```

Warnings

1. See Section 3.6.1.1 Warnings
2. Size of Patchlist Dump will vary

   The size of a Patchlist Dump depends on the number of patch nodes (3 32-bit words of header per node) and the amount of data stored in each node.
3. Entire dump required to interpret

   Because the patch nodes are stored in RAM from high-memory to low-memory, the node headers follow their respective data arrays, and the data arrays are variable in length, the end of the patchlist dump must be received to interpret nodes preceding it in memory. It is difficult to interpret nodes prior to any telemetry outages or corruptions in the patchlist dump.

3.6.2.5 System Configuration Table

Description

In order to dump the contents of the ACIS System Configuration Table (see Section 3.3), the user issues a “Dump SysConfig” command packet. Upon receipt of the command, the Memory Server task telemeters the table using a single “BEP Read Reply” telemetry packets, tagged using the TTAG_DUMP_SYS_CONFIG telemetry format tag.

Commands

Command to dump the System Configuration Table

```
dumpSysConfig: CMDOP_DUMP_SYS_CONFIG
{
   No additional parameters required
}
```

Science Telemetry

Command Echo of a “Dump SysConfig” request

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   dumpHuffman: CMDOP_DUMP_SYS_CONFIG
   {
            }
        }
```

The overall format of the read-reply telemetry packet is identical to that in Section 3.6.1.1 , except the telemetry format tag will be set to TTAG_DUMP_SYS_CONFIG.

The format of the data portion of the dump consists of an array of 16-bit words. The first two words contain a 32-bit checksum of the table. The remaining words are indexed by the system configuration item identifiers, SYSSET_*, used to modify the table (see Section 3.3).

<u>Warnings</u>

1. See Section 3.6.1.1 <u>Warnings</u>
2. Video Boards are only loaded upon start of science run

   Because video board parameters are loaded into the DEA upon starting a science run, the video board items in the dumped System Configuration table reflect what is is in the table, but not necessarily what is currently loaded into the CCDs video boards.

3.7 Memory Loads

The ACIS software provides three mechanisms for loading code and/or data into the instrument. The first method uses the “Load from Uplink” feature, mentioned in Section 3.1.4 . The second method is to load patches into the instrument, which when the instrument is warm-booted, overwrite sections of code and/or data copied from ROM into RAM. The third method is to use one of the “Write Memory” commands to directly write into the various memory types on the system. The following sections describe these second and third methods.

NOTE: In general, patching and writing to RAM on ACIS can crash the software. The loads for these writes are intended to be built by the ACIS software maintenance team.

3.7.1 Patching the software

ACIS maintains a set of patch nodes in the top of its instruction cache. When the instrument is warm-booted (see Section 3.1.3.3), code and data are copied from ROM into the instrument RAM, and items specified in the patch nodes are then written on top of the copied code and data.

3.7.1.1 Loading the Patches

Description

The user loads patch nodes into the instrument using “Load Patch” commands. Each node specifies a 16-bit identifier (which must not be 0xffff, see Section 3.7.1.3), the destination address to write to when the instrument is warm-booted, and an array of up to 125 32-bit words to write. The instrument determines the number of words in the array from the command packet length. If a patch requires more than 125 words, more than 1 patch node must be used to perform the write.

Commands

Command to add a patch
```
addPatch: CMDOP_ADD_PATCH
{
   patchId      = id of patch- This uniquely identifies the patch
                               node (note: within the instrument no
                               more than 1 patch node can have the
                               same id at a time)
   writeAddress = address     - This specifies where to write the data
                               to upon warm-boot
   writeData[]  = data        - This is the array of 32-bit data words
                               to write
}
```

For example, a command to cause the software version number to be changed to

0x4321 upon the next warm-boot, assuming for this example that the address of the version number variable is 0x8001fdf0, is:

```
addPatch: CMDOP_ADD_PATCH
{
   patchId      = 0
   writeAddress = 0x8001fdf0
   writeData[]  =
   {
      0x4321
   }
}
```

Science Telemetry

Command Echo of a successful “Add Patch” command:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   addPatch: CMDOP_ADD_PATCH
   {
      patchId      = id of patch
      writeAddress = address to patch
      writeData[]  = data to write
   }
}
```

If an error is encountered, such as the patchId already being used by a patch node loaded earlier, the Command Echo “result” field will contain CMDRESULT_BAD_ARGUMENT, and the patch will not be stored in the list.

Warnings

1. Bad patches can crash the instrument software

   Since patches can write anything, anywhere, it is possible that a patch may cause the instrument to crash upon a warm-reboot. If this happens, when the instrument re-boots, it will detect that it was reset due to the watchdog timer and will not reapply the patches to the load copied from ROM (see Section 3.1.3.4).

2. No checks for full patch list

   In order to allow the maintainer to work around unforeseen problems, the instrument software does not check to see if an added patch node overwrites any code or tables in the instruction cache. It is the maintainer’s responsibility to predict where the nodes will go, and to ensure that they do not overwrite any existing tables and/or code.

3. Patches are lost upon power-on boot or cold-boot

   Upon a power-cycle or cold-boot (see Section 3.1.3.1 and Section 3.1.3.2), the on-board patch list is reset to an empty state (NOTE: On a cold-boot, there is a “back-door” recovery strategy. If you know the where the end of the patch list is, and what the checksum value is (see Section 3.6.2.4), you can “Write BEP” the end-of-list pointer and checksum fields to recover the list. This trick won’t work for the power-cycle case, however, because the memory contents will decay once the power is removed from the BEP).

4. Adding a patch may hide a corrupted list

   The checksum for the patch list is only checked when the instrument is re-booted. Whenever a patch node is added to the instrument, the checksum is re-computed and stored. If the patch list is in a corrupt state prior to adding a new patch to the list, the adding of the patch will hide the corruption by re-computing the checksum and storing it.

3.7.1.2 Installing the Patches (Warm Boot)

Description

Once the patch nodes have been loaded into ACIS, they are installed whenever the instrument performs a warn re-boot. The patches are applied in the order in which they were loaded into the instrument. Section 3.1.3.3 describes the overall sequence of steps performed during an warm boot of the instrument, including the resulting command sequence and telemetry indicators

Warnings

1. See Section 3.1.3.3 Warnings
2. Watchdog re-boots will not re-install the patches

   If the instrument watchdog resets, due to a fatal error, or some lockup causing a watchdog time-out, the instrument will perform a Watchdog Reset (see Section 3.1.3.4). This reset will reload code and data from ROM, but will not reapply the patch nodes. The patch nodes themselves, however, will remain in RAM, and will be re-applied upon the next commanded warm re-boot.

3.7.1.3 Removing Patches

Description

ACIS provides two mechanisms to reset the patch list, and one mechanism to remove single patch nodes from the list. To remove one or more patch node from the list, the user issues a “Remove Patches” command, specifying a list of Patch Identifiers to remove. When the instrument receives this command, it locates and removes all nodes which contain a Patch Identifier which matches one of those in the list. The removal process compacts the patch list, releasing memory space at the end (low-memory) of the list.

There are two mechanisms which will remove all of the patches in the list. As mentioned earlier, a power-cycle or cold re-boot of the instrument will empty the contents of the patch list (see Section 3.1.3.1 and Section 3.1.3.2). The user can also reset the entire patch list by issuing a “Remove Patches” command, and specifying 0xffff as the Patch Identifier. This will cause the instrument to reset the contents of the entire patch list.

Commands

To remove a set of patches from the list:

```
removePatches: CMDOP_REMOVE_PATCH
{
   patchIds[] = array of patch ids to remove
}
```

To remove all patches in the list:

```
removePatches: CMDOP_REMOVE_PATCH
{
   patchIds[] = 0xffff    - Reset the entire list
}
```

NOTE: If any of items in the patchIds[] array is 0xffff, the entire list will be removed.

NOTE: To remove the effect of the patches from the code and data in RAM, the BEP must be reset.

Science Telemetry

Command Echo of a “Remove Patches” command

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   addPatch: CMDOP_REMOVE_PATCH
   {
      patchIds[]   = array of patch ids.
   }
}
```

If there an error is encountered, e.g. if the specified patchId is not currently present in the stored patchlist, the Command Echo “result” field will contain CMDRESULT_BAD_ARGUMENT. The instrument, however, will continue to remove the remaining patchIds specified in the command.

Warnings

1. A warm-boot is required to remove the effect of a patch

   Once patches are removed from the patch list, they will no longer be applied when the instrument performs a warm-boot. However, the effect of any patches installed by the last warm-boot will remain in RAM until the BEP is reset.

2. Removing a patch may hide a corrupted list

   The checksum for the patch list is only checked when the instrument is re-booted. Whenever a patch node is removed from the instrument, the checksum is re-computed and stored. If the patch list is in a corrupt state prior to removing a patch from the list, the removal of the patch will hide the corruption by re-computing the checksum and storing it.

3.7.1.4 Dumping the Patch List

See Section 3.6.2.4

3.7.2 Writing to BEP Memory

Description

The patches described in the preceding section will only overwrite BEP memory after a successful warm boot. By contrast, ACIS also provides for instantaneous memory writes to the BEP using the Memory Server task. The ACIS maintainer can write to any location in the BEP address space using a “Write BEP” command packet. The command packet specifies which location to write to, and an array of up to 125 32-bit data words to write. The instrument determines the number of data words using the command packet length. The destination address must be aligned to a 32-bit word location (i.e. evenly divisible by 4).

Upon receipt of the command, the Memory Server task will copy the supplied data words to the specified location.

NOTE: If data is written to storage used by the instrument for code or data, the next reset of the instrument will overwrite the written area with the code and/or data from ROM.

The constraints for memory writes are the same as those for memory reads:

- The start address must be aligned to a 32-bit word boundary (i.e. evenly divisible by 4)

  If not, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo “result” field.
- The region being read must not cross an Instruction Cache address boundary

  These addresses are: 0x80080000 and 0x800fffff. If the region crosses one of these boundaries, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo “result” field.
- If the region is in a FEP shared memory area, the associated FEP must be powered on

  The FEP shared memory regions are:

```
        FEP 0  - 0xa8000000 0xa8ffffff
        FEP 1  - 0xa9000000 0xa9ffffff
        FEP 2  - 0xaa000000 0xaaffffff
```

```
FEP 3  - 0xab000000 0xabffffff
FEP 4  - 0xac000000 0xacffffff
FEP 5  - 0xad000000 0xadffffff
```

If the FEP does not have power, a BEP Bus Error Exception will be raised and will cause a Fatal Error and subsequent BEP re-boot.

Commands

Command to write a block of BEP memory

```
writeBep: CMDOP_WRITE_BEP
{
   writeAddress = start   - First location to write to
   writeData[]  =         - array of data words to write
}
```

For example, a command to cause the software version number to be immediately changed to 0x4321, assuming for this example that the address of the version number variable is 0x8001fdf0, is:

```
writeBep: CMDOP_WRITE_BEP
{
   writeAddress = 0x8001fdf0
   writeData[]  =
   {
      0x4321
   }
}
```

Science Telemetry

Command Echo of a successful “Write BEP” command:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   writeBep: CMDOP_WRITE_BEP
   {
      writeAddress = address to write
      writeData[]  = data to write
   }
}
```

If an error is encountered, such as specifying an invalid address, the Command Echo “result” field will contain CMDRESULT_BAD_ARGUMENT, and the memory will not be over-written.

If the Memory Server task is in the process of executing an earlier write or execute command, the Command Echo “result” field will contain CMDRESULT_BUSY, and the new data will not be written.

If the Memory Server task is in the process of performing an earlier memory read or dump, the Command Echo “result” field will contain

CMDRESULT_CLOBBERED, the read operation will be aborted, and the new data will be written.

Warnings

1. Writes of the wrong data to the wrong locations can crash the software

   It is assumed that the maintainers know what they're writing and why.

2. BEP hardware is memory mapped

   Write BEP commands can write to BEP hardware registers.

3. Aborts earlier read

   If, when the command is received, the Memory Server task is executing an earlier read command, such as a dump of the Huffman Tables, the earlier read will be aborted, possibly causing incomplete information to be telemetered. The write will then commence.

4. Written data may be lost upon next re-boot

   Unless the writes are being used to modify the patch list, any data written to areas overwritten by the ROM load, or initialized and used by the running software, will be lost upon the next re-boot of the instrument.

5. Fatal Errors due to writes to un-powered FEP shared memory

   Attempts to write to shared memory regions belonging to FEPs which do not have power will result in a BEP Bus Error exception, Fatal Error Message and BEP re-boot.

3.7.3 Writing to FEP Memory

Description

The ACIS maintainer initiates memory writes to a FEP using a "Write FEP" command packet, which specifies which FEP to write to, the starting location to write, and an array of up to 125 32-bit words to write. The instrument determines the number of words to write using the command packet length.

ACIS implements user-initiated memory writes to the FEP using the Memory Server task. If the write is to the FEP's shared memory, the writes are made directly by the BEP into that memory. If, however, the locations are not directly accessible by the BEP (e.g. within the FEP's I-cache or D-cache), the BEP will collaborate with the software running on the FEP to perform the write.

The constraints for memory writes are the same as those for memory reads:

- The start address must be aligned to a 32-bit word boundary (i.e. evenly divisible by 4)

  If not, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo "result" field.

- The region being written to must not cross an Instruction Cache address boundary

These addresses are: 0x80080000 and 0x800fffff. If the region crosses one of these boundaries, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo "result" field.

- The FEP must be powered when the write command is received

  Attempts to write to a FEP when the BEP knows that it is not powered will cause the write to be rejected. The Command Echo result field will be set to CMDRESULT_BOARD_OFF.

- The FEP must remain powered during the write

  Attempts to write to a FEP's memory when the FEP is not powered may cause the write to abort, or possibly cause a BEP Bus Error Exception, which will subsequently cause a Fatal Error and BEP re-boot.

<u>Commands</u>

Command to write a block of FEP memory

```
writeFep: CMDOP_WRITE_FEP
{
   fepId        = FEP_0..FEP_5   - Specifies which FEP to write to
   writeAddress = start          - First location to write to
   writeData[]  =                - array of data words to write
}
```

<u>Science Telemetry</u>

Command Echo of a successful "Write FEP" command:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   writeFep: CMDOP_WRITE_FEP
   {
      fepId        = FEP to write to
      writeAddress = address to write
      writeData[]  = data to write
   }
}
```

If an error is encountered, such as specifying an invalid address, the Command Echo "result" field will contain CMDRESULT_BAD_ARGUMENT, and the patch will not be stored in the list.

If the Memory Server task is in the process of executing an earlier write or execute command, the Command Echo "result" field will contain CMDRESULT_BUSY, and the new data will not be written.

If the Memory Server task is in the process of performing an earlier memory read or dump, the Command Echo "result" field will contain CMDRESULT_CLOBBERED, the read operation will be aborted, and the new data will be written.

If the BEP knows that the FEP is not powered at the time of the request, the Command Echo “result” result field will contain CMDRESULT_BOARD_OFF and the new data will not be written.

Warnings

1. Writes of the wrong data to the wrong locations can crash the FEP software

   It is assumed that the maintainers know what they’re writing and why.

2. FEP hardware is memory mapped

   Write FEP commands can write to FEP hardware registers.

3. Aborts earlier read

   If, when the command is received, the Memory Server task is executing an earlier read command, such as a dump of the Huffman Tables, the earlier read will be aborted, possibly causing incomplete information to be telemetered. The write will then commence.

4. Written data may be lost upon next re-boot of the FEP

   Any data written to areas overwritten by the FEP program load, or initialized and used by the running software, will be lost upon the next re-boot of the FEP.

5. Fatal Errors due to writes to un-powered FEP shared memory

   Attempts to write to shared memory regions belonging to FEPs which do not have power may result in a BEP Bus Error exception, Fatal Error Message and BEP re-boot.

3.7.4 Writing to DEA Sequencer Memory

Description

Although it is not recommended, the maintainer can initiate writes to DEA video board SRAM and PRAM using “Write SRAM” and “Write PRAM” commands, respectively, specifying the index of the CCD associated with that video board (see Section 3.3.4), the starting SRAM or PRAM index to write to, and a number of 16-bit data words to write. The ACIS software determines the number of words to write using the command packet length. The following constraints apply:

- The selected Video Board must be powered

  Attempts to write to a Video Board when the board is not powered will cause the write to be rejected. The Command Echo result field will be set to CMDRESULT_BOARD_OFF (see Warnings concerning DEA power)

- The sequencers must be idle

If a science run is in progress when a write is requested, or starts while a write request is active, the write will be aborted. NOTE: It is STRONGLY recommended that one does NOT perform a direct SRAM or PRAM write during the setup stage of a science run, since the Science task may also be loading SRAM and PRAM (NOTE: once the sequencers are running, the hardware prevents reads and writes to the sequencer RAM). There is little risk to the hardware, however, since the ACIS on-board Sequence Checker will verify that contents of the SRAM and PRAM cannot hurt the hardware.

Commands

Command to write a block of SRAM memory

```
writeSram: CMDOP_WRITE_SRAM
{
   ccdId        = I0..S5         - Specifies the CCD Video Board
   writeIndex   = start          - First SRAM location to write to
   writeData[]  =                - array of 16-bit data words to write
}
```

Command to write a block of PRAM memory

```
writePram: CMDOP_WRITE_PRAM
{
   ccdId        = I0..S5         - Specifies the CCD Video Board
   writeIndex   = start          - First PRAM location to write to
   writeData[]  =                - array of 16-bit data words to write
}
```

Science Telemetry

Command Echo of a generic SRAM write request:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   writeSram: CMDOP_WRITE_SRAM
   {
      ccdId        = specified Video Board
      writeIndex   = specified start index
      writeData[]  = array of 16-bit words to write
   }
}
```

Command Echo of a generic PRAM read request:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   readPram: CMDOP_WRITE_PRAM
   {
      ccdId        = specified Video Board
      writeIndex   = specified start index
```

```
            writeData[]  = array of 16-bit words to write
        }
    }
```

If the write interrupts an earlier on-going read, the Command Echo “result” field will contain CMDRESULT_CLOBBERED.

If the write attempts to interrupt a Write-Memory or Execute-Memory activity, the read is not performed. In this case, the Command Echo “result” field will contain CMDRESULT_BUSY.

If an attempt is made to write to a Video Board whose power is off, the command will be rejected and the Command Echo “result” field will contain CMDRESULT_BOARD_OFF.

Warnings

1. Don’t use this to load science sequencer code

   The ACIS science software is designed to re-load the SRAM and PRAM for each video board at the start of each science run. If a literal load of SRAM or PRAM is required for a run, the science parameter blocks provide a mechanism to reference the content of these loads (see ACIS IP&CL Software Structure Definitions, MIT-CSR 36-53204.0204, the deaLoadOverride field of the loadTeBlock and loadCcBlock definitions).

2. Aborts earlier read

   If, when the command is received, the Memory Server task is executing an earlier read command, such as a dump of the Huffman Tables, the earlier read will be aborted, possibly causing incomplete information to be telemetered. The write will then commence.

3. Written data may be lost upon the setup for the next science run

   During the setup stage for each science run, the ACIS software resets all of the video boards and re-loads their SRAM and PRAM with the code to use for the run. This may overwrite areas written to by earlier Write SRAM and Write PRAM commands.

4. Written data may be lost upon next power-cycle of the video board

   Any data written into the video boards will be lost when power is removed from the board.

5. DEA power must remain on

   The ACIS software recognizes that a video board has power if it successfully sends the command to power the board. If the DEA is off when the command is issued, the resulting error will prevent the software from flagging the video board as powered. If, however, the DEA is powered off after successfully powering on video boards, the ACIS software continue to think that the video boards have power. Under these conditions, ACIS will accept and acknowledge the request to write to the video board with a CMDRESULT_OK, but will subsequently encounter an error when attempts to execute the write. This error will be indicated in Software Housekeeping as a SWSTAT_DEABOARD_ERROR report.

3.8 Calling Subroutines

ACIS provides mechanisms for the maintainer to directly call C, C++ or assembler subroutines in both the BEP and the FEPs.

NOTE: In general, calling code directly within ACIS can crash the software. These types of commands are intended to be built by the ACIS software maintenance team.

3.8.1 Calling BEP subroutines

Description

ACIS implements commanded function calls on the BEP using the Memory Server task. The maintainer prepares an “Execute BEP” command packet, specifying the address of the function to call, and a list of zero to up to 20, 32-bit arguments to pass to the called subroutine.

Upon receipt of the command, the Memory Server task calls the subroutine indirectly via a pointer, always passing 20 arguments. The values specified in the command packet will appear in order at the beginning of the argument list, and the remaining passed arguments will contain unpredictable values. In C and C++, one can always pass more arguments to a subroutine than the subroutine demands (NOTE: This may or may not be allowed for certain assembly language subroutines. If needed, the maintainer can load a translation subroutine using “Write BEP” (see Section 3.7.2), which deals with the extra arguments and invokes the target routine).

The called subroutine runs under the Memory Server task, which will be blocked until the called routine returns. Attempts to execute subsequent Memory Server commands, such as another execute command or a read or write command, while the called subroutine has control, will be rejected with a CMDRESULT_BUSY reply in their respective Command Echoes.

Once the called subroutine returns, the Memory Server task records the return value in a “BEP Execute Reply” telemetry packet. For the C and C++ compilers used for the Mongoose, this value is always in the same CPU register. If the function does not return any value, the recorded reply value will be meaningless. The maintainer should be aware of what the return value, if any, means, and therefore should know how to deal with it.

The constraints for execute memory commands are as follows:

- The function address must be aligned to a 32-bit word boundary (i.e. evenly divisible by 4)

  If not, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo “result” field.

- The function being executed cannot be located in the BEP's Data Cache

  These addresses are: 0x80000000 and 0x8007ffff. If the specified function address is in this region, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo "result" field.

- If the function is in a FEP shared memory area, the associated FEP must be powered on (NOTE: Executing BEP code which is stored in FEP shared-memory has not yet been shown to work)

  The FEP shared memory regions are:

```
FEP 0  - 0xa8000000 0xa8ffffff
FEP 1  - 0xa9000000 0xa9ffffff
FEP 2  - 0xaa000000 0xaaffffff
FEP 3  - 0xab000000 0xabffffff
FEP 4  - 0xac000000 0xacffffff
FEP 5  - 0xad000000 0xadffffff
```

  If the FEP does not have power, a BEP Bus Error Exception will be raised and will cause a Fatal Error and subsequent BEP re-boot.

<u>Commands</u>

Command to execute a routine on a BEP:

```
executeBep: CMDOP_EXEC_BEP
{
   functionAddress     = address of subroutine
   functionArguments[] =      up to 20 arguments to pass to routine
}
```

For example, the maintainer has a diagnostic program to run on FEP_2, and wants to load it using the FepManager's "loadRunProgram" member function, located at 0x8008939c. The first argument to the loadRunProgram is the address of the fep-Manager object (in C++, the first argument of all non-static member functions is a pointer to the object), which is 0x80003da4. The second argument is the FEP Id, which, for FEP_2, has a value of 2, and the third is a pointer to the FepProgram to load. Assume that the program has already been loaded into RAM using a series "Write BEP" commands, at location 0x80030000. The command to call the sub-routine is:

```
executeBep: CMDOP_EXEC_BEP
{
   functionAddress  = 0x8008939c - &FepManager::loadRunProgram
   functionArguments[] =
   {
      0x80003da4,   - &fepManager
      2,            - FEP_2
      0x80030000    - address of pre-loaded FEP load image
   }
}
```

<u>Science Telemetry</u>

The command echo from an "Execute BEP" command is:

```
commandEcho: TTAG_CMD_ECHO
```

```
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   executeBep: CMDOP_EXEC_BEP
   {
      functionAddress
      functionArguments[]
   }
}
```

If an error is encountered, such as specifying an invalid address, the Command Echo "result" field will contain CMDRESULT_BAD_ARGUMENT, and the function will not be called.

If the Memory Server task is in the process of executing an earlier write or execute command, the Command Echo "result" field will contain CMDRESULT_BUSY, and the function will not be called.

If the Memory Server task is in the process of performing an earlier memory read or dump, the Command Echo "result" field will contain CMDRESULT_CLOBBERED, the read operation will be aborted, and the function will be called.

When the called function returns, the Memory Server task will produce a "BEP Execute Reply" telemetry packet:

```
bepExecuteReply: TTAG_EXEC_BEP
{
   commandId        = id of command containing call
   bepTickCounter   = BEP 10Hz tick counter at the time of the call
   functionAddress  = address of the function which was called
   returnedValue    = 32-bit value returned by the called func-
                      tion.(NOTE: This field is meaningless if the
                      function is "void")
}
```

Warnings

1. Calls to the wrong locations or with the wrong arguments can crash the software

   It is assumed that the maintainers know what they are calling and why

2. Aborts earlier read

   If, when the command is received, the Memory Server task is executing an earlier read command, such as a dump of the Huffman Tables, the earlier read will be aborted, possibly causing incomplete information to be telemetered. The execute will then commence.

3. Routines must return within 7 minutes or manage task monitor requests

   Within the ACIS software, there is a Task Monitor task which issues an event to each task in sequence. The target task must then respond to the Task Monitor. Once the targeted task responds, the Task Monitor "touches" the hardware watchdog timer, sleeps, for about 1 second, and then moves on to the next task. If a task does not respond within about 8 minutes, the hardware watchdog timer will expire and reset the BEP. Since subroutines invoked by the "Execute BEP" command run under the Memory Server task, they must either return to the calling Memory Server code within about 7 minutes, so the Memory Server code can respond to the Task Monitor, or must themselves respond to the Task Monitor queries. If not, the called routine will starve the watchdog timer, and cause the BEP to reset.

3.8.2 Call FEP subroutines

Description

ACIS implements commanded function calls on the FEP using the Memory Server task, in cooperation with software already running on the targeted FEP. The maintainer prepares an "Execute FEP" command packet, specifying which FEP to use, the address in FEP memory space of the function to call, and a list of zero to up to 20, 32-bit arguments to pass to the called subroutine.

Upon receipt of the command, the Memory Server task calls a routine which forms and issues a request to the FEP software, via the BEP to FEP mailbox, and waits for a reply. The FEP software then calls the subroutine indirectly via a pointer, always passing 20 arguments. The values specified in the command packet will appear in order at the head of the argument list, and the remaining passed arguments will contain unpredictable values. In C and C++, one can always pass more arguments to a subroutine than the subroutine demands (NOTE: This may or may not be allowed for certain assembly language subroutines. If needed, the maintainer can load a translation subroutine using "Write FEP" (see Section 3.7.2), which deals with the extra arguments and invokes the target routine).

The dispatch of the request to the FEP runs under the Memory Server task, which will be blocked until the FEP replies to the request, or until the BEP times out waiting for the reply. The called routine on the FEP must return within 10 seconds to avoid a time-out (NOTE: The FEP routine will continue to run after the time-out has expired. The BEP will reset and re-load the FEP if it remains non-responsive during the setup stage of the next science run). Attempts to execute subsequent Memory Server commands, such as another execute command or a read or write command, while the called subroutine has control, will be rejected with a CMDRESULT_BUSY reply in their respective Command Echoes.

Once the called subroutine returns on the FEP, the FEP software packages the returned value in its reply to the BEP. The Memory Server task records the returned value in a "FEP Execute Reply" telemetry packet. For the C compilers used for the Mongoose, this value is always in the same CPU register. If the function does not return any value, the recorded reply value will be meaningless. The maintainer should be aware of what the return value, if any, means, and therefore should know how to deal with it.

The constraints for execute FEP memory commands are as follows:

- The start address must be aligned to a 32-bit word boundary (i.e. evenly divisible by 4)

  If not, the command will be rejected, with a CMDRESULT_ARGUMENT in the Command Echo “result” field.

- The FEP must be powered when the execute command is received

  Attempts to call a routine on a FEP when the BEP knows that the FEP is not powered will cause the execute to be rejected. The Command Echo result field will be set to CMDRESULT_BOARD_OFF.

- The FEP must remain powered during the call

  Attempts to call a FEP routine when the FEP is not powered may cause the request to abort, or possibly cause a BEP Bus Error Exception, which will subsequently cause a Fatal Error and BEP re-boot.

<u>Commands</u>

Command to execute a routine on a FEP:

```
executeFep: CMDOP_EXEC_FEP
{
   fepId              = FEP_0..FEP_5   - FEP to use
   functionAddress    = address        - address of subroutine
   functionArguments[] =               - up to 20 arguments to pass
                                         to routine
}
```

<u>Science Telemetry</u>

The command echo from an “Execute FEP” command is:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   executeFep: CMDOP_EXEC_FEP
   {
      fepId
      functionAddress
      functionArguments[]
   }
}
```

If an error is encountered, such as specifying an invalid address, the Command Echo “result” field will contain CMDRESULT_BAD_ARGUMENT, and the function will not be called.

If the Memory Server task is in the process of executing an earlier write or execute command, the Command Echo “result” field will contain CMDRESULT_BUSY, and the function will not be called.

If the Memory Server task is in the process of performing an earlier memory read or dump, the Command Echo “result” field will contain CMDRESULT_CLOBBERED, the read operation will be aborted, and the function will be called.

When the called function returns, the Memory Server task will produce a “FEP Execute Reply” telemetry packet:

```
fepExecuteReply: TTAG_EXEC_FEP
{
   commandId        = id of command containing call
   fepId            = id of FEP used
   bepTickCounter   = BEP 10Hz tick counter at the time of the call
   functionAddress  = address of the function which was called
   returnedValue    = 32-bit value returned by the called func-
                      tion.(NOTE: This field is meaningless if the
                      function is “void”)
}
```

NOTE: If the called subroutine does not return within 10 seconds, the BEP will not issue the “FEP Execute Reply” telemetry packet. The FEP routine may continue to execute until the BEP attempts to setup for a subsequent science run. If the FEP remains non-responsive at this point, the BEP will reset and re-load the FEP.

Warnings

1. Calls to the wrong locations or with the wrong arguments can crash the FEP

   It is assumed that the maintainers know what they are calling and why

2. Aborts earlier read

   If, when the command is received, the Memory Server task is executing an earlier read command, such as a dump of the Huffman Tables, the earlier read will be aborted, possibly causing incomplete information to be telemetered. The execute will then commence.

3. FEP routines must return within 7 minutes or manage the FEP watchdog timer

   Software running on the FEP is jointly responsible for maintaining the watchdog timer. This timer will expire and reset the FEP if it is not reset at least once every 8 minutes. All called subroutines on the FEP must either return within 7 minutes, or must touch the watchdog timer themselves.

4. FEP routines must return within 10 seconds to avoid reply time-out on the BEP

   The BEP maintains a 10 second time-out when waiting for replies from the FEP. If the subroutine on the FEP does not return within 10 seconds, the BEP will not issue a “FEP Execute Reply” telemetry packet.

3.9 Radiation Monitor Triggers

Description

It is expected that ACIS will be subject to high radiation conditions under two scenarios. The first is when the spacecraft passes through the Van Allen belts as part of its orbit. The second is radiation due to solar flares.

Although the ACIS hardware is fairly robust with respect to radiation, repeated exposure to high levels of radiation will gradually degrade the hardware. The effect of the damage to the hardware can be reduced by powering off sensitive subsystems during exposures to high radiation levels. The most sensitive parts of ACIS, with respect to radiation damage, are the CCDs, the Analog-to-Digital Converters, and the Digital-to-Analog Converters; both of the latter are on the DEA video boards.

During each orbit, the spacecraft, and therefore ACIS, will pass through radiation belts. The location and levels of radiation in these belts will be fairly deterministic, and can be factored into the operations scheduling system. To reduce the degradation of the CCDs due to routine exposure to the belts, the video boards to the CCDs should be powered off during belt passage. Since knowledge of when the spacecraft will enter and exit the belts can be predicted by ground control, the ground should not schedule observations during belt passage, and should issue commands to power-down the video boards when entering the belts, and power them up again once the belts have been traversed. Refer to Section 5.7 for a description of ACIS operations during radiation belt passage.

The spacecraft contains a science instrument, known as an Electron, Proton, Helium Instrument (EPHIN), which monitors the levels of different types of radiation. If, during a solar flare or other unanticipated period of high-radiation, an ACIS-specific level is exceeded, the spacecraft invokes an on-board command sequence, which will issue a hardware serial digital command to ACIS. This command will cause the DPA hardware to set a radiation monitor flag. The ACIS System Configuration task polls this flag once per second. If the flag is asserted, the ACIS software suspends the current science observation, and issues commands to power off all of the video boards (takes about 10 seconds). Once the radiation level subsides, the spacecraft will invoke another “all-clear” command sequence, which issues another hardware serial digital command. This command clears the DPA radiation flag. The System Configuration task will then restore power to the video boards , and re-start the suspended science run, or start a new science run, depending on the ACIS commands that may have been received while the monitor was asserted.

Commands

Radiation Alert Hardware Serial Digital Command Mnemonics

```
1RMONIRM (v=1)   - Assert Radiation Monitor Flag
1RMONIRM (v=0)   - De-assert Radiation Monitor Flag
```

Engineering Telemetry

As the science run is stopped, and as the video boards are powered off, the electric current levels will drop. See Appendix A for a table of current levels under different instrument configurations.

Science Telemetry

When the radiation monitor flag is asserted, the instrument reports the following to Software Housekeeping:

```
SWSTAT_SCI_INHIBIT_ON   - Inhibited science runs
SWSTAT_DEACCD_POWEROFF - Power off to the DEA video boards
```

Once the radiation subsides and the monitor flag is deasserted, ACIS reports the following housekeeping:

```
SWSTAT_SCI_INHIBIT_OFF - Science runs are allowed again
SWSTAT_DEACCD_POWERON  - Re-powering the video boards (if necessary)
```

If a science run was in progress when the radiation monitor is asserted, the run will be aborted. The resulting “Science Report” telemetry packet (see Section 4.1) will report a termination code of SMTERM_RADMON, indicating that the run was terminated due to the radiation monitor.

If a command is issued to start a science run while the monitor flag is asserted, the start will be deferred until the flag is de-asserted. The Command echo will indicate this in the “result” code with a CMDRESULT_INHIBITED.

If a command is issued to stop a science run while the monitor flag is asserted, the current run will not resume once the flag is de-asserted. The command echo will supply a “result” code of CMDRESULT_INHIBITED.

Warnings

1. Bias will always be re-computed on next science run

   Because the radiation may corrupt the contents of the FEP bias maps, these maps are always re-computed at the start of the resumed run, or next science run. This may affect the start-up time of an observation.

2. Thermal conditions may change

   When the video boards are autonomously powered off, the DEA will get a little colder.

3. DEA Housekeeping will report 0xffff for video boards

   If a DEA Housekeeping run is in progress which queries one or more of the video boards, the corresponding entries will report 0xffff while the boards are off.

4. Desired science run state is “remembered”

ACIS remembers the most recent science run command when the radiation flag is asserted. Once it is deasserted, ACIS attempts to place the instrument into the science configuration that was desired for that moment in the observatory timeline.

For example, if ACIS is performing science run 1, and the monitor goes off, 1 is aborted. Once the monitor subsides, science run 1 is re-started, with a forced recomputation of its bias map. If, however, a stop command followed by a start run 2 command is received while the monitor is active, science run 2 will be started when the monitor flag is de-asserted.

4.0 Instrument Science Activities

This section describes the various activities used to perform routine science operations using the ACIS instrument software.

4.1 Performing a Science Run

The overall process of performing a science run within ACIS is as described in the ACIS Software Requirements Specification (SRS), MIT 36-01103. This section covers some of the implementation details not provided in the SRS, and recapitulates the overall process. (Although, a science run consists of the following overall steps, there are a variety of variants and details which are not addressed by this template, e.g. if and when DEA Housekeeping runs are started):

- Power on DEA video boards and FEPs required for the run
- Load a parameter block
- Start the run
- Wait for the desired observation time
- Stop the run

Within ACIS, there are basically two types of science runs: Timed Exposure Science Runs, and Continuous Clocking Science Runs. In Timed Exposure runs, the CCDs are statically integrated for some period of time, and then the resulting image is "captured" and processed (NOTE: The "capture" step introduces a "smear" integration cycle). In order to avoid overloading the ACIS DEA Power Supply, the "capture" step for each CCD is staggered in time. In Continuous Clocking runs, the CCD rows are continuously shifted, and are only "statically" integrated during the pause while a row is clocked out to the DEA/DPA to be processed. There is no "staggering" in Continuous Clocking mode.

4.1.1 Estimating Minimum Integration Times

4.1.1.1 Continuous Clocking Mode

<u>Description</u>

In Continuous Clocking (CC) mode, the user cannot explicitly set the integration time of each row. Instead, the user can only control the time using the side-effects of the summing parameters, the output register mode, and the number of overclocks parameter within the CC Mode Parameter Block.

The rate at which summed rows are clocked in CC mode is given by:

```
seconds_per_row= (row_transfers * pixel_clocks_per_transfer) +
                 underclocks + pixels + dummy_overclocks + overclocks +
                 end_of_row_marker) / pixel_clocks_per_second
```

```
                    = (2^rowSum * 4 pixel_clocks_per_transfer) + (4 underclocks
                      + n)/m pixels + 4 + (overclockPairsPerNode * 2) + 1
                      end_of_row_marker) / 100,000 pixel_clocks_per_second

      where: "n" is 256 if outputRegisterMode in FULL or DIAG mode and
             "n" is 512 if the outputRegisterMode is AC or BD mode.
      and:   "m" is 1 if columnSum == 0, and
             "m" is 2 if columnSum != 0.
```

The “static” integration time of a summed row (i.e. the time spent integrating the row while the CCDs were not moving the charge within the row) is the row rate, minus the pixel transfer time: (summed rows * 4 pixel clocks/transfer).

Parameters

The parameters which govern the row rate are contained in the Continuous Clocking Parameter Block, defined by the “loadCcBlock” command packet definition:

```
loadCcBlock: CMDOP_LOAD_CC
{
    ...
    rowSum    = Row Summation Factor  - summed_rows = 2^rowSum rows
    columnSum = Column Summing Factor - column_summing = FALSE if
                                        columnSum == 0, otherwise TRUE
    overclockPairsPerNode
              = # of overclock pairs  - number_of_overclocks =
                                        overclockPairsPerNode * 2
    outputRegisterMode
              = OR Clocking Mode      - output_register_mode = FULL,
                                        DIAG, AC or BD...
}
```

4.1.1.2 Timed Exposure Mode

Description

In Timed Exposure (TE) mode, the user explicitly specifies up to two static integration times, with a duty cycle parameter governing the ratio between the two times, within the TE Mode Parameter Block. Normally, the integration of one exposure overlaps, in time, the transfer of the previous exposure to the DEA/DPA for processing, which usually takes about 3.3 seconds. If, however, the integration time is less than this transfer time, TE mode does not overlap the two, and flushes the charge out of the CCDs prior to each short integration. This achieves the shorter integration time (i.e. less than 3.3 seconds), but reduces the effective exposure time by the time taken to transfer images to the DEA/DPA.

Due to the “staggered” image capture, the minimum integration time that can be specified is governed by the number of CCDs used for the run. The minimum integration time that can be specified, with flushes, is as follows:

```
1 CCD           - 0.1 second
2 CCDs          - 0.1 second
3 CCDs          - 0.2 seconds
4 CCDs          - 0.2 seconds
```

```
5 CCDs          - 0.3 seconds
6 CCDs          - 0.3 seconds
```

In order to avoid the flushes due to a short integration time, the integration time must be set to be greater than or equal to the time taken to capture (smear time) and transfer an image from the CCD to the DEA/DPA (transfer time). This time depends on a number of clocking parameters:

- Number of configured CCDs
- Subarray Start Row
- Number of Subarray Rows
- On-chip Summing Flag
- Output Register Mode
- Number of Overclock Pixels

The minimum non-short integration time that can be specified is as follows:

```
min_time  =  staggered_smear_time + transfer_time
          =  ([(1026 rows * 4 pixel_clocks_per_parallel_transfer) *
                 (“k” CCDs - 1)] +
              [(subarrayStartRow + 2 overhang_rows) * “k” CCDs *
                   4 pixel_clocks_per_parallel_transfer] +
              (2 flushes * [(4 underclocks + “n” pixels) / “m”]) +
              [(subarrayRowCount + 1)/”m” *
                 (4 pixel_clocks_per_parallel_transfer +
                 [(4 underclocks + “n” pixels) / “m”] +
                 4 dummy_clocks +
                 (overclockPairsPerNode * 2) +
                 1 pixel_clock_per_end_of_row_marker)]
             ) * 1x10^-5seconds_per_pixel_clock
where:“k” is the number of entries in the fepCcdSelect array which are
       not set to CCD_DESELECT. Since the “smear” time of the current
       CCD is not counted against its “static” integration time, the
       expression computes the staggered “smear” time of the remaining
       CCDs.
and:   “n” is 256 when the outputRegisterMode is FULL or DIAG mode and
       “n” is 512 when the outputRegisterModeis in AC or BD mode.
and:   “m” is 1 if onChip2x2Summing is 0 and
       “m” is 2 if onChip2x2Summing is not 0
```

For example, for a configuration with all 6 CCDs, no on-chip summing, a full image of 1024 rows, the output register mode in FULL mode, and 30 overclock pixels per node, the minimum non-short integration time that can be specified is 3.27224 seconds. Since the time must be expressed in 1/10th second increments, this becomes 3.3 seconds.

For another example, consider a configuration with a single CCD, 2x2 on-chip sum, a subarray with 100 rows, starting at row 0, with the output registers in FULL mode, and only 2 overclock pixels per node, the time would be 0.14368 seconds, becoming 0.2 seconds.

Parameters

The parameters which govern the minimum integration times are contained in the Timed Exposure Parameter Block, defined by the "loadTeBlock" command packet definition:

```
loadTeBlock: CMDOP_LOAD_TE
{
    ...
    fepCcdSelect[6] =   FEP Indexed array of CCD Ids
                                    - the number of configured CCDs
                                      is the number of entries in this
                                      array which are not set to
                                      CCD_DESELECT
    onChip2x2Summing = 2x2 sum flag   - if 0, do not perform on-chip
                                      sumation. If not 0, perform 2x2
                                      on-chip summation
    subarrayStartRow = 1st row of subarray
                                    - start_row = subarrayStartRow
    subarrayRowCount = # rows         - # rows in subarray =subarray-
                                      RowCount + 1
    overclockPairsPerNode
            = # of overclock pairs  - overclock_pixels =
                                      overclockPairsPerNode * 2
    outputRegisterMode
            = OR Clocking Mode      - outputRegisterMode = FULL,
                                      DIAG, AC or BD
    ...
}
```

4.1.2 Estimating Setup Time

Description

Upon receipt of one of the "Start Science Run" commands, the Science Manager task starts setting up for the run. This setup time varies, depending on the various parameters contained in the parameter block and the state of hardware and software in the system. This section describes the setup process, the approximate time for each step, and a summary of the parameters which affect this time. This section does not describe the time taken to perform a bias-recomputation, and the time taken to telemeter the bias maps. For bias computation and telemetry estimates, see Section 4.2.1 and Section 4.2.2 .

The overall steps performed by the Science Manager task are as follows, with time estimates of each step. For simplicity, time estimates of each step are approximated to the second or minute, as appropriate:

- Copy the parameter blocks

  In order to avoid parameter block integrity problems, the Science Manager makes a copy, at the start of a science run, of the parameter block that it is about to use. The time taken to perform this step is very small, and always less than 1 second.

- Dump the parameter blocks to telemetry

Once the parameter blocks have been staged, they are copied into telemetry packets and posted for transfer. The copy takes much less than 1 second, and the time to telemeter the blocks also takes less than 1 second.

- Setup data processing

  Once the parameter blocks have been posted, the Science Manager sets up its internal data structures and variables to process the data for the current mode. This setup stage takes less than 1 second.

- Reset Video Boards

  In accordance with recommendations from the DEA analog designer, the Science Manager task resets the DEA video boards prior to the start of each run. The reset action takes much less than 1 second.

- Load Video Board Registers and DACs from System Configuration Table

  After the video boards have been reset, the Science Manager task loads the register values and Digital-to-Analog converter values into each video board. To ensure that the video boards are communicating properly, each write is followed by a read and compare. On the video boards, different register actions require different time-delays to take effect prior to issuing another command. In order to be robust to changes in the video board timing requirements, the ACIS software always inserts a 1 second time-delay after each video board register write. Although there are only 4 control registers for each video board, some of the control bits are “edge-driven”. This implies that some actions require writing a 0 to a control register bit, followed by writing a ‘1’, or visa-versa. As a result, the setup requires a total of 5 control register writes for each video board (not including starting the sequencers). The time to actually perform each write and each read after a write is no more than 100μs each, leading to a worst-case time which is on the order of 10ms for all registers on all 10 video boards. This is small compared to the 1 second delay added after each write. The time taken to load and read-back the control registers on all of the configured boards:

  ```
  video board register load time= (5 * 1 second) * configured CCDs
  ```

  There are 23 video board Digital-to-Analog Converter registers on each video board. Because the science run parameter blocks duplicate the video ADC offset DAC values, ACIS performs 27 DAC writes to the video boards. Since the DAC values are write-only, there are no read-backs. Each write takes no longer than 100μs. It takes less than 0.5 seconds to load the DACs into all 10 video boards. Using 1 second for all of the I/O times to and from the video boards for the register loads and DAC loads, leads to:

  ```
  video board register and DAC load time  =  (5 * 1 second) * configured
                                             CCDs + 1 second
  ```

- Load SRAM Library into Video Boards

  Each video board contains 64K words of Sequencer RAM, although the current SRAM library uses only 3648 words. Once the video board registers have been loaded, the Science Manager task loads the SRAM library into each configured video board. Each word of the library is written and then read-back to ensure that the board is communicating properly. Each write and read-back takes on the order of 100μs each. The worst case time to load all of SRAM in each board is:

```
worst case load time per board=  65535 * 200µs * configured CCDs
                              =  13.107 seconds/board
```

The current load time per board, given the delivered size of the SRAM library, is much shorter:

```
current load time per board   =  3648 * 200µs * configured CCDs
                              =  0.7296 seconds/board
```

For example, the load of the current SRAM library into 6 CCDs takes about 4.3776 seconds.

- Generate and Load PRAM into Video Boards

  The Science Manager task then constructs a sequencer Program RAM (PRAM) load which implements a clocking sequence with the configured properties. The video boards each contain 64K words of PRAM, so the worst-case transmission of the PRAM load is on the order of that for SRAM, about 14 seconds/board. In practice, however, most PRAM loads are much smaller. The size and complexity of a PRAM load depends non-linearly on the clocking parameters chosen for the run. In general, single integration times, with no on chip summation lead to the smallest PRAM loads, on the order of 100 words. Some larger PRAM loads are on the order of 1600 words. Assuming the larger load size, the load time per board is:

```
example PRAM load time per board =  1600 * 200µs * configured CCDs
                                 =  0.32 seconds/board
```

  For example, a load of 1600 words of PRAM into 6 CCDs takes a total of about 1.92 seconds

- Re-load and re-run FEP code if necessary

  The Science Manager task queries each configured FEP as to the state of its bias map. If a FEP does not respond, the Science Manager resets and re-loads software into the offending FEP, and starts the loaded code. Under most circumstances, the FEPs do not required resets. If, however, a radiation monitor alert, or aborted science run (aborts occur if a science run is already active when a subsequent science run request is received) resets the FEPs, all of the configured FEPs would be re-loaded. Each FEP takes approximately 10 seconds to reset and re-load (see Section 3.3.2). It takes about 60-70 seconds to re-start all 6 FEPs.

- Load parameters into FEP

  Once the FEPs are running and responding, the Science Manager loads the parameters into the FEPs. This operation takes less than 1 second to configure all 6 FEPs.

- Acquire and telemeter standard DEA housekeeping

  The Science Manager queries all of the housekeeping channels of all of the configured DEA video boards and telemeters the information in a DEA Housekeeping telemetry packet. Each analog housekeeping channel takes 0.5 seconds to read (see Section 3.4.6 Warnings), and there is a total of 20 analog channels to read on each board. Reading the video control registers from all the boards takes well under 1/10th second, and the time to pack the data into the telemetry packet takes under 1/2 second, so the total time to acquire, pack and post the DEA video board housekeeping is:

```
video housekeeping = (0.5 seconds * 20) * configured boards + 1
```

- Check the safety of the DEA sequencer loads

  To ensure that the loaded DEA sequence cannot overheat and subsequently damage the video board driver circuits, the Science Manager task checks the contents of the video board's PRAM and SRAM loads. The speed of the checking algorithm depends heavily on the nature of the loads. In most cases, the algorithm's buffering allows the PRAM and SRAM to be read back once. In some cases, however, the algorithm may run out of buffer space and read back a given section of PRAM or SRAM many times while checking the load.

  During testing, the time taken by the checking algorithm on a large, complex load, has never exceeded 2 seconds.

- Flush charge from the CCDs (Jitter DACs)

  In order to flush electrical charge built up in the CCDs while they were idle since the previous science run, the Science Manager task must adjust the video board DAC levels, and clock the CCDs for at least 1 exposure cycle. To simplify the design, the Science Manager always clocks the CCDs for 11 seconds, which is greater than the longest frame time.

4.1.3 Estimating Telemetry Utilization

Description

ACIS manages telemetry buffering using fixed numbers of fixed sized buffers for each type of application, where a telemetry packet must fit into a single buffer, and a single buffer never contains more than 1 packet. ACIS manages the average allocation of telemetry bandwidth by controlling the number and size of buffers allocated for each purpose. The effect is that, although at one moment, the peak allocation may be dominated by one particular application, on the average, each application can utilize its maximum allocation if it needs to, minus inefficiencies due to unused space in the fixed size buffers. If, however, at certain moments, an application does not require its full telemetry allocation, the unused bandwidth can be used by an application requiring more bandwidth. Although the use of fixed size buffers simplified the ACIS design and improved its reliability, it makes the job of estimating telemetry utilization more difficult.

For example, the science data processing applications are allocated 400 buffers of 512 32-bit words each, whereas memory dumps are allocated only 4 buffers of 1023 words each. Assuming that nothing else is being telemetered, that the current science mode always has data to send, filling 70% of each telemetry buffer it uses, and that a dump is in progress and is using 100% of each memory dump buffer, the science task can buffer (ignoring FEP ring-buffers and image buffers for the moment) ((0.7 * 512) * 400) 32-bit words in the ACIS telemetry buffers. The memory server can buffer 4092 words. If the telemetry stream is saturated, and the buffers fill, the science task will consume about 97.3% of the telemetry bandwidth, and the memory server will consume 2.7%.

NOTE: In prior software releases, science had been allocated 200 buffers of 1023 32-bit words. However, most science modes did not usually fill these buffers, and this led to inefficient use of the science buffer allocation.

Buffer Allocations

Table 14 lists the current buffer sizes and allocations for each type of telemetry packet (NOTE: These do not include the ring buffers on each FEP which can buffer up to 8K events each). Note that in some cases, the buffer size exceeds the fixed packet size. This allows patching of the output format without having to modify the buffer size allocation.

TABLE 14. Telemetry Buffer Allocations

Buffer Pool	Buffer Count	Buffer Size (bytes)	Telemetry Packet Types	Packet Size (bytes)	Frequency
Bias Thief	20	4092	dataCcBiasMap	1564	If configured to compute and send the map, “n” per science run, where “n” is the number of configured CCDs
			dataTeBiasMap	44 + (4 * “n” compressed words)	Bad or no compression yields 1 1024-pixel bias row per packet. Excellent compression yields about 8 rows per packet. The system sends 1 map per configured CCD.
DEA House.	8	1024	deaHousekeepingData	20 + (4 * “n” queries)	1 every “sampleRate” seconds
Command Echo	4	2048	commandEcho	520	1 per command
Fatal	1	256	fatalMessage	20	1 per software crash
Memory	1	4092	bepReadReply	28 + (4 * “n” words)	vary depending on read, dump, and execute commands
			fepReadReply	28 + (4 * “n” words)	
			pramReadReply	24 + (2 * “n” entries)	
			sramReadReply	24 + (2 * “n” entries)	
			bepExecuteReply	24	
			fepExecuteReply	24	

TABLE 14. Telemetry Buffer Allocations

Buffer Pool	Buffer Count	Buffer Size (bytes)	Telemetry Packet Types	Packet Size (bytes)	Frequency
Science	400	2048	dumpedTeBlock	<= 820	1 per TE Science Run
			dumpedCcBlock	<= 724	1 per CC Science Run
			dataTeFaint	12 + (16 * “n” events)	zero or more per exposure, depending on event rate
			dataTeGraded	12 + (7.25 * “n” events)	
			dataTeFaintBias	12 + (29.5 * “n” events)	
			dataTeVeryFaint	12 + (40 * “n” events)	
			dataCcFaint	12 + (6.875 * “n” events)	
			dataCcGraded	12 + (4.25 * “n” events)	
			dataTeHist	16 + (4 * “n” bins)	In FULL or DIAG mode, ~ 33 packets per “histogramCount” exposures. In AC or BD mode, ~ 17 packets.
			dataTeRaw	24 + (4 * “n” compressed words)	Bad or no compression yields 1 row per packet. Excellent compression yields about 8 rows per packet.
			dataCcRaw		
			dataBiasError	20 + (4 * “n” bias errors)	Zero or more per exposure, depending on corruption rate of bias map
			exposureCcRaw	72	“n” per exposure, where “n” is the number of configured CCDs
			exposureCcFaint	36	
			exposureTeFaint	72	
			exposureTeFaintBias	80	
			exposureTeHistogram	52	
			exposureTeRaw	36	
			scienceReport	48	1 per science run
Startup	1	4092	bepStartupMessage	28	1 per BEP startup
SwHouse	8	3088	swHousekeeping	16 + (12 * “n” reports)	1 every ~64 seconds

4.1.4 Loading Parameter and Window Blocks

ACIS conatins a total of 25 slots to hold various types of control blocks - Te, Cc, 1d 2d and Dea. There are 5 slots reserved for each type of block. The following sections describe these parameter blocks and how they are managed.

4.1.4.1 Timed Exposure Parameter Blocks

<u>Description</u>

Slots

ACIS has five numbered slots in which it stores Timed Exposure Parameter blocks. The parameter blocks are loaded into a particular slot using a “Load TE Block” command packet, where the slot number (0 - 4) is specified in the teBlockSlotIndex field. When parameter block is loaded into a slot, the slot’s contents are overwritten by the new block contained in the command. If the instrument is power-cycled or cold-booted (see Section 3.1.3.1 , Section 3.1.3.2), the contents of the slots will be restored to their “as-launched” values. A warm-boot (see Section 3.1.3.3), however, retains the contents of the most recently loaded slots.

Checksum

Each parameter block contains a checksum field. The checksum is the bit-wise XOR of each 16-bit word in the command packet following the checksum field (i.e. starting with the lowest address 16-bit word of the parameterBlockId field).

Parameter Block Id

The Parameter Block Identifier is a 32-bit value which can be used by the ground to uniquely identify the parameter block. ACIS reports the value of the Parameter Block Id in all exposure record, bias map, bias error packets and science report packets which used that parameter block for a bias computation and/or science run. In some cases, the Parameter Block used for science run data processing may be different than the parameter block used to compute the bias map used for the run, in which case, the telemetered Bias Parameter Block Id will be different than the main data processing Parameter Block Id (these are distinct fields in the telemetry packets).

NOTE: The intent of the Parameter Block Id is to aid bookkeeping on the ground. ACIS does not use the id internally for any particular purpose, and therefore, the users are free to use the field as they wish.

CCD Selection

The Timed Exposure Parameter Block contains an array of 6 CCD identifiers which are indexed by FEP Id. The user selects which CCDs to use for the science run by assigning the CCD Id to the particular FEP. A FEP can be designated as unused by storing the CCD_DESELECT value in the CCD Id field. Data from a particular CCD may be processed by more than one FEP, but subsequent parameters may conflict (see below). In general, if more than one FEP is processing data from the same CCD, the FEP with the larger FEP Id index will overwrite the parameters used by the lower (NOTE: This is a side-effect. ACIS applies all of the parameters from all of the FEPs, in ascending FEP Id order. The last FEP wins). The affected parameters are the CCD Video Response parameters and the FEP Video Offset parameters. For example, if one configured:

```
      fepCcdSelect[FEP_0]          = CCD_I0
      fepCcdSelect[FEP_1]          = CCD_I1
      fepCcdSelect[FEP_2]          = CCD_I0
      fepCcdSelect[FEP_3 .. FEP_5]  = CCD_DESELECT
   and
      ccdVideoResponse[FEP_0]      = 0
      ccdVideoResponse[FEP_1]      = 0
      ccdVideoResponse[FEP_2]      = 1
   and
      fep0VideoOffset[FEP_0]       = [60,60,60,60]
      fep0VideoOffset[FEP_1]       = [60,60,60,60]
      fep0VideoOffset[FEP_2]       = [30,30,30,30]
```

then CCD_I0 would use the low-gain video response (Video Response = 1) and 30 for the video ADC offsets.

On the other hand, the CCD images are captured in the order they first appear in the FEP CCD selection list, so in the above example, CCD_I0 would perform its image-to-frame captures first, followed by CCD_I1.

NOTE: Since the image transfer from the CCD framestores to the DEA/DPA is performed synchronously, images captured earlier spend more time in the CCD framestore. The time taken for each capture is approximately 40ms, so images from the CCD used by FEP_0 would spend 40ms longer in the framestore than images from the CCD used by FEP_1. The user can control this by selecting which CCDs are placed in the low order FEPs. For example, if a particular CCD performs better if its images are retained longer in the framestore, then the user would probably assign it to FEP_0.

If a CCD’s video board is not powered, or if its assigned FEP is not powered, or if an error is encountered while setting up for the run, the offending FEP or CCD will not be used for the run. This will not, however, affect the image capture sequence. Because the time spent in the framestore is indirectly configured by the user (i.e. the user may be counting on this behavior for some reason), ACIS attempts to preserve the specified behavior by keeping the relative time delays the same. Unfortunately, this is not true for the video response and video offset parameters. If a FEP is not powered or has encountered an error, the most recent successful FEP’s parameters will be used. The user can avoid this problem by ensuring that the video response and offset parameters are the same for all FEPs processing the same CCD during a run.

Processing Modes

The user selects the ACIS processing via the fepMode and bepPackingMode parameters of the loadTeBlock command. The following indicates how the modes are selected, and what other parameters are affected (note that the ignore initial frames, clocking parameters, video offsets and special parameters are always enabled):

TABLE 15. Timed Exposure Processing Mode Selection

Mode	fepMode	BEP Packing Mode	Telemetry Packets	Enabled Parameters
Raw	Raw	ignored	1. (1) dumpedTeBlock 2. (1) deaHousekeepingData 3a. (1..n) dataTeRaw 3b. (1..n) exposureTeRaw 4. scienceReport	rawCompressionSlotIndex windowSlotIndex
Histogram	Histogram	ignored	1. (1) dumpedTeBlock 2. (1) deaHousekeepingData 3a. (17 or 33) dataTeHist 3b. (1..n) exposureTeHistogram 4. (1) scienceReport	histogramCount

TABLE 15. Timed Exposure Processing Mode Selection

Mode	fepMode	BEP Packing Mode	Telemetry Packets	Enabled Parameters
Faint 3x3	3x3	Faint	1. (1) dumpedTeBlock 2. (1) deaHousekeepingData 3. (0..n) dataTeBiasMap 4a. (0..n) dataTeFaint 4b. (1..n) exposureTeFaint 5. (1) scienceReport	all bias parameters fepEventThreshold fepSplitThreshold lowerEventAmplitude eventAmplitudeRange gradeSelections windowSlotIndex
Faint with Bias 3x3	3x3	FaintBias	1. (1) dumpedTeBlock 2. (1) deaHousekeepingData 3. (0..n) dataTeBiasMap 4a. (0..n) dataTeFaintBias 4b. (1..n) exposureTeFaintBias 5. (1) scienceReport	
Graded	3x3	Graded	1. (1) dumpedTeBlock 2. (1) deaHousekeepingData 3. (0..n) dataTeBiasMap 4a. (0..n) dataTeGraded 4b. (1..n) exposureTeFaint 5. (1) scienceReport	
Faint 5x5	5x5	Faint	1. (1) dumpedTeBlock 2. (1) deaHousekeepingData 3. (0..n) dataTeBiasMap 4a. (0..n) dataTeVeryFaint 4b. (1..n) exposureTeFaint 5. (1) scienceReport	

Clocking Parameters

TBD - Reference to copy of pramte.doc description of Timed Exposure Clocking

The Timed Exposure Parameter Block contains a set of parameters which controls how the CCDs are clocked, and the format of the resulting images (NOTE: some combinations of these parameters are not supported. Refer to TBD). The clocking parameters are as follows.

- On-Chip Summing Flag

If this flag is 0, then the system will not perform any charge summing on the CCD. If the flag is set to 1, then the charge from each pair of CCD rows and columns are summed on the CCD. This halves the number of rows and columns clocked out for each image, and affects the time taken to transfer the image to the DEA/DPA for processing (see Section 4.1.1).

NOTE: Because this parameter affects the image geometry, if this parameter is different than that used for last bias computation, the user must ensure that a new bias map is computed with the changed parameter.

- Subarray Start Row

  This specifies the first CCD row to transfer to the DEA/DPA for processing. If 0, then the first row used for imaging (there are two dummy rows prior to the first image row) is sent. If not zero, the image is down shifted in the framestore until the specified row is reached, and the resulting image is transferred to the instrument. This parameter, coupled with the on-chip summing flag and the subarray row count, affects the resulting image geometry. This parameter also affects the time taken to transfer the image to the DEA/DPA (see Section 4.1.1). In order to avoid overheating the DEA driver circuit, this parameter must be less than 924 (i.e. for each image, at least 100 rows must be clocked from the CCD to the DEA) (NOTE: This is checked by the instrument software, and then effectively re-checked by the DEA sequence load checker. The ACIS software ensures that a bad assignment or corruption of this parameter cannot damage the hardware).

  NOTE: Because this parameter affects the image geometry, if this parameter is different than that used for last bias computation, the user must ensure that a new bias map is computed with the changed parameter.

- Subarray Row Count

  This specifies the number of (possibly summed) rows to clock out of the CCD minus 1. This parameter, coupled with the on-chip summing flag and the subarray row count, affects the image geometry. This parameter also affects the time taken to transfer to image to the DEA/DPA (see Section 4.1.1). In order to avoid overheating the DEA driver circuit, the subarray row count must be at least 99 (i.e. 100 rows) (NOTE: This is checked by the instrument software, and is then effectively re-checked by the DEA sequence load checker. The ACIS software ensures that a bad assignment or corruption of this parameter cannot damage the hardware).

  NOTE: Because this parameter affects the image geometry, if this parameter is different than that used for last bias computation, the user must ensure that a new bias map is computed with the changed parameter.

- Overclock Pairs Per Node

  This specifies the number of pairs of overclock pixels to clock out of each CCD serial register output node. For example, if a value of 15 is specified, when the Output Register Mode is in FULL mode, 30 overclocks per node per row will be produced, or a grand total of 120 overclock pixels per row.

In raw mode, this parameter affects the telemetered image geometry, and is not affected by the windowing parameters. Although, the entire image may be clipped out with a window, the overclocks will be telemetered.

This parameter affects the time taken to transfer images to the DEA/DPA for further processing (see Section 4.1.1).

NOTE: Because this parameter may affect baseline overclock levels, it is recommended that the bias map be recomputed when this parameter changes.

- Output Register Mode

  This parameter specifies the configuration of the CCD output nodes. If in FULL mode, the images are clocked out of all four CCD output nodes, A, B, C, and D. If in DIAG mode, the images are clocked in a reverse direction, away from the four output nodes. This is used to measure clocking noise in the DEA electronics by clocking the CCDs, but not moving any charge into the electronics. From the instrument software's point of view, the resulting image appears as if it were clocked in FULL mode. AC and BD modes are intended for use if one of the video chains should fail. AC mode clocks the serial registers through the A and C nodes. In BD mode, only the B and D nodes are used. Since only half the video chains are used, these modes take twice as long as FULL or DIAG mode (see Section 4.1.1).

  NOTE: Because this parameter changes the relationship between CCD pixel and output node, and therefore, the baseline overclock levels, it is recommended that the bias maps be re-computed whenever this parameter is changed.

- Video Responses

  These parameters control the gain of the measured signal by changing the A/D integration timing on the video boards. If 0, then normal gain is used. If set to 1, then the corresponding CCD signal gain is cut by about 75%. Currently, it is not expected that this feature will ever be used. This parameter only affects the signal quality, and does not affect the image geometry or transfer time to the DEA/DPA. This parameter may require adjustments to the FEP Video Offset parameters, and the various threshold and bias parameters.

  NOTE: Because these parameters dramatically affects the signal gains, the bias-maps should be recomputed whenever these parameters change.

- Primary and Secondary Integration Times and Duty Cycle

  These parameters control the static integration times of the CCD images, and the ratios of integration times used. For example, if the primary integration time is 0.3 seconds, and the secondary time is 3.3 seconds, and the duty cycle is 10, the CCD will first integrate for 0.3 seconds and transfer the image, and then perform a series of 10 3.3 second integrations. It will then repeat the cycle. Refer to Section 4.1.1 for a description of the minimum integration times, and the effect of short integration times. The effect of these parameters are coupled to the time taken to transfer images to the DEA/DPA.

  If the dutyCycle is non-zero, only images integrated using the secondary exposure times are used when computing the bias maps.

  NOTE: Because the integration times may affect the CCD bias levels, it is recommended that the bias maps be re-computed whenever these parameters are changed.

Video ADC Offsets

Because the clocking parameters can affect the signal levels measured on the video boards, the Timed Exposure Parameter block specifies the Analog-to-Digital Converter (ADC) video offset levels, which override those stored in the System Configuration Table (see Section 3.3.4). The video offset parameters act inversely to the video chain signal level (i.e. larger values produce a lower offset). The conversion formula for these values is specified in TBD.

NOTE: Because the video offset parameters may affect the signal measurement levels and affect the CCD bias levels, it is recommended that the bias maps be recomputed whenever these parameters are changed, although, if the offsets are adjusted to compensate for changes in the other parameters in order to maintain a consistent bias level, one may not require a bias re-computation.

Bias (and related) Parameters

ACIS re-computes the bias maps whenever one or more of the following conditions are met:

1. A science run is started and the recomputeBias flag is set in its parameter block
2. A Bias-Only science run is started (see Section 4.2)
3. A science run is started in an event processing mode was requested and either

- The FEP's bias offsets were corrupt, or
- The previous Science Run was not a Timed Exposure run, or
- The radiation monitor had been asserted since the last bias computation

NOTE: Many of the CCD clocking parameters can affect the bias levels of the CCD images, and require re-computation of the map. The instrument software does NOT automatically detect these changes. It is the user's responsibility to determine when the bias maps should be re-calculated, and request computation of the bias maps when needed.

The ignoreInitialFrames parameter indicates how many exposures the FEPs are to throw away prior to computing the bias map. This is to allow for the bias-levels of the CCDs to settle. (NOTE: For data processing, 2 exposures are always discarded, hence, all reported exposure numbers start at 2).

ACIS supports a variety of bias computation algorithms, each requiring a collection of parameters. The algorithm selection and parameters are supplied independently for each FEP in the biasAlgorithmId and biasArg parameters. The details of the algorithms and the effect of the parameters are provided in TBD.

If the ignoreBadPixelMap is 0, ACIS installs the bad pixel map contained in the Bad Pixel Map into the bias maps whenever the maps are re-computed. If the ignoreBadPixelMap is 1, the pixels are not flagged. The same applies to the ignoreBadColumnMap, where the bad columns are retrieved from the Timed Exposure Bad Column Map (see Section 3.5).

If a bias is re-computed, and the trickleBias flag set to a 1, then the bias maps from each used FEP are telemetered prior to the start of data processing. If the trickleBias flag is 0, then the maps are not telemetered. If the maps are trickled, the biasCompressionSlotIndex indicates which compression table to use to pack the data. If the field is set to 255, then the maps will be bit-packed, with no compression. Refer to TBD for a description of the current Huffman compression tables.

NOTE: The maps are sent only when they are re-computed. If trickleBias is 1, but the map wasn't computed as part of the run, the map will not be telemetered at that time. In order to maintain a stable bias-level, the sequencers will continue to clock the CCDs while the maps are telemetered. The FEPs will ignore the clocked data.

Event Selection Parameters

The event selection parameters are only used when the ACIS is configured for Faint 3x3, FaintBias, Graded, or Faint 5x5 modes. The details of the event selection algorithm are provided in TBD. These parameters are:

- FEP Pixel Thresholds

  The FEP pixel thresholds, fepEventThreshold, are indexed by FEP and output node. For each FEP, the BEP loads the four output node thresholds into the corresponding Front End Processor. The user supplies these values in bias-corrected Analog-to-Digital unit (ADU) values. The FEP hardware uses the thresholds to locate pixels, in incoming CCD images, whose measured pulse heights are large enough to indicate possible detected X-ray events. The FEP software and hardware effectively adjust these values to correct for drifts in the average overclock levels from the DEA video chains, and for pixel-to-pixel bias differences, using the computed pixel bias maps.

- BEP Split Thresholds

  The BEP split thresholds, fepSplitThreshold, are indexed by FEP and output node. The Back End software uses these values to determine which pixels, surrounding the center of a candidate X-ray event, contain charge from the event. The values are specified in bias-corrected ADUs. The BEP use this to compute an estimate of the total amplitude of the X-ray event, and its grade code. The amplitude and grade code are then used by the BEP for subsequent event filtering, and if in Graded Mode, are telemetered.

- Event Amplitude Range

  The lowerEventAmplitude and eventAmplitudeRange parameters specify the range of estimated amplitudes to accept from candidate events. If a candidate event's estimated amplitude is less than the lowerEventAmplitude or is greater than or equal to (lower-

EventAmplitude + eventAmplitudeRange), the event is rejected and not sent to the ground. If the event is within the range, then its is passed on to the grade and possibly window selection filters.

- Event Grade Selection

  The gradeSelections parameter is an array of 256 bits, where each bit corresponds to 1 grade code. If the user sets a bit to 1, then events whose grade code matches the corresponding entry may be telemetered. If a bit is 0, then such events are rejected, and are not telemetered. If an event matches a grade which is enabled, then the event is passed to the window filter.

  TBD - Reference grade code description

- Window Processing

  If the windowSlotIndex is 0xffff, then no window processing is performed on the candidate events, otherwise, the 2-D Window Parameter Block stored in the corresponding slot is used to process the candidate events. See Section 4.1.4.2 for a description of this processing.

Raw Image Processing

If the run is performing Raw Mode, then the following parameters are used to filter and telemeter the data:

- Window Processing

  If the windowSlotIndex is 0xffff, then no window processing is performed on the raw image, otherwise, the 2-D Window Parameter Block stored in the corresponding slot is used to process the image. See Section 4.1.4.2 for a description of this processing.

  NOTE: Window processing is not applied to the overclock data. All overclocks from all of the rows in the image are telemetered.

- Compression table

  The rawCompressionSlotIndex indicates which compression table to use to pack the raw data. If the field is set to 255, then the maps will be bit-packed, with no compression. Refer to TBD for a description of the current Huffman compression tables

Special Cases

The DEA Load Override parameter is provided to allow the maintainer to specify an explicit PRAM/SRAM image to load into the sequencers.This parameter causes the software to by-pass the SRAM library, and the on-board PRAM builder, and will ignore all clocking parameters when building the load. It is the user’s responsibility to place the load image somewhere in the BEPs data memory, and ensure that the clocking parameters in the parameter block are consistent with the images produced by the explicit load.

NOTE: The DEA sequence checker will still verify that the loaded PRAM/SRAM does not threaten the health of the DEA electronics.

The FEP Load Override parameter can be used by the maintainer to provide alternate software to run in the FEP hardware. It is the user's responsibility to ensure that the FEP load image is stored somewhere in the BEP's data memory, and that the loaded program is compatible with the BEP-FEP protocols expected by the BEP software.

Commands

Summary of Load Timed Exposure Parameter Block (see TBD for an example):

```
loadTeBlock: CMDOP_LOAD_TE
{
   teBlockSlotIndex    = 0..5    - Specify which slot to load
   checksum            = checksum- 16-bit wide XOR of remainder of
                                   packet (illustrated in bold-type)
   parameterBlockId    = id      - Ground selected Parameter Block Id
   fepCcdSelect[6]     = CCD_I0..CCD_S5, CCD_DESELECT
                                 - Array indexed by FEP Id of which
                                   CCD each FEP should use
   fepMode             = Raw, Histogram, 3x3 or 5x5
   bepPackingMode      = Faint 3x3, FaintBias, Graded, Faint 5x5
   onChip2x2Summing    = 0..1    - Select 2x2 on-chip summing
   ignoreBadPixelMap   = 0..1    - Don't load bad pixels after bias
   ignoreBadColumnMap  = 0..1    - Don't load bad columns after bias
   recomputeBias       = 0..1    - Force re-computation of bias
   trickleBias         = 0..1    - Send bias maps after computation
   subarrayStartRow    = 0..923  - First row of subarray
   subarrayRowcount    = 100..1023- Number of rows in subarray
   overclockPairsPerNode= 0..15  - Number of overclock pairs
   outputRegisterMode  = FULL, DIAG, AC, BD
                                 - Output register configuration
   ccdVideoResponse[6] = 0..1    - Normal or low-gain, indexed by FEP
   primaryExposure     = 0.1..10s- Primary static integration time
   secondaryExposure   = 0.1..10s- Secondary static integration time
   dutyCycle           = 0..15   - Ratio of Secondary to Primary times
   fep*EventThreshold[4]= -4096..4095
                                 - FEP threshold values for each FEP.
                                   Indexed by output node.
   fep*SplitThreshold[4]= 0..4095- BEP threshold values for data from
                                   each FEP. Indexed by output node.
   lowerEventAmplitude = 0..4095 - Minimum event amplitude
   eventAmplitudeRange = 0..65535- Maximum amplitude above minimum
   gradeSelections[256]= 0..1    - Select accepted event grade codes
   windowSlotIndex     = 0..4,255- Select 2-D window slot
   histogramCount      = 0..65535- Number of exposures in histogram
   biasCompressionSlotIndex[6] = 0..255- Compression table for bias
                                   maps, indexed by FEP
   rawCompressionSlotIndex= 0..255- Compression table for raw images
   ignoreInitialFrames= 0..65535 - Exposures to drop prior to bias
   biasAlgorithmId[6]  = 0..255  - Bias Algorithm to use, indexed by
                                   FEP
   biasArg*[6]         =  value  - Bias Arguments, indexed by FEP
   fep*VideoOffset[4]  = 0..255  - ADC Video Offsets for each FEP,
                                   indexed by output node.
   deaLoadOverride     = 0, pointer- If not 0, BEP Memory pointer to
                                   DEA load image
   fepLoadOverride     = 0, pointer- If not 0, BEP Memory pointer to
                                   FEP load image
}
```

Science Telemetry

Command Echo on a successful load:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   loadTeBlock: CMDOP_LOAD_TE
   {
      teBlockSlotIndex    = 0..5
      checksum            = checksum
      parameterBlockId    = id
      ...
   }
}
```

If the checksum of the packet doesn't match its contents, the block will not be loaded into the slot, and the Command echo will report the error by setting the result field to CMDRESULT_STORE_ERROR.

NOTE: If the parameter block contains fields which are out of range, or which are illegal in combination with other fields, an attempt to start a science run will abort, with the terminationReason in the Science Report set to SMTERM_PROC_PARM_INVALID, SMTERM_DEA_PARM_INVALID, SMTERM_FEP_PARM_INVALID. However, if the user specifies a set of clocking parameters which are legal, but not supported by the on-board SRAM library, it will appear as DEA error, and the terminationReason will be set to SMTERM_DEA_IO_ERROR.

Warnings

1. Video Offsets are overrides

   The video offset parameters in the parameter block override those specified in the System Configuration Table (see Section 3.3.4). The video offset parameters in the System Configuration Table are never used.

2. Use System Configuration to control board power

   The science parameter blocks DO NOT apply or remove power from hardware boards. If a video board or FEP is not powered, it will not be used for the run. If none of the requested boards are powered, the run will be aborted.

3. Some clocking combinations are not supported

   Certain combinations of clocking parameters cause the software to access SRAM library components which don't exist, and will abort the run. See TBD.

4.1.4.2 2-D Window Parameter Blocks

Description

Slots

ACIS has five slots in which it stores 2-D Window Parameter blocks. The parameter blocks are loaded into a particular slot using a “Load 2D Block” command packet, where the slot is specified in the windowSlotIndex field. When parameter block is loaded into a slot, the slot’s contents are overwritten by the new block contained in the command. If the instrument is power-cycled or cold-booted (see Section 3.1.3.1 , Section 3.1.3.2), the contents of the slots will be restored to their “as-launched” values. A warm-boot (see Section 3.1.3.3), however, retains the contents of the most recently loaded blocks.

Checksum

Each parameter block contains a checksum field. The checksum is the bit-wise XOR of each 16-bit word in the command packet following the checksum field (i.e. starting with the least-significant 16-bit word of the windowBlockId field).

Window Block Id

The Window Block Identifier is a 32-bit value which can be used by the ground to uniquely identify the parameter block. ACIS sends the value of the Window Block Id in all exposure record packets and science report packets which used the windows for a science run.

NOTE: The intent of the Window Block Id is to aid bookkeeping on the ground. ACIS does not use the id internally for any particular purpose, and therefore, the users are free to use the field as they wish.

Window Definitions

The remainder of the window parameter block contains a series of zero or more window definitions. The number of windows in the block is determined using the command packet length. Each window definition specifies the CCD to which the window applies, the row and column position of the bottom left corner of the window, and the width and height of the window, all in CCD coordinates. Each window also specifies an event amplitude range, which is only used for event processing, and a sample field, which is used for both event processing, and for raw-mode data clipping. Refer to TBD for an explicit description of window processing of event data, and TBD for a description of window processing of raw mode data. ACIS is capable of supporting up to 6 windows for each CCD used for a run. The parameter block can hold up to 49 window definitions, and therefore, is capable of specifying 6 windows for 6 CCDs used for a run. The parameter block may contain windows for CCDs not used in the run, but does not have enough space, however, to specify all 6 windows for all 10 CCDs.

Commands

Summary of Load 2DWindow Block (see TBD for an example):

```
load2dBlock: CMDOP_LOAD_2D
{
   windowSlotIndex     = 0..5    - Specify which slot to load
   checksum            = checksum- 16-bit wide XOR of remainder of
                                   packet (illustrated in bold-type)
   windowBlockId       = id      - Ground selected Window Block Id
   windows[]           =         - Array of window definitions
   {
      ccdId            = I0..S5  - Identify CCD
      ccdRow           = 0..1023 - Bottom row of window
      ccdColumn        = 0..1023 - Left most column of window
      width            = 0..1023 - Width of window - 1
      height           = 0..1023 - Height of window - 1
      sampleCycle      = 0..255  - Event sample count, or raw pixel
                                   selection
      lowerEventAmplitude = 0..4095 - Minimum event amplitude
      eventAmplitudeRange = 0..65535- Maximum amplitude above minimum
   }
   {
      ccdId ...
   }
   ...
}
```

Science Telemetry

Command Echo on a successful load:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   load2dBlock: CMDOP_LOAD_2D
   {
      windowSlotIndex     = 0..5
      checksum            = checksum
      windowBlockId       = id
      windows[]           =
      {
         ccdId            = I0..S5
         ccdRow           = 0..1023
         ccdColumn        = 0..1023
         width            = 0..1023
         height           = 0..1023
         sampleCycle      = 0..255
         lowerEventAmplitude = 0..4095
         eventAmplitudeRange = 0..65535
      }
      {
```

```
                ccdId ...
            }
        ...
        }
    }
```

If the checksum of the packet doesn't match its contents, the block will not be loaded into the slot, and the Command echo will report the error by setting the result field to CMDRESULT_STORE_ERROR.

NOTE: If the parameter block contains fields which are out of range, or which are illegal in combination with other fields, an attempt to start a science run will abort, with the terminationReason in the Science Report set to SMTERM_PROC_PARM_INVALID.

Warnings

1. First window in the list hides subsequent windows

   Contrary to how bit-mapped graphics are produced, and paste-up sheets are overlaid on a table, the windows in the parameter block overlap in a queue-like fashion, where the last window in the parameter block is the bottom most window.

2. Cannot specify 6 windows for all 10 CCDs in 1 block

   The parameter block can only hold up to 49 windows.

3. Row and Column 0 and 1023 never produce events

   In order to handle the event detection boundary conditions in the Front End Processors, the FEPs never produce events centered on row 0, column 0, row 1023 or column 1023.

4. Only event centers must be in the window

   For event processing, only the center pixel of the event must be within the window for the event to be processed by the window. In raw mode, any raw pixel which is within the window is processed by the window. The wording in the Software Requirements Specification is a little confusing on this point.

4.1.4.3 Continuous Clocking Parameter Blocks

Description

Slots

ACIS has five slots in which it stores Continuous Clocking Parameter blocks. The parameter blocks are loaded into a particular slot using a "Load CC Block" command packet, where the slot is specified in the ccBlockSlotIndex field. When a parameter block is loaded into a slot, the slot's contents are overwritten by the new block contained in the command. If the instrument is power-cycled or cold-booted (see Section 3.1.3.1 , Section 3.1.3.2), the contents of the slots will be restored to their "as-launched" values. A warm-boot (see Section 3.1.3.3), however, retains the contents of the most recently loaded blocks.

Checksum

Each parameter block contains a checksum field. The checksum is the bit-wise XOR of each 16-bit word in the command packet following the checksum field (i.e. starting with the least-significant 16-bit word of the parameterBlockId field).

Parameter Block Id

The Parameter Block Identifier is a 32-bit value which can be used by the ground to uniquely identify the parameter block. ACIS sends the value of the Parameter Block Id in all exposure record, bias map, bias error packets and science report packets which used the parameter block for a bias computation and/or science run. In some cases, the Parameter Block used for science run data processing may be different than the parameter block used to compute the bias map used for the run, in which case, the telemetered Bias Parameter Block Id will be different than the main data processing Parameter Block Id (these are distinct fields in the telemetry packets).

NOTE: The intent of the Parameter Block Id is to aid bookkeeping on the ground. ACIS does not use the id internally for any particular purpose, and therefore, the user's are free to use the field as they wish.

CCD Selection

The Continuous Clocking Parameter Block contains an array of CCDs which are indexed by FEP Id. The user selects which CCDs to use for the science run by assigning the CCD to the particular FEP. A FEP can be designated as unused by assigning the CCD_DESELECT value for the CCD Id field. A CCD may be processed by more than 1 FEP, however, subsequent parameters may conflict (see below). In general, if a more than 1 FEP is processing data from the same CCD, the higher order FEP overwrites the parameters used by the lower (NOTE: This is a side-effect. ACIS applies all of the parameters from all of the FEPs, in FEP order. The last FEP wins). The affected parameters are the CCD Video Response parameters and the FEP Video Offset parameters. For example, if one configured:

```
fepCcdSelect[FEP_0]          = CCD_I0
fepCcdSelect[FEP_1]          = CCD_I1
fepCcdSelect[FEP_2]          = CCD_I0
fepCcdSelect[FEP_3 .. FEP_5]  = CCD_DESELECT
and
ccdVideoResponse[FEP_0]       = 0
ccdVideoResponse[FEP_1]       = 0
ccdVideoResponse[FEP_2]       = 1
and
fep0VideoOffset[FEP_0]        = [60,60,60,60]
fep0VideoOffset[FEP_1]        = [60,60,60,60]
fep0VideoOffset[FEP_2]        = [30,30,30,30]
```

then CCD_I0 would use the low-gain video response (Video Response = 1) and 30 for the video ADC offsets.

In Continuous Clocking mode, all the CCDs shift their respective rows in unison, avoiding the staggered capture complexities of Timed Exposure mode.

If a CCD’s video board is not powered, or if the FEP is not powered, or if an error is encountered while setting up for the run, the offending item will not be used for the run.

Processing Modes

The user selects which ACIS processing mode to use with the fepMode and bepPackingMode parameters. The following indicates how the modes are selected, and what other parameters are affected (note the ignore initial frames, clocking parameters, video offsets and special parameters are always enabled):

TABLE 16. Continuous Clocking Processing Mode Selection

Mode	fepMode	BEP Packing Mode	Telemetry Packets	Enabled Parameters
Raw	Raw	ignored	1. (1) dumpedCcBlock 2. (1) deaHousekeepingData 3a. (1..n) dataCcRaw 3b. (1..n) exposureCcRaw 4. scienceReport	rawCompressionSlotIndex windowSlotIndex
Faint 1x3	1x3	Faint	1. (1) dumpedCcBlock 2. (1) deaHousekeepingData 3. (0 or n) dataCcBiasMap 4a. (1..n) dataCcFaint 4b. (1..n) exposureCcFaint 5. (1) scienceReport	all bias parameters fepEventThreshold fepSplitThreshold lowerEventAmplitude eventAmplitudeRange gradeSelections windowSlotIndex
Graded	3x3	Graded	1. (1) dumpedCcBlock 2. (1) deaHousekeepingData 3. (0..n) dataCcBiasMap 4a. (1..n) dataCcGraded 4b. (1..n) exposureCcFaint 5. (1) scienceReport	

Clocking Parameters

TBD - Reference to copy of pramcc.doc description of Continuous Clocking

Continuous Clocking always clocks a collection 512 rows from the CCD to the DEA/DPA. Although the timing relationship between sets of 512 rows is seamless, this unit is used to re-compute average overclocks and adjust the threshold levels, and is treated by the system as an image, or “exposure”. The Continuous Clocking Parameter Block contains a set of parameters which control how the CCDs are clocked, and the format of the resulting images (NOTE: some combinations of these parameters are not supported. Refer to TBD). The clocking parameters are as follows:

- Row Sum and Column Sum

  These parameters control the on-chip summing of rows and columns performed in Continuous Clocking mode. Their values are expressed in terms of powers of 2, so, for example, if Row Sum is 3, then 8 CCD rows are summed on the CCD and sent to the DEA/DPA. These parameters divide, by powers of 2, the number of rows and columns clocked out for each image, and affect the time taken to transfer the image to the DEA/ DPA for processing (see Section 4.1.1).

  NOTE: Because this parameter affects the image geometry and measured bias level, if this parameter is different than that used for last bias computation, the user must ensure that a new bias map is computed with the changed parameter.

- Overclock Pairs Per Node

  This specifies the number of overclock pixels to clock out of the CCD serial register. The parameter actually specifies the number of pairs of overclocks for each output node. For example, if a value of 15 is specified, when the Output Register Mode is in FULL mode, a total of 30 overclocks per node per row will be produced, or a grand total of 120 overclock pixels per row.

  In raw mode, this parameter affects the telemetered image geometry, and is not affected by the windowing parameters. Although, the entire image may be clipped out with a window, the overclocks will be telemetered.

  This parameter affects the time taken to transfer images to the DEA/DPA for further processing (see Section 4.1.1).

  NOTE: Because this parameter may affect baseline overclock levels, it is recommended that the bias map be recomputed when this parameter changes.

- Output Register Mode

  This parameter specifies the configuration of the CCD output nodes. If in FULL mode, the images are clocked out of all four CCD output nodes, A, B, C, and D. If in DIAG mode, the images are clocked ina reverse direction, away from the four output nodes. This is used to measure clocking noise in the DEA electronics by clocking the CCDs, but not moving any charge into the electronics. From the instrument software’s point of view, the resulting image appears as if it were clocked in FULL mode. AC and BD modes are intended for use if one of the video chains fail. AC mode clocks the serial registers through the A and C nodes. In BD mode, only the B and D nodes are used. Since only half the video chains are used, these modes take twice as long FULL or DIAG mode (see Section 4.1.1).

  NOTE: Because this parameter changes the output nodes, and therefore, the baseline overclock levels, it is recommended that the bias maps be re-computed whenever this parameter is changed.

- Video Responses

These parameters control the gain of the measured signal by changing the A/D integration timing on the video boards. If 0, then normal gain is used. If set to 1, then the corresponding CCD signal gain is cut by about 75%. Currently, it is not expected that this feature will ever be used. This parameter only affects the signal quality, and does not affect the image geometry or transfer time to the DEA/DPA. This parameter may require adjustments to the FEP Video Offset parameters, and the various threshold and bias parameters.

NOTE: Because these parameters dramatically affects the signal gains, the bias-maps should be recomputed whenever these parameters change.

Video ADC Offsets

Because the clocking parameters can affect the signal levels measured on the video boards, the Timed Exposure Parameter block specifies the Analog-to-Digital Converter (ADC) video offset levels, which override those stored in the System Configuration Table (see Section 3.3.4). The video offset parameters act inversely to the video chain signal level (i.e. larger values produce a lower offset). The conversion formula for these values is specified in TBD.

NOTE: Because the video offset parameters may affect the signal measurement levels and affect the CCD bias levels, it is recommended that the bias maps be recomputed whenever these parameters are changed, although, if the offsets are adjusted to compensate for changes in the other parameters in order to maintain a consistent bias level, one may not require a bias re-computation.

Bias (and related) Parameters

ACIS re-computes the bias maps under one or more of the following conditions:

1. The recomputeBias flag is set in the parameter block
2. A Bias-Only run was requested (see Section 4.2)
3. An event processing mode and

- The FEP’s bias offsets were corrupt or
- The previous Science Run was not a Continuous Clocking run or
- The radiation monitor had been asserted since the last bias computation

NOTE: Many of the CCD clocking parameters can affect the bias levels of the CCD images, and require re-computation of the map. The instrument software does NOT automatically detect these changes. It is the user’s responsibility to determine when the bias maps should be re-calculated, and request computation of the bias maps when needed.

The ignoreInitialFrames parameter indicates how many exposures the FEPs are to throw away prior to computing the bias map. This is to allow for the bias-levels of the CCDs to settle. (NOTE: For data processing, 2 exposures are always discarded, hence, all reported exposure numbers start at 2).

ACIS supports a variety of bias computation algorithms, each requiring a collection of parameters. The algorithm selection and parameters are supplied independently for each FEP in the biasAlgorithmId and biasArg parameters. The details of the algorithms and the effect of the parameters are provided in TBD.

If the ignoreBadColumnMap is 0, ACIS installs the bad columns, indicated in the Continuous Clocking bad column map, into the bias maps whenever the maps are re-computed. If the ignoreBadColumnMap is 1, the column pixels are not flagged. (see Section 3.5).

If a bias is re-computed, and the trickleBias flag set to a 1, then the bias maps from each used FEP are telemetered prior to the start of data processing. If the trickleBias flag is 0, then the maps are not telemetered.

NOTE: The maps are sent only when they are re-computed. If trickleBias is 1, but the map wasn't computed as part of the run, the map will not be telemetered at that time.

Event Selection Parameters

The event selection parameters are only used when the ACIS is configured for Faint 1x3 and Graded modes. The details of the event selection algorithm are provided in TBD. These parameters are:

- FEP Pixel Thresholds

  The FEP pixel thresholds, fepEventThreshold, are indexed by FEP and output node. For each FEP, the BEP loads the four output node thresholds into the corresponding Front End Processor. The user supplies these values in bias-corrected Analog-to-Digital unit (ADU) values. The FEP hardware uses the thresholds to locate pixels, in incoming CCD images, whose measured pulse heights are large enough to indicate possible detected X-ray events. The FEP software and hardware effectively adjust these values to correct for drifts in the average overclock levels from the DEA video chains, and for pixel-to-pixel bias differences, using the computed pixel bias maps.

- BEP Split Thresholds

  The BEP split thresholds, fepSplitThreshold, are indexed by FEP and output node. The Back End software uses these values to determine which pixels, surrounding the center of a candidate X-ray event, contain charge from the event. The values are specified in bias-corrected ADUs. The BEP use this to compute an estimate of the total amplitude of the X-ray event, and its grade code. The amplitude and grade code are then used by the BEP for subsequent event filtering, and if in Graded Mode, are telemetered.

- Event Amplitude Range

The lowerEventAmplitude and eventAmplitudeRange parameters specify the range of estimated amplitudes to accept from candidate events. If a candidate event's estimated amplitude is less than the lowerEventAmplitude or is greater than or equal to (lower-EventAmplitude + eventAmplitudeRange), the event is rejected and not sent to the ground. If the event is within the range, then it is passed on to the grade and possibly window selection filters.

- Event Grade Selection

  The gradeSelections parameter is an array of 4 bits, where each bit corresponds to 1 grade code. If the user sets a bit to 1, then events whose grade code matches the corresponding entry may be telemetered. If a bit is 0, then such events are rejected, and are not telemetered. If an event matches a grade which is enabled, then the event is passed to the window filter.

  TBD - Reference Grade Code Table

- Window Processing

  If the windowSlotIndex is 0xffff, then no window processing is performed on the candidate events, otherwise, the 1-D Window Parameter Block stored in the corresponding slot is used to process the candidate events. See Section 4.1.4.2 for a description of this processing.

Raw Image Processing

If the run is performing Raw Mode, then the following parameters are used to filter and telemeter the data:

- Window Processing

  If the windowSlotIndex is 0xffff, then no window processing is performed on the raw image, otherwise, the 1-D Window Parameter Block stored in the corresponding slot is used to process the image. See Section 4.1.4.2 for a description of this processing.

  NOTE: Window processing is not applied to the overclock data. All overclocks from all of the rows in the image are telemetered.

- Compression table

  The rawCompressionSlotIndex indicates which compression table to use to pack the raw data. If the field is set to 255, then the maps will be bit-packed, with no compression. Refer to TBD for a description of the current Huffman compression tables

Special Cases

The DEA Load Override parameter is provided to allow the maintainer to specify an explicit PRAM/SRAM image to load into the sequencers.This parameter causes the software to by-pass the SRAM library, and the on-board PRAM builder, and will ignore all clocking parameters when building the load. It is the user's responsibility to place the load image somewhere in the BEPs data memory, and ensure that the clocking parameters in the parameter block are consistent with the images produced by the explicit load.

NOTE: The DEA sequence checker will still verify that the loaded PRAM/SRAM does not threaten the health of the DEA electronics.

The FEP Load Override parameter can be used by the maintainer to provide alternate software to run in the FEP hardware. It is the user’s responsibility to ensure that the FEP load image is stored somewhere in the BEP’s data memory, and that the loaded program is compatible with the BEP-FEP protocols expected by the BEP software.

Commands

Summary of Load Continuous Clocking Parameter Block (see TBD for an example):

```
loadCcBlock: CMDOP_LOAD_CC
{
   ccBlockSlotIndex    = 0..5    - Specify which slot to load
   checksum            = checksum- 16-bit wide XOR of remainder of
                                   packet (illustrated in bold-type)
   parameterBlockId    = id      - Ground selected Parameter Block Id
   fepCcdSelect[6]     = CCD_I0..CCD_S5, CCD_DESELECT
                                 - Array indexed by FEP Id of which
                                   CCD each FEP should use
   fepMode             = Raw, or 1x3
   bepPackingMode      = Faint, or Graded
   ignoreBadPixelMap   = 0..1    - Don’t load bad pixels after bias
   ignoreBadColumnMap  = 0..1    - Don’t load bad columns after bias
   recomputeBias       = 0..1    - Force re-computation of bias
   trickleBias         = 0..1    - Send bias maps after computation
   rowSum              = 0..9    - Number of rows to sum (powers of 2)
   columnSum           = 0..6    - Number of columns to sum (powers of
                                   2)
   overclockPairsPerNode= 0..15  - Number of overclock pairs
   outputRegisterMode  = FULL, DIAG, AC, BD
                                 - Output register configuration
   ccdVideoResponse[6] = 0..1    - Normal or low-gain, indexed by FEP
   fep*EventThreshold[4]= -4096..4095
                                 - FEP threshold values for each FEP.
                                   Indexed by output node.
   fep*SplitThreshold[4]= 0..4095- BEP threshold values for data from
                                   each FEP. Indexed by output node.
   lowerEventAmplitude = 0..4095 - Minimum event amplitude
   eventAmplitudeRange = 0..24570- Maximum amplitude above minimum
   gradeSelections[4]  = 0..1    - Select accepted event grade codes
   windowSlotIndex     = 0..4,255- Select 1-D window slot
   rawCompressionSlotIndex= 0..255- Compression table for raw images
   ignoreInitialFrames= 0..65535 - Exposures to drop prior to bias
   biasAlgorithmId[6]  = 0..255  - Bias Algorithm to use, indexed by
                                   FEP
   biasRejection[6]    =  value  - Bias Rejection Level, per FEP
   fep*VideoOffset[4]  = 0..255  - ADC Video Offsets for each FEP,
                                   indexed by output node.
   deaLoadOverride     = 0, pointer- If not 0, BEP Memory pointer to
                                   DEA load image
   fepLoadOverride     = 0, pointer- If not 0, BEP Memory pointer to
                                   FEP load image
}
```

Science Telemetry

Command Echo on a successful load:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   loadCcBlock: CMDOP_LOAD_CC
   {
      ccBlockSlotIndex    = 0..5
      checksum            = checksum
      parameterBlockId    = id
      ...
   }
}
```

If the checksum of the packet doesn't match its contents, the block will not be loaded into the slot, and the Command echo will report the error by setting the result field to CMDRESULT_STORE_ERROR.

NOTE: If the parameter block contains fields which are out of range, or which are illegal in combination with other fields, an attempt to start a science run will abort, with the terminationReason in the Science Report set to SMTERM_PROC_PARM_INVALID, SMTERM_DEA_PARM_INVALID, SMTERM_FEP_PARM_INVALID. However, if the user specifies a set of clocking parameters which are legal, but not supported by the on-board SRAM library, it will appear as DEA error, and the terminationReason will be set to SMTERM_DEA_IO_ERROR.

Warnings

1. Video Offsets are overrides

   The video offset parameters in the parameter block override those specified in the System Configuration Table (see Section 3.3.4). The video offset parameters in the System Configuration Table are never used.

2. Use System Configuration to control board power

   The science parameter blocks DO NOT apply or remove power from hardware boards. If a video board or FEP is not powered, it will not be used for the run. If none of the requested boards are powered, the run will be aborted.

3. Some clocking combinations are not supported

   Certain combinations of clocking parameters cause the software to access SRAM library components which don't exist, and will abort the run. See TBD.

4.1.4.4 1-D Window Parameter Blocks

Description

Slots

ACIS has five slots in which it stores 1-D Window Parameter blocks. The parameter blocks are loaded into a particular slot using a “Load 1D Block” command packet, where the slot is specified in the windowSlotIndex field. When a parameter block is loaded into a slot, the slot’s contents are overwritten by the new block contained in the command. If the instrument is power-cycled or cold-booted (see Section 3.1.3.1 , Section 3.1.3.2), the contents of the slots will be restored to their “as-launched” values. A warm-boot (see Section 3.1.3.3), however, retains the contents of the most recently loaded blocks.

Checksum

Each parameter block contains a checksum field. The checksum is the bit-wise XOR of each 16-bit word in the command packet following the checksum field (i.e. starting with the least-significant 16-bit word of the windowBlockId field).

Window Block Id

The Window Block Identifier is a 32-bit value which can be used by the ground to uniquely identify the parameter block. ACIS sends the value of the Window Block Id in all exposure record packets and science report packets which used the windows for a science run.

NOTE: The intent of the Window Block Id is to aid bookkeeping on the ground. ACIS does not use the id internally for any particular purpose, and therefore, the users are free to use the field as they wish.

Window Definitions

The remainder of the window parameter block contains a series of zero or more window definitions. The number of windows in the block is determined using the command packet length. Each window definition specifies the CCD to which the window applies, the column position of the left edge of the window, and the width of the window, all in CCD coordinates. Each window also specifies an event amplitude range, which is only used for event processing, and a sample field, which is used for both event processing, and for raw-mode data clipping. Refer to TBD for an explicit description of window processing of event data, and TBD for a description of window processing of raw mode data. ACIS is capable of supporting up to 6 windows for each CCD used for a run. The parameter block can hold up to the maximum of 60 window definitions (6 windows x 10 CCDs).

Commands

Summary of Load 1DWindow Block:

```
load1dBlock: CMDOP_LOAD_1D
{
   windowSlotIndex     = 0..5    - Specify which slot to load
   checksum            = checksum- 16-bit wide XOR of remainder of
                                   packet (illustrated in bold-type)
   windowBlockId       = id      - Ground selected Window Block Id
   windows[]           =         - Array of window definitions
   {
      ccdId            = I0..S5  - Identify CCD
      ccdColumn        = 0..1023 - Left most column of window
      width            = 0..1023 - Width of window - 1
      sampleCycle      = 0..255  - Event sample count, or raw pixel
                                   selection
      lowerEventAmplitude = 0..4095 - Minimum event amplitude
      eventAmplitudeRange = 0..24570- Maximum amplitude above minimum
   }
   {
      ccdId ...
   }
   ...
}
```

Science Telemetry

Command Echo on a successful load:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   load1dBlock: CMDOP_LOAD_1D
   {
      windowSlotIndex     = 0..5
      checksum            = checksum
      windowBlockId       = id
      windows[]           =
      {
         ccdId            = I0..S5
         ccdColumn        = 0..1023
         width            = 0..1023
         sampleCycle      = 0..255
         lowerEventAmplitude = 0..4095
         eventAmplitudeRange = 0..24570
      }
      {
         ccdId ...
      }
   ...
   }
}
```

If the checksum of the packet doesn't match its contents, the block will not be loaded into the slot, and the Command echo will report the error by setting the result field to CMDRESULT_STORE_ERROR.

NOTE: If the parameter block contains fields which are out of range, or which are illegal in combination with other fields, an attempt to start a science run will abort, with the terminationReason in the Science Report set to SMTERM_PROC_PARM_INVALID.

Warnings

1. First window in the list hides subsequent windows

   Contrary to how bit-mapped graphics are produced, and paste-up sheets are overlaid on a table, the windows in the parameter block overlap in a queue-like fashion, where the last window in the parameter block is the bottom most window.

2. Column 0 and 1023 never produce events

   In order to handle the event detection boundary conditions in the Front End Processors, the FEPs never produce events centered on column 0 or column 1023.

3. Only event centers must be in the window

   For event processing, only the center pixel of the event must be within the window for the event to be processed by the window. In raw mode, any raw pixel which is within the window is processed by the window. The wording in the Software Requirements Specification is a little confusing on this point.

4.1.5 Starting a Science Run

Description

Prior to starting a science run, the user should have powered on the FEPs and Video boards needed for the run (see Section 3.3.2 and Section 3.3.3) and planned and loaded the parameter and window blocks needed for the run (see Section 4.1.1 , Section 4.1.2 , Section 4.1.3 , Section 4.1.4). The user initiates a science run using the “Start Science” command packet, where the opcode of the command selects the type of science run to perform, and the blockSlotIndex parameter selects which parameter block to use to configure the run. If the opcode is CMDOP_START_TE, then a Timed Exposure run is performed, and the parameter block is read from the slots reserved for Timed Exposure runs. If the opcode is CMDOP_START_CC, the a Continuous Clocking run is performed, and the parameter block is read from the Continuous Clocking slot array.

Once started, the Science Manager will configure the processing parameters, load the CCD settings into the DEA video boards, load the sequencer memory of the video boards, configure the Front End Processor parameters. It will then “jitter” the video board DACs to clear charge out of the CCDs. If a bias computation is required, the task will command the FEPs to compute a bias, and then start the video board sequencers. Once the bias is complete, the task will stop the sequencers, and will send the bias maps to telemetry, if requested by the “trickleBias” flag. Once the bias maps have been sent, the task will command the FEPs to prepare for data, and re-start the video board sequencers. The FEPs will then ignore the first two images from the CCDs, and then start processing the data from the subsequent exposures, producing data packets and exposure records. The task will continue to do so, until either it is commanded to stop, via a “Stop Science” command, inhibited by the radiation monitor, aborted by a second “Start Science” command (in which case, a new run will be configured and started), or until the run is aborted due to errors (NOTE: The system attempts to degrade gracefully. If a FEP produces an error, it is not used for the remainder of the run. The remaining FEPs, however, will continue to be used. If all of the FEPs are in error, then the run is aborted because there’s nothing to do).

Commands

Command to start a Timed Exposure Science Run:

```
startScience: CMDOP_START_TE
{
   blockSlotIndex   = 0..4 - Index Timed Exposure Parameter Block Slot
}
```

Command to start a Continuous Clocking Science Run:

```
startScience: CMDOP_START_CC
{
   blockSlotIndex   = 0..4 - Index Continuous Clocking Parameter Slot
}
```

Engineering Telemetry

When performing a science run, the Software Housekeeping task blinks the LED values between two values, about once a minute (see Section 3.4.3.1).

If system had not watchdog-reset since the last commanded reset:

```
LED_RUN_SCIENCE_A
LED_RUN_SCIENCE_B
```

If the system had watchdog-reset:

```
LED_WD_SCIENCE_A
LED_WD_SCIENCE_B
```

Science Telemetry

Command Echo on a successful receipt of Timed Exposure start request:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP 10Hz Ticks)
   result    = CMDRESULT_OK (other values indicate an error)

   startScience: CMDOP_START_TE
   {
```

```
            blockSlotIndex   = 0..4
         }
      }
```

Command Echo on a successful receipt of Continuous Clocking start request:

```
      commandEcho: TTAG_CMD_ECHO
      {
         arrival   = time command arrived (BEP 10Hz Ticks)
         result    = CMDRESULT_OK (other values indicate an error)

         startScience: CMDOP_START_CC
         {
            blockSlotIndex   = 0..4
         }
      }
```

If a run had been started, and not yet stopped, when the command is received, the "result" field of the echo will contain CMDRESULT_CLOBBERED. In this case, the earlier run will be aborted (NOTE: The FEPs will be reset), and the new run will be configured and started.

If the radiation monitor had been activated, then the start of the run will be deferred until the monitor is de-activated. In this case the "result" field will contain CMDRESULT_INHIBITED.

If the selected parameter block checksum does not match its contents (i.e. has been corrupted), the system will attempt to determine if the run was configured to use the Imaging CCDs or the Spectroscopy CCDs. If any of the CCDs in the fepSelect field of the parameter block choose an Imaging CCD (I0..I3), the system will attempt to use the parameter block contained in slot 0. If not, it will try to default to the parameter block in slot 1. If the default is not corrupt, the intent was for the system to use the default parameter block, and the "result" field will be set to CMDRESULT_CORRUPT_DEFAULT. Due to a "bug" in the software, however, the system will use the most recently configured parameter block for the type of run that was chosen, either Timed Exposure or Continuous Clocking (NOTE: Some might consider this a feature).

If the original parameter block is corrupt, and the selected default parameter block is also corrupt (or they're one and the same), the run will not be started, and the "result" field will contain CMDRESULT_CORRUPT_IDLE.

Since the field ranges within the parameter block are only checked during the set-up stage of the run, after the receipt of the command has been acknowledged via the command echo, if the a parameter contained in the block is invalid or unsupported, the Command Echo will not indicate the problem, and supply a CMDRESULT_OK. Instead, the run will not be started, and a "Science Report" packet will indicate the problem within its terminationReason field.

At the start of the run, the Science Manager will fetch and telemeter the parameter blocks used for the run. In Timed Exposure Mode, the dump appears as:

```
dumpedTeBlock: TTAG_DUMP_TE
{
   dumped TE Block:
   loadTeBlock: CMDOP_LOAD_TE
   [
      ...
   }
   if windows were used, this is followed by:
   a dumped 2d Window Block, aligned to the next 32-bit word:
   load2dBlock: CMDOP_LOAD_2D
   {
      ...
   }
}
```

In Continuous Clocking Mode, the dump appears as:

```
dumpedCcBlock: TTAG_DUMP_CC
{
   dumped CC Block:
   loadCcBlock: CMDOP_LOAD_CC
   [
      ...
   }
   if windows were used, this is followed by:
   a dumped 1d Window Block, aligned to the next 32-bit word:
   load1dBlock: CMDOP_LOAD_1D
   {
      ...
   }
}
```

Once the parameter blocks have been dumped, the Science Manager acquires and telemeters the DEA Video Board housekeeping information for all CCDs used for the run (NOTE: Unless all 10 video boards are powered on, the analog values in the packet will be of marginal use. They will indicate if the given channel is completely broken, but won't really indicate the level that the DAC was set to). The content and form of this information is described in Section 3.4.6.6 .

If a bias map was re-computed, and the "trickleBias" flag was set in the parameter block, the system will telemeter the contents of the computed bias maps of each FEP, in FEP order (i.e. FEP_0 first, then FEP_1, etc.). The Timed Exposure Bias Map telemetry packets are of the form:

```
dataTeBiasMap: TTAG_SCI_TE_BIAS
{
   biasStartTime            = 100KHz time-stamp at start of bias
   biasParameterId          = Parameter Block Id used for bias
   ccdId                    = CCD which produced the map
   fepId                    = FEP used to produce the map
   dataPacketNumber         = packet sequence number within this map
   initialOverclocks[4]     = DC-offset of bias values, by quadrant
   pixelsPerRow             = number of pixels in each bias map row
   rowsPerBias              = number of rows in the map
```

```
        ccdRow                      = starting row of data in the packet
        ccdRowCount                 = number of rows packed into the packet
        compressiontTableSlotIndex = table slot value used to compress data
        pixelCount                  = total number of pixels in the packet
        data[]                      = packed or compressed data, as array of
                                      32-bit words
    }
```

The Continuous Clocking Bias Map telemetry packets are of the form:

```
    dataCcBiasMap: TTAG_SCI_CC_BIAS
    {
        biasStartTime               = 100KHz time-stamp at start of bias
        biasParameterId             = Parameter Block Id used for bias
        ccdId                       = CCD which produced the map
        fepId                       = FEP used to produce the map
        data[1024]                  = packed 12-bit pixel bias values
    }
```

Once processing data, the Science Manager task produces a series of telemetry data packets, followed by 1 exposure record, for each FEP. The data and exposure packets for the different FEPs are asynchronously interleaved into the telemetry stream. The form of the data and exposure packets vary, depending on the type of mode being run. See Table 15 and Table 16 for cross-references between the type of mode and the resulting telemetry packet types.

Warnings

1. Command Echo does not check parameter ranges

   Parameter block fields are only checked as they are used to setup for the run. If there is an error in one or more fields, the run may be prematurely terminated, with the Science Report's terminationReason indicating the type of error. ACIS does not explicitly call-out which parameter is in error, however.

2. Re-starting aborts the earlier run

   If a run is in progress, and a subsequent "Start Science" command is received, the FEPs will be reset and the earlier run will be aborted, after which the new run will be configured and started. Sometimes this takes a long time, and may lead to confusion when a "human" is interacting with the instrument. The condition is indicated in the command echo to the "Start Science" command, and the bi-levels accurately reflect the current state of the instrument.

3. Re-starting runs may cause a FEP error to be reported

   When a run is re-started, the earlier run is aborted. This abort forces a reset of the FEPs to ensure that nothing gets locked up. If the previous run is interacting with a FEP when the reset is issued, its Science Report may indicate that FEP errors have occured.

4. Radiation Monitor can inhibit a run

   If a "Start Science" is received while the radiation monitor flag is asserted, the run won't start until the radiation monitor is de-asserted.

5. Due to a "bug", corrupted blocks will cause the use of the most recent block

Although the software requirement states that either parameter block slot 0 or 1 will be used in the event that the requested parameter block is corrupted, the system actually re-uses the parameters from the previous run of the same type, either Timed Exposure or Continuous Clocking. The run may fail to start if no runs of the given type have been successfully started since the instrument was last powered on.

6. See Section 3.4.6.6 Warnings

4.1.6 Stopping a Science Run

Description

The user stops a science run by issuing a “Stop Science” command packet. Upon receipt of the command, the Science Manager task will issue a stop request to the FEPs, and poll them until they indicate that they have no more data to send from the current exposure. Once the current exposure is complete, the Science Manager task stops the sequencers and forms and posts a “Science Report” telemetry packet.

Commands

Command to stop a science run:

```
stopScience: CMDOP_STOP_SCIENCE
{
   No additional parameters are required
}
```

Engineering Telemetry

Once a science run is complete, the Software Housekeeping task will revert the LEDs to their idle blinking states (see Section 3.4.3.1).

If system had not watchdog-reset since the last commanded reset:

```
LED_RUN_IDLE_A
LED_RUN_IDLE_B
```

If the system had watchdog-reset:

```
LED_WD_IDLE_A
LED_WD_IDLE_B
```

Science Telemetry

Once a run terminates, either due to a stop command, a radiation monitor inhibit, or an abort, the Science Manager produces a “Science Report” telemetry packet:

```
scienceReport: TTAG_SCI_REPORT
{
   runStartTime          = 100KHz time-stamp when data started
   parameterBLockId      = Parameter Block Id used for data
   windowBlockId         = Window Block Id used to filter
   biasStartTime         = 100KHz time-stamp when bias was computed
   biasParameterId       = Parameter Block Id used for bias
```

```
    exposuresProduced      = Largest Exposure Number produced by FEPs
    exposuresSent          = Number of exposures telemetered
    biasErrorCount         = Number of bias errors telemetered
    fepErrorCodes[6]       = FEP Error codes, indexed by FEP
    ccdErrorFlags[6]       = Video Board error flags, indexed by FEP
    deaInterfaceErrorFlag  = DEA Interface board error flag
    terminationCode        = Reason the run was stopped
}
```

The FEP Error codes are (likely user-inflicted cases are in **bold**):

```
FEP_CMD_NOERR           - no errors detected, the FEP is ok
FEP_CMD_ERR_NO_RUN      - stop received, but it was already stopped.
                          This may occur if many stop commands are
                          issued to the instrument in a row.
FEP_CMD_ERR_UNK_CMD     - Unknown command type. SEU or bug.
FEP_CMD_ERR_PARM_LEN    - Parameter block too long. SEU or bug.
FEP_CMD_ERR_PARM_TYPE   - Unknown parameter block type. SEU or bug.
FEP_CMD_ERR_QUAD_CODE   - Unknown quadrant code. SEU or bug.
FEP_CMD_ERR_BIAS_TYPE   - Bad biasAlgorithmId in parameter block
FEP_CMD_ERR_BIAS_PARM0  - Bad biasArg0 in parameter block.
FEP_CMD_ERR_NROWS       - Bad number of rows. SEU or bug.
FEP_CMD_ERR_NCOLS       - Bad number of columns. SEU or bug.
FEP_CMD_ERR_NOCLK       - Bad number of overclocks. SEU or bug.
FEP_CMD_ERR_NHIST       - histogramCount was zero in Histogram Mode
FEP_CMD_ERR_NO_PARM     - No parameter block loaded. SEU or bug.
FEP_CMD_ERR_BAD_CMD     - Illegal secondary command. SEU or bug.
FEP_CMD_ERR_NO_BIAS     - No bias map stored. SEU or bug

FEP_ERR_LOCK_TIMEOUT    - Time-out on FEP lock. SEU, bug or FEP crash
FEP_ERR_NO_POWER        - FEP is not powered on
FEP_ERR_IS_RESET        - FEP is reset, FEP probably crashed.
FEP_ERR_NO_CMDRING      - FEP program has no command mailbox, bad fep-
                          LoadOverride.
FEP_ERR_REPLY_TIMEOUT   - FEP reply timed-out. FEP probably in the pro-
                          cess of crashing but hasn’t watchdogged yet.
FEP_ERR_BAD_REPLY_TYPE  - FEP produced bad reply. SEU or bug
FEP_ERR_BAD_MBOX_STATE  - FEP mailbox in wrong state. SEU or bug.
```

The Termination Codes are (normal and user-inflicted cases are in **bold**):

```
SMTERM_UNUSED              - Unused. Should never see this one.
SMTERM_STOPCMD             - Commanded to Stop i.e. normal term.
SMTERM_BIASDONE            - Bias-only Run completed
SMTERM_RADMON              - Radiation Monitor was asserted
SMTERM_CLOBBERED           - Clobbered by another start command
SMTERM_FEP_BIAS_START      - FEP Bias Processing did not start
SMTERM_FEP_DATA_START      - FEP Data Processing did not start
SMTERM_CCD_BIAS_START      - Start clock failed. CCDs for bias failed
SMTERM_CCD_DATA_START      - Start clock failed. CCDs for data failed
SMTERM_CCD_BIAS_STOP       - Stop clock failed. CCDs for bias failed
SMTERM_PROC_PARM_INVALID   - Processing Parameter out of range
SMTERM_DEA_PARM_INVALID    - DEA Parameter out of range
SMTERM_FEP_PARM_INVALID    - FEP Parameter out of range
SMTERM_FEP_CONFIG_ERROR    - FEP Configuration Error
SMTERM_DEA_IO_ERROR        - I/O errors, or no CCD controllers on
SMTERM_FEP_IO_ERROR        - I/O errors, or no FEPs are on
SMTERM_UNSPECIFIED         - Reason is unspecified
```

Warnings

1. Runs stopped due to radiation monitor may re-start later

   Science runs which terminate due to the assertion of the radiation monitor, will re-start once the monitor is de-asserted, unless a “Stop Science” is received.

2. Runs may abort due to bad parameters

   Invalid parameters will cause a science run to abort during its setup stage. The report will set the termination code to either: SMTERM_PROC_PARM_INVALID, SMTERM_DEA_PARM_INVALID, SMTERM_FEP_PARM_INVALID, or SMTERM_FEP_CONFIG_ERROR.

3. Stopping runs take time

   Upon receipt of a “Stop Science” command, ACIS attempts to complete the current exposure. If the system is in “Raw” mode, or if there are a large number of events in the exposures, this may take quite a while to finish. If, during that time, another “Start Science” command is received, the stop will be aborted and the FEPs will be reset. Depending on the timing, the resulting termination code may be SMTERM_CLOBBERED, or may indicate some form of FEP error.

4. exposuresProduced is confusing

   The exposuresProduced field is badly named. Although its name implies that its the number of exposures clocked out of all of the CCDs, it is actually 1 plus the largest exposure number produced by any of the FEPs.

5. exposuresSent is confusing

   The exposuresSent field indicates the sum of the number of exposures telemetered from all of the FEPs.

4.2 Performing a Bias-Only Science Run

This section describes the activities used to perform a bias-only science run. Some of the information presented in this section also applies to the optional bias computation stage of a science run (see Section 4.1).

The overall process of performing a bias-only science run within ACIS is similar to that for a normal science run, except that the run automatically terminates once the bias maps have been calculated and, if configured to do so, telemetered.

4.2.1 Estimating Bias Computation Time

4.2.1.1 Continuous Clocking Mode

Description

In Continuous Clocking Mode, the bias values for the columns is computed using 1024 samples, from two consecutive data sets of 512 rows each. The computation time is almost always less than the frame time.

Frame Time

The frame time in Continuous Clocking Mode is given by:

```
frame_time := 512 * second/row
where the second/row is a function of the summing parameters. Refer to
Section 4.1.1.1 for the formula.
```

Bias Time

Rounding up to the nearest frame time for time to calculate the actual bias values, the total bias computation time for Continuous Clocking mode is:

```
bias_time (seconds) := (frame_time * (2 + 1))
```

4.2.1.2 Timed Exposure Mode

Description

In Timed Exposure Mode, the time taken to compute the bias maps depends heavily on the number of exposures needed to compute the bias, which, in turn, depends on the clocking parameters and bias parameters used, and the number of exposures discarded because the algorithm did not keep up with the incoming exposures.

Whether or not the algorithm keeps up is a function of speed of the algorithm running in the FEPs, and the number of events and background events in the CCD images.

Ignoring the problem of discarded exposures for the moment (which may multiply the time by factors of 2 or more), the following describes the amount of time required to compute a bias for Timed Exposure Mode, in terms of the of the parameters in the Timed Exposure Parameter Block (see Section 4.1.4.1).

Frame Time

First, compute the frame time for the primaryExposure, and if dutyCycle is not zero, the secondaryExposure times. If the desired exposure time is not a “short-exposure” (see Section 4.1.1.2), the frame time for the primaryExposure is:

```
(primaryExposure / 10) + 0.04104 seconds
(i.e. the static integration time plus the “smear” time)
```

If the exposure time is a “short-exposure”, then the frame time is:

```
(staggered_smear_time + transfer_time) + (primaryExposure / 10) +
0.04104 seconds

Where (staggered_smear_time + transfer_time) is the minimum non-short
integration time described in Section 4.1.1.2
```

For the frame time of the secondaryExposure field, replace “primaryExposure” with “secondaryExposure” in the above expressions.

Whole-Frame Mode - Number of Frames

The number of frames used to compute a bias level in Whole-Frame mode is specified by either the biasArg0 or biasArg1, which ever is greater.

```
number_of_frames := max(biasArg0, biasArg1)
```

Strip-Mode - Number of Frames

In “Strip” mode, the image buffer is divided into “strips” of rows, where each strip contains 1 sample for each pixel in the collection of rows. The bias algorithm collects these strips for a set of rows, and then computes the bias level for those rows. There is always at least 1 exposure dropped during the computation phase. The following estimate assumes that only 1 exposure is dropped. However, in practice, the number of exposures dropped will vary.

If the number of exposures per pixel, biasArg0, divides evenly into the image buffer size of 1024 rows, then the number of frames is:

```
number_of_frames := (biasArg0 + 1) * biasArg0
```

If biasArg0 does not divide evenly, the expression is:

```
number_of_frames := ((biasArg0 + 1) * biasArg0) + biasArg0

(this assumes the computation of the last set of strips is small com-
pared to the frame time)
```

Bias Time - Single Integration Time

If using a single integration time (i.e. dutyCycle == 0), then the time taken to compute the bias is:

```
      bias_time (seconds) := primary_frame_time *
                             (ignoreInitialFrames +
                              number_of_frames)
```

Bias Time - Multiple Integration Times

If using multiple integration times (dutyCycle != 0), and using an initial discard count that divides evenly by the number of exposures in a cycle ((ignoreInitialFrames MOD (dutyCycle + 1)) == 0), then the time to compute the bias is:

```
      number_of_primaries :=
                [ignoreInitialFrames DIV (dutyCycle + 1)] +
                (number_of_frames DIV dutyCycle) +
                (1 if number_of_frames MOD dutyCycle != 0)

      number_of_secondaries :=
                number_of_frames +
                [(ignoreInitialFrames * dutyCycle) DIV (dutyCycle + 1)]

      bias_time (seconds) := (primary_frame_time * number_of_primaries) +
                             (secondary_frame_time * number_of_secondaries)
```

If the initial discard count does not divide evenly, the time to compute the bias is:

```
      number_of_primaries :=
                [ignoreInitialFrames DIV (dutyCycle + 1)] +
                (number_of_frames DIV dutyCycle) +
                (1 if number_of_frames MOD dutyCycle != 0)

      number_of_secondaries :=
                number_of_frames +
                [(ignoreInitialFrames * dutyCycle) DIV (dutyCycle + 1)] +
                ([ignoreInitialFrames MOD (dutyCycle + 1)] - 1)

      bias_time (seconds) := (primary_frame_time * number_of_primaries) +
                             (secondary_frame_time * number_of_secondaries)
```

TBD -complete this section to deal with a method of determining when exposures will be dropped and how many.

4.2.2 Estimating Bias Transmission Time

Description

Continuous Clocking Mode uses less than 1 telemetry packet for each bias map, and takes less than 1 second to form and post the maps for each FEP.

Timed Exposure Mode, on the other hand, takes many packets to send the FEP bias maps. The time taken depends on the number of maps being sent, the number of rows in the bias maps, and whether or not the maps are compressed, and if so, how well they compressed.

Total Number of Pixels

The total number of rows being telemetered is:

```
number_of_rows   :=     number_of_FEPs * (subArrayRowCount+1)
pixels_per_row   :=     1024 if onChip2x2Summing == 0 or
                 :=     512 if onChip2x2Summing == 1

number_of_pixels :=     number_of_rows * pixels_per_row
```

Time if Un-compressed, 24Kbps

If the bias maps are not compressed, each pixel is packed into 12-bits in the bias data telemetry packets. Ignoring the overhead, the time to send the maps, assuming 24K bits per second of telemetry, is:

```
bias_telemetry_time (seconds) := number_of_rows * 12 bits / 24Kbps
```

For example, it takes about 53 minutes to send 6 un-compressed FEP bias maps, given a standard 1024 row image, without on-chip summing.

Time if Compressed, 24Kbps

The time taken to send compressed bias maps depends on how well the maps compress. At XRCF, the compression factor was about 4 to 1 (this is about as good as it gets), and took about 13 minutes to send all 6 bias maps. As the CCDs erode over the life of the mission, the compression will become less effective, and take longer to send the maps.

4.2.3 Loading a Parameter Block

The commands needed to load the parameter blocks for a bias-only run are identical to those for a normal science run, except that the state of the recomputeBias flag is ignored. A bias is always re-computed for a bias-only run, and the system behaves as if the recomputeBias flag is asserted. For a description of loading parameter blocks, see Section 4.1.4.1 and Section 4.1.4.3 (NOTE: Although windows are not used during a bias-only run, if a parameter block references a window parameter block, the window parameter block must not be corrupt).

4.2.4 Starting the Bias-Only Run

<u>Description</u>

Prior to starting a bias-only science run, the user should have powered on the FEPs and Video boards needed for the run (see Section 3.3.2 and Section 3.3.3) and planned and loaded the parameter blocks needed for the run (see Section 4.1.1 , Section 4.1.2 , Section 4.1.3 , Section 4.1.4). The user initiates a science run using the “Start Science” command packet, where the opcode of the command selects the type of bias-only science run to perform, and the blockSlotIndex parameter selects which parameter block to use to configure the run. If the opcode is CMDOP_BIAS_TE, then a Timed Exposure bias-only run is performed, and the parameter block is read from the slots reserved for Timed Exposure runs. If the opcode is CMDOP_BIAS_CC, the a Continuous Clocking bias-only run is performed, and the parameter block is read from the Continuous Clocking slot array.

Just as for a normal science run, once started, the Science Manager will configure the processing parameters, load the CCD settings into the DEA video boards, load the sequencer memory of the video boards, configure the Front End Processor parameters. It will then "jitter" the video board DACs to clear charge out of the CCDs. The task will command the FEPs to compute a bias, and then start the video board sequencers. Once the bias is complete, the task will stop the sequencers, and will send the bias maps to telemetry, if requested by the "trickleBias" flag. Once the bias maps have been sent, the task will automatically terminate the run.

Commands

Command to start a Timed Exposure Bias-Only Science Run:

```
startScience: CMDOP_BIAS_TE
{
   blockSlotIndex   = 0..4 - Index Timed Exposure Parameter Block Slot
}
```

Command to start a Continuous Clocking Bias-Only Science Run:

```
startScience: CMDOP_BIAS_CC
{
   blockSlotIndex   = 0..4 - Index Continuous Clocking Parameter Slot
}
```

Engineering Telemetry

When performing the bias run, the Software Housekeeping task blinks the LED values between two values, about once a minute (see Section 3.4.3.1).

If system had not watchdog-reset since the last commanded reset:

```
LED_RUN_SCIENCE_A
LED_RUN_SCIENCE_B
```

If the system had watchdog-reset:

```
LED_WD_SCIENCE_A
LED_WD_SCIENCE_B
```

Science Telemetry

Command Echo on a successful receipt of Timed Exposure start request:

```
commandEcho: TTAG_CMD_ECHO
{
   arrival   = time command arrived (BEP Ticks)
   result    = CMDRESULT_OK (other values indicate an error)
```

```
          echoed command =
          startScience: CMDOP_BIAS_TE
          {
             blockSlotIndex   = 0..4
          }
       }
```

Command Echo on a successful receipt of Continuous Clocking start request:

```
       commandEcho: TTAG_CMD_ECHO
       {
          arrival   = time command arrived (BEP Ticks)
          result    = CMDRESULT_OK (other values indicate an error)

          echoed command =
          startScience: CMDOP_BIAS_CC
          {
             blockSlotIndex   = 0..4
          }
       }
```

If a science run had been started, and not yet stopped, when the command is received, the "result" field of the echo will contain CMDRESULT_CLOBBERED. In this case, the earlier run will be aborted (NOTE: The FEPs will be reset), and the new run will be configured and started.

If the radiation monitor had been activated, then the start of the run will be deferred until the monitor is de-activated. In this case the "result" field will contain CMDRESULT_INHIBITED.

If the selected parameter block checksum does not match its contents (i.e. has been corrupted), the system will attempt to determine if the run was configured to use the Imaging CCDs or the Spectroscopy CCDs. If any of the CCDs in the fepSelect field of the parameter block choose an Imaging CCD (I0..I3), the system will attempt to use the parameter block contained in slot 0. If not, it will try to default to the parameter block in slot 1. If the default is not corrupt, the intent was for the system to use the default parameter block, and the "result" field will be set to CMDRESULT_CORRUPT_DEFAULT. Due to a "bug" in the software, however, the system will use the most recently configured parameter block for the type of run that was chosen, either Timed Exposure or Continuous Clocking (NOTE: Some might consider this a feature).

If the original parameter block is corrupt, and the selected default parameter block is also corrupt (or they're one and the same), the run will not be started, and the "result" field will contain CMDRESULT_CORRUPT_IDLE.

Since the field ranges within the parameter block are only checked during the setup stage of the run, after the receipt of the command has been acknowledged via the command echo, if a parameter contained in the block is invalid or unsupported, the Command Echo will not indicate the problem, and supply a

CMDRESULT_OK. Instead, the run will not be started, and a “Science Report” packet will indicate the problem within its terminationReason field.

At the start of the run, the Science Manager will fetch and telemeter the parameter blocks used for the run. In Timed Exposure Mode, the dump appears as:

```
dumpedTeBlock: TTAG_DUMP_TE
{
   dumped TE Block:
   loadTeBlock: CMDOP_LOAD_TE
   [
      ...
   }
   if windows were used, this is followed by:
   a dumped 2d Window Block, aligned to the next 32-bit word:
   load2dBlock: CMDOP_LOAD_2D
   {
      ...
   }
}
```

In Continuous Clocking Mode, the dump appears as:

```
dumpedCcBlock: TTAG_DUMP_CC
{
   dumped CC Block:
   loadCcBlock: CMDOP_LOAD_CC
   [
      ...
   }
   if windows were used, this is followed by:
   a dumped 1d Window Block, aligned to the next 32-bit word:
   load1dBlock: CMDOP_LOAD_1D
   {
      ...
   }
}
```

Once the parameter blocks have been dumped, the Science Manager acquires and telemeters the DEA Video Board housekeeping information for all CCDs used for the run (NOTE: Unless all 10 video boards are powered on, the analog values in the packet will be of marginal use. They will indicate if the given channel is completely broken, but won’t really indicate the level that the DAC was set to). The content and form of this information is described in Section 3.4.6.6 .

If the “trickleBias” flag was set in the parameter block, the system will telemeter the contents of the computed bias maps of each FEP, in FEP order (i.e. FEP_0 first, then FEP_1, etc.). The Timed Exposure Bias Map telemetry packets are of the form:

```
dataTeBiasMap: TTAG_SCI_TE_BIAS
{
   biasStartTime              = 100KHz time-stamp at start of bias
   biasParameterId            = Parameter Block Id used for bias
   ccdId                      = CCD which produced the map
   fepId                      = FEP used to produce the map
```

```
      dataPacketNumber          = packet sequence number within this map
      initialOverclocks[4]      = DC-offset of bias values, by quadrant
      pixelsPerRow              = number of pixels in each bias map row
      rowsPerBias               = number of rows in the map
      ccdRow                    = starting row of data in the packet
      ccdRowCount               = number of rows packed into the packet
      compressionTableSlotIndex = table slot value used to compress data
      pixelCount                = total number of pixels in the packet
      data[]                    = packed or compressed data, as array of
                                  32-bit words
   }
```

The Continuous Clocking Bias Map telemetry packets are of the form:

```
   dataCcBiasMap: TTAG_SCI_CC_BIAS
   {
      biasStartTime             = 100KHz time-stamp at start of bias
      biasParameterId           = Parameter Block Id used for bias
      ccdId                     = CCD which produced the map
      fepId                     = FEP used to produce the map
      data[1024]                = packed 12-bit pixel bias values
   }
```

NOTE:If column summing is used in Continuous Clocking, the bias values will not fill the “data” array, leaving garbage in the unused portion.

Once the bias is complete, the Science Manager terminates the run and produces a “Science Report” telemetry packet. Under normal circumstances, the “terminationReason” field will contain a SMTERM_BIASDONE.

```
   scienceReport: TTAG_SCI_REPORT
   {
      runStartTime          = will always be 0xffffffff
      parameterBlockId      = Parameter Block Id (see biasParameterId)
      windowBlockId         = Window Block Id specified in block
      biasStartTime         = 100KHz time-stamp when bias was computed
      biasParameterId       = Parameter Block Id used for bias
      exposuresProduced     = will always be 0
      exposuresSent         = will always be 0
      biasErrorCount        = will always be 0
      fepErrorCodes[6]      = FEP Error codes, indexed by FEP
      ccdErrorFlags[6]      = Video Board error flags, indexed by FEP
      deaInterfaceErrorFlag = DEA Interface board error flag
      terminationCode       = SMTERM_BIASDONE
   }
```

Refer to Section 4.1.6 Science Telemetry for a description of the various error flags and other termination reason codes.

Warnings

1. See Section 4.1.5 Warnings
2. See Section 3.4.6.6 Warnings
3. CC Bias data array may contain some garbage

The Continuous Clocking bias map telmetry always consists of a packed array of 1024, 12-bit elements. If column summing is used in Continuous Clocking mode, the unused portion will contain garbage.

5.0 Example Scenarios

This section illustrates how some of the features of the instrument may be used in combination to perform various procedures. such as restarting the instrument after a power-off. The actual procedures used in-flight will probably be different, due to system requirements beyond that of the software, or due to experience developed as the system is used.

5.1 On-Orbit Checkout

The purposes of the on-orbit check-out is to verify that none of the hardware broke during launch and on-orbit insertion.

TBD - Talk to Bob and get a better picture of what should go here. Maybe variation of short-form, long-form?

1. Pre-Power On check of passive analog telemetry and S/C conditions
2. DPA Side-A Power On
3. DPA Voltage, Current and Thermal Check
4. Select BEP A
5. Cold Boot BEP (prior to cold boot after DEA power-on. If a problem arose in commanding DEA after latter cold boot, ya might not know if it was the cold-boot, or the DEA interface which killed ya).
6. Some form of Command Check (e.g. send a "no-op" software command and check that ya gets an echo).
7. ROM Dump and Compare (does a good telemetry check)
8. System Configuration Dump (verify temperature set-points prior to DEA power-on, etc.)
9. FEP Power On Check
10. DPA Voltage, Current, and Thermal Check
11. FEP I-cache Read Check (more to test FEP aliveness than anything else)
12. DEA Side-A Power On
13. DPA/DEA Voltage, Current and Thermal Check
14. Cold-Boot BEP (to load System Configuration Info into newly powered DEA board)
15. Start interface board DEA Housekeeping
16. Check DEA housekeeping info (including relay states)
17. ...

5.2 Instrument Restart after a Power-Off

When a BEP is powered off (i.e. its corresponding power-supply side is turned off or is disabled), it loses its patch list. When the power is re-applied, the bad pixel and column maps, system configuration table, huffman table modifications, and all parameter blocks are overwritten by those stored in ROM.

The italicized checks may be performed out-of-sequence (i.e. not in real-time) if the hardware and the procedure have been thoroughly tested in-flight, and the spacecraft is not in ground contact when the power-on procedure needs to run.

5.2.1 Normal - BEP A and B powered, BEP A Selected

The normal use of the system is to have both BEP A and B powered, allowing use of all 6 FEPs, and BEP A as the selected BEP.

1. If DPA Side-B is on, select B, power off FEPs, wait 10 seconds, select A (Section 3.1.2 , Section 3.3.2)
2. If DPA Side-B is off, enable DPA Side B, power on DPA Side B (Section 3.1.1)
3. Power On DPA Side A (Section 3.1.1)
4. *Check Voltage, Current, and Thermal (Section 3.4.1)*
5. *Check BEP Startup Message (Section 3.4.2)*
6. Re-load patch list (Section 3.7.1.1)
7. Dump patch list *and check (Section 3.6.2.4)*
8. Ensure the desired DEA side is on (Section 3.3.3)
9. Warm Boot (Section 3.1.3.3)
10. *Check BEP Startup Message (look at version) (Section 3.4.2)*
11. Re-load System Configuration parameters (especially focal plane temperatures) (Section 3.3)
12. *Check Voltage, Current and Thermal since video boards and FEPs may be powered, and the focal plane temperature set-point will be modified (Section 3.4.1)*
13. Dump System Configuration table *and check (Section 3.6.2.5)*
14. Re-load Bad Pixel and Column Maps (Section 3.5)
15. Re-load static DEA, TE, CC, 2D, and 1D parameter blocks (i.e. those not automatically re-loaded at start of each run) (Section 3.4.6.1 , Section 4.1.4)
16. Dump Bad Pixel and Column maps, parameter block slots, and Huffman table *and check (Section 3.6.2.1 , Section 3.6.2.3 , Section 3.6.2.2)*
17. Re-load Huffman table changes (if any) using Write-BEP commands (Section 3.7.2)

18. Wait longer than 8 minutes after step 9 (1 minute to power on boards, and 7 minutes for FEP time-stamps to synchronize), where the FEP power may have been re-applied
19. Start the first observation, ensuring that bias-maps are re-computed and telemetered (Section 4.1)

5.2.2 Contingency/Test - BEP A and B powered, BEP B Selected

One contingency or test scenario is to have both BEP A and B powered, allowing use of all 6 FEPs, but using BEP B as the selected BEP.

1. If DPA Side-A is on, select A, power off FEPs, wait 10 seconds, hold reset A (Section 3.1.2 , Section 3.3.2 , Section 3.2.2)
2. If DPA Side-A is off, enable DPA Side A, power on DPA Side A, hold reset A (Section 3.1.1 , Section 3.2.2)
3. Power On DPA Side B (reset A is still driving the interfaces, should see 0xb7 in software telemetry stream) (Section 3.1.1)
4. *Check Voltage, Current, and Thermal (Section 3.4.1)*
5. Select BEP B (BEP B is now driving the interfaces, packets should appear) (Section 3.1.2)
6. *Check BEP Startup Message (Section 3.4.2)*
7. Re-load patch list (Section 3.7.1.1)
8. Dump patch list *and check (Section 3.6.2.4)*
9. Ensure the desired DEA side is on (Section 3.3.3)
10. Warm Boot (Section 3.1.3.3)
11. *Check BEP Startup Message (look at version) (Section 3.4.2)*
12. Re-load System Configuration parameters (especially focal plane temperatures) (Section 3.3)
13. *Check Voltage, Current and Thermal since video boards and FEPs may be powered, and the focal plane temperature set-point will be modified (Section 3.4.1)*
14. Dump System Configuration table *and check (Section 3.6.2.5)*
15. Re-load Bad Pixel and Column Maps (Section 3.5)
16. Re-load Huffman table changes (if any) using BEP Write-Memory commands (Section 3.7.2)
17. Re-load static DEA, TE, CC, 2D, and 1D parameter blocks (i.e. those not automatically re-loaded at start of each run) (Section 3.4.6.1 , Section 4.1.4)
18. Dump Bad Pixel and Column maps, parameter block slots, and Huffman table *and check (Section 3.6.2.1 , Section 3.6.2.3 , Section 3.6.2.2)*
19. Wait longer than 8 minutes after step 9 (1 minute to power on boards, and 7 minutes for FEP time-stamps to synchronize), where the FEP power may have been re-applied

20. Start the first observation, ensuring that bias-maps are re-computed and telemetered (Section 4.1)

5.2.3 Contingency/Test - Only BEP A powered

If BEP B fails in a manner that requires that it not be powered, such as a short on the board, only 3 of the 6 FEPs may be used, and only BEP A may be used.

1. Power On DPA Side A (Section 3.1.1)
2. *Check Voltage, Current, and Thermal (Section 3.4.1)*
3. *Check BEP Startup Message (Section 3.4.2)*
4. Re-load patch list (Section 3.7.1.1)
5. Dump patch list *and check (Section 3.6.2.4)*
6. Ensure the DPA Side-B is off (Section 3.2.1)
7. Ensure the desired DEA side is on (Section 3.3.3)
8. Warm Boot (Section 3.1.3.3)
9. *Check BEP Startup Message (look at version) (Section 3.4.2)*
10. Re-load System Configuration parameters (especially focal plane temperatures) WARNING: Ensure that only FEPs 0, 1 and 2 are powered. FEPs 3, 4 and 5 are powered by the Side-B DPA power supply (Section 3.3)
11. *Check Voltage, Current and Thermal since video boards and FEPs may be powered, and the focal plane temperature set-point will be modified (Section 3.4.1)*
12. Dump System Configuration table *and check (Section 3.6.2.5)*
13. Re-load Bad Pixel and Column Maps (Section 3.5)
14. Re-load Huffman table changes (if any) using BEP Write-Memory commands (Section 3.7.2)
15. Re-load static DEA, TE, CC, 2D, and 1D parameter blocks (i.e. those not automatically re-loaded at start of each run) (Section 3.4.6.1 , Section 4.1.4)
16. Dump Bad Pixel and Column maps, parameter block slots and Huffman table *and check (Section 3.6.2.1 , Section 3.6.2.3 , Section 3.6.2.2)*
17. Wait longer than 8 minutes after step 9 (1 minute to power on boards, and 7 minutes for FEP time-stamps to synchronize), where the FEP power may have been re-applied
18. Start the first observation, ensuring that bias-maps are re-computed and telemetered. NOTE: FEPs 3, 4, and 5 cannot be used for science, since their corresponding power supply is off. Ensure that the primary CCDs of interest are configured for FEPs 0, 1, or 2. (Section 4.1)

5.2.4 Contingency/Test - Only BEP B powered

If BEP A fails in a manner that requires that it not be powered, such as a short on the board, only 3 of the 6 FEPs may be used, and only BEP B may be used.

1. Ensure the desired DEA side is on (Section 3.3.3)
2. Power On DPA Side B (no board will be driving the interface at this point) (Section 3.1.1)
3. *Check Voltage, Current, and Thermal (Section 3.4.1)*
4. Select BEP B (BEP will now start driving the interfaces) (Section 3.1.2)
5. *Check BEP Startup Message (Section 3.4.2)*
6. Re-load patch list (Section 3.7.1.1)
7. Dump patch list *and check (Section 3.6.2.4)*
8. Ensure the DPA Side-A is off (Section 3.2.1)
9. Warm Boot (Section 3.1.3.3)
10. *Check BEP Startup Message (look at version) (Section 3.4.2)*
11. Re-load System Configuration parameters (especially focal plane temperatures) WARNING: Ensure that only FEPs 3, 4 and 5 are powered. FEPs 0, 1 and 2 are powered by the Side-A DPA power supply (Section 3.3)
12. *Check Voltage, Current and Thermal since video boards and FEPs may be powered, and the focal plane temperature set-point will be modified (Section 3.4.1)*
13. Dump System Configuration table *and check (Section 3.6.2.5)*
14. Re-load Bad Pixel and Column Maps (Section 3.5)
15. Re-load Huffman table changes (if any) using BEP Write-Memory commands (Section 3.7.2)
16. Re-load static DEA, TE, CC, 2D, and 1D parameter blocks (i.e. those not automatically re-loaded at start of each run) (Section 3.4.6.1 , Section 4.1.4)
17. Dump Bad Pixel and Column maps, parameter block slots and Huffman table *and check (Section 3.6.2.1 , Section 3.6.2.3 , Section 3.6.2.2)*
18. Wait longer than 8 minutes after step 9 (1 minute to power on boards, and 7 minutes for FEP time-stamps to synchronize), where the FEP power may have been re-applied
19. Start the first observation, ensuring that bias-maps are re-computed and telemetered. NOTE: FEPs 0, 1, and 2 cannot be used for science, since their corresponding power supply is off. Ensure that the primary CCDs of interest are configured for FEPs 3, 4, or 5 (Section 4.1)

5.3 “Warm” Instrument Re-start

The user performs “warm” boots of the instruments either after removing or installing patches, or in order to “fix” a software or hardware problem by gently re-starting the BEP. The procedure used to perform a warm re-boot is as follows:

1. Ensure that the desired DPA side is on (Section 3.1.1)
2. Ensure the desired DEA side is on (Section 3.3.3)
3. *Optionally dump the patch list and check (Section 3.6.2.4)*
4. Perform a Warm Boot (Section 3.1.3.3)
5. *Check BEP Startup Message (look at version) (Section 3.4.2)*
6. *Check Voltage, Current and Thermal since video boards will be power-cycled, the powered FEPs will be reset, and the focal plane temperature set-point will be re-loaded (Section 3.4.1)*
7. Dump System Configuration table *and check (Section 3.6.2.5)*
8. Dump Bad Pixel and Column maps, parameter block slots and Huffman table *and check (Section 3.6.2.1 , Section 3.6.2.3 , Section 3.6.2.2)*
9. Wait longer than 20 seconds for the video boards to be power-cycled.
10. Start the first observation, ensuring that bias-maps are re-computed and telemetered (Section 4.1)

5.4 Timed Exposure Science Run with Imaging CCDs

This section provides an example Timed Exposure Science Run.

5.4.1 Parameter Block Setup

This section describes the example set of parameters and reasons why the parameters were chosen. This section follows the field descriptions provided in Section 4.1.4.1 .

5.4.1.1 Parameter Block Id

For the purposes of this example, assume that the user assigns a unique parameter block id of 0x635231a2 to the paramter block.

5.4.1.2 CCD Selection

This example uses all 4 Imaging CCDs and the two center spectrosocpy CCDs, of which CCD_S3 is a back-side CCD. According to the CCD-lab, the backside CCDs have marginally better performance if their images are held in the CCD framestore for longer periods of time. In order to take advantage of this, CCD_S3 is configured for the first FEP (see Section 4.1.1.2).

The FEP selection array is configured as:

```
fepCcdSelect[FEP_0] := CCD_S3
fepCcdSelect[FEP_1] := CCD_I0
fepCcdSelect[FEP_2] := CCD_I1
fepCcdSelect[FEP_3] := CCD_I2
fepCcdSelect[FEP_4] := CCD_I3
fepCcdSelect[FEP_5] := CCD_S2
```

5.4.1.3 Mode Selection

For this example, assume that the source is expected to produce no more than about 80 events per second (including background events which “sneak” through the filters) across all CCDs, and that we don’t want to spend the time to send the entire bias map. The event rate for this example is low enough to support Timed Exposure Faint with bias mode, so we choose:

```
fepMode := FEP_TE_MODE_EV3x3
bepMode := BEP_TE_MODE_FAINTBIAS
```

5.4.1.4 Clocking Parameters

For this example, we’re not doing on-chip summing, not using a subarray, using the maximum of 30 overclocks per output node, and using a normal video gain. Assume that all of the video chains are working and will be used. The minimum, normal integration time for 6 CCDs is 3.3 seconds (see Section 4.1.1.2). For this example, assume that the source is diffuse enough and the spacecraft dither rate is fast enough to avoid pileup with the 80 counts/second, given a 3.3 second static integration time. This leads to the following:

```
primaryExposure     := 33
secondaryExposure   := 0
dutyCycle           := 0
onChip2x2Summing    := 0
subarrayStartRow       := 0
subarrayRowCount       := 1023
overclockPairsPerNode  := 15
outputRegisterMode     := QUAD_FULL
ccdVideoResponse[FEP_0]:= 0
ccdVideoResponse[FEP_1]:= 0
ccdVideoResponse[FEP_2]:= 0
ccdVideoResponse[FEP_3]:= 0
ccdVideoResponse[FEP_4]:= 0
ccdVideoResponse[FEP_5]:= 0
```

5.4.1.5 Video ADC Offsets

Each video board channel has a characteristic video offset value. The offset value is tuned to provide a decent dynamic range, while avoiding clipping the measurment below zero or above 4095. These offsets may be adjusted by the maintainers over the life of the mission. The video offsets provided in the parameter blocks in ROM, and used during calibration are provided in TBD. The offset parameters are indexed by the FEP using the particular CCD, so the mapped parameters are as follows:

```
CCD_S3
fep0VideoOffset[ONODE_A] := 79
fep0VideoOffset[ONODE_B] := 79
fep0VideoOffset[ONODE_C] := 79
fep0VideoOffset[ONODE_D] := 77
CCD_I0
fep1VideoOffset[ONODE_A] := 87
fep1VideoOffset[ONODE_B] := 86
fep1VideoOffset[ONODE_C] := 76
fep1VideoOffset[ONODE_D] := 89
CCD_I1
fep2VideoOffset[ONODE_A] := 83
fep2VideoOffset[ONODE_B] := 69
fep2VideoOffset[ONODE_C] := 79
fep2VideoOffset[ONODE_D] := 83
CCD_I2
fep3VideoOffset[ONODE_A] := 86
fep3VideoOffset[ONODE_B] := 65
fep3VideoOffset[ONODE_C] := 82
fep3VideoOffset[ONODE_D] := 89
CCD_I3
fep4VideoOffset[ONODE_A] := 76
fep4VideoOffset[ONODE_B] := 68
fep4VideoOffset[ONODE_C] := 79
fep4VideoOffset[ONODE_D] := 80
CCD_S2
fep5VideoOffset[ONODE_A] := 90
fep5VideoOffset[ONODE_B] := 86
fep5VideoOffset[ONODE_C] := 79
fep5VideoOffset[ONODE_D] := 94
```

5.4.1.6 Bias Parameters

For this example, the observation is going to re-compute the bias, but not send it. The algorithm will use “Whole-Frame” mode, with 5 conditioning exposures followed by 11 approximation-to-mean exposures. The low-pixel elimination feature will not be used. The event rejection threshold for the backside CCD, CCD_S3, will be 26 ADU above the the approximated bias level, and will be 50 ADU for all of the front-side CCDs. The pixel rejection threshold for the approximation-to-mean stage will be maintained at 20 ADU above the current bias approximation for all of the CCDs. In order to allow the video board temperatures to settle, and to effectively flush the charge from the CCDs, the system will ignore the first 30 exposures prior to starting the bias computation. The current bad pixel and Timed Exposure column lists will be applied to the bias maps once the maps have been computed. This leads to the following parameters:

```
ignoreBadPixelMap      := 0
ignoreBadColumnMap     := 0
recomputeBias          := 1
trickleBias            := 0
ignoreInitialFrames    := 30
CCD_S3
biasAlgorithmId[FEP_0] := FEP_BIAS_1
biasArg0[FEP_0]        := 5
biasArg1[FEP_0]        := 16
biasArg2[FEP_0]        := 0
biasArg3[FEP_0]        := 26
biasArg4[FEP_0]        := 20
biasCompressionSlotIndex[FEP_0]  := 255
CCD_I0
biasAlgorithmId[FEP_1] := FEP_BIAS_1
biasArg0[FEP_1]        := 5
biasArg1[FEP_1]        := 16
biasArg2[FEP_1]        := 0
biasArg3[FEP_1]        := 50
biasArg4[FEP_1]        := 20
biasCompressionSlotIndex[FEP_1]  := 255
CCD_I1
biasAlgorithmId[FEP_2] := FEP_BIAS_1
biasArg0[FEP_2]        := 5
biasArg1[FEP_2]        := 16
biasArg2[FEP_2]        := 0
biasArg3[FEP_2]        := 50
biasArg4[FEP_2]        := 20
biasCompressionSlotIndex[FEP_2]  := 255
CCD_I2
biasAlgorithmId[FEP_3] := FEP_BIAS_1
biasArg0[FEP_3]        := 5
biasArg1[FEP_3]        := 16
biasArg2[FEP_3]        := 0
biasArg3[FEP_3]        := 50
biasArg4[FEP_3]        := 20
biasCompressionSlotIndex[FEP_3]  := 255
CCD_I3
biasAlgorithmId[FEP_4] := FEP_BIAS_1
biasArg0[FEP_4]        := 5
biasArg1[FEP_4]        := 16
biasArg2[FEP_4]        := 0
biasArg3[FEP_4]        := 50
```

```
biasArg4[FEP_4]         := 20
biasCompressionSlotIndex[FEP_4]  := 255
CCD_S2
biasAlgorithmId[FEP_5]  := FEP_BIAS_1
biasArg0[FEP_5]         := 5
biasArg1[FEP_5]         := 16
biasArg2[FEP_5]         := 0
biasArg3[FEP_5]         := 50
biasArg4[FEP_5]         := 20
biasCompressionSlotIndex[FEP_5]  := 255
```

5.4.1.7 Event Selection Parameters

For this example, the Front End Processors will select a candidate event center if the center pixel values are 20 ADU above their bias levels for the back-side CCD (CCD_S3), and 38 ADU for the remaining front-side CCDs. The event's split threshold is 13 ADU above the pixel bias values for all of the CCDs. In this example, the system will only accept events whose grade codes map to ASCA grades 0, 2, 3, 4 and 6 (unsplit, vertical splits, left and right horizontal splits, and L-square corner splits). No window filters or pulse height filters will be applied.

This leads to the following event selection parameters:

```
lowerEventAmplitude                 := 0
eventAmplitudeRange                 := 65535
gradeSelections[0,2,8,10,11,12,16,17,
18,22,34,48,49,50,54,64,65,68,69,72,76,
80,81,104,108,130,136,138,140,162,208,
209]                                := 1
all other gradeSelections[]         := 0
windowSlotIndex                     := 255 (no windows)
CCD_S3
fep0Threshold[ONODE_A..ONODE_D]      := 20
fep0SplitThreshold[ONODE_A..ONODE_D] := 13
CCD_I0
fep1Threshold[ONODE_A..ONODE_D]      := 38
fep1SplitThreshold[ONODE_A..ONODE_D] := 13
CCD_I1
fep2Threshold[ONODE_A..ONODE_D]      := 38
fep2SplitThreshold[ONODE_A..ONODE_D] := 13
CCD_I2
fep3Threshold[ONODE_A..ONODE_D]      := 38
fep3SplitThreshold[ONODE_A..ONODE_D] := 13
CCD_I3
fep4Threshold[ONODE_A..ONODE_D]      := 38
fep4SplitThreshold[ONODE_A..ONODE_D] := 13
CCD_S2
fep5Threshold[ONODE_A..ONODE_D]      := 38
fep5SplitThreshold[ONODE_A..ONODE_D] := 13
```

5.4.1.8 Parameter Block Summary

The following is the resulting parameter block, where the grade selection bit-map has been expressed as a series of 32-bit words:

```
loadTeBlock: CMDOP_LOAD_TE
{   teBlockSlotIndex       := not-yet specified
    checksum               := 65458
    parameterBlockId       := 0x635231a2
    fepCcdSelect[FEP_0..FEP_5] := CCD_S3, CCD_I0, CCD_I1, CCD_I2,
                                  CCD_I3, CCD_S2
    fepMode                := FEP_TE_MODE_EV3x3
    bepPackingMode         := BEP_TE_MODE_FAINTBIAS
    onChip2x2Summing       := 0
    ignoreBadPixelMap      := 0
    ignoreBadColumnMap     := 0
    recomputeBias          := 1
    trickleBias            := 0
    subarrayStartRow       := 0
    subarrayRowcount       := 1023
    overclockPairsPerNode  := 15
    outputRegisterMode     := FULL
    ccdVideoResponse[FEP_0..FEP_5]:= 0,0,0,0,0,0
    primaryExposure        := 33
    secondaryExposure      := 0
    dutyCycle              := 0
    fep0EventThreshold[ONODE_A..ONODE_D]:= 20,20,20,20
    fep1EventThreshold[ONODE_A..ONODE_D]:= 38,38,38,38
    fep2EventThreshold[ONODE_A..ONODE_D]:= 38,38,38,38
    fep3EventThreshold[ONODE_A..ONODE_D]:= 38,38,38,38
    fep4EventThreshold[ONODE_A..ONODE_D]:= 38,38,38,38
    fep5EventThreshold[ONODE_A..ONODE_D]:= 38,38,38,38
    fep0SplitThreshold[ONODE_A..ONODE_D]:= 13,13,13,13
    fep1SplitThreshold[ONODE_A..ONODE_D]:= 13,13,13,13
    fep2SplitThreshold[ONODE_A..ONODE_D]:= 13,13,13,13
    fep3SplitThreshold[ONODE_A..ONODE_D]:= 13,13,13,13
    fep4SplitThreshold[ONODE_A..ONODE_D]:= 13,13,13,13
    fep5SplitThreshold[ONODE_A..ONODE_D]:= 13,13,13,13
    lowerEventAmplitude    := 0
    eventAmplitudeRange    := 65535
    gradeSelections[0..255]:= 0x00471d05, 0x00470004, 0x00031133,
                              0x00001100, 0x00001d04, 0x00000004,
                              0x00030000, 0x00000000
    windowSlotIndex        := 255
    histogramCount         := 0
    biasCompressionSlotIndex[FEP_0..FEP_5] := 255,255,255,255,255,255
    rawCompressionSlotIndex:= 255
    ignoreInitialFrames    := 30
    biasAlgorithmId[6]     := FEP_BIAS_1, FEP_BIAS_1, FEP_BIAS_1,
                              FEP_BIAS_1, FEP_BIAS_1, FEP_BIAS_1
    biasArg0[FEP_0..FEP_5] := 5,5,5,5,5,5
    biasArg1[FEP_0..FEP_5] := 16,16,16,16,16,16
    biasArg2[FEP_0..FEP_5] := 0,0,0,0,0,0
    biasArg3[FEP_0..FEP_5] := 26,50,50,50,50,50
    biasArg4[FEP_0..FEP_5] := 20,20,20,20,20,20
    fep0VideoOffset[ONODE_A..ONODE_D]:= 79,79,79,77
    fep1VideoOffset[ONODE_A..ONODE_D]:= 87,86,76,89
    fep2VideoOffset[ONODE_A..ONODE_D]:= 83,69,79,83
    fep3VideoOffset[ONODE_A..ONODE_D]:= 86,65,82,89
```

```
      fep4VideoOffset[ONODE_A..ONODE_D]:= 76,68,79,80
      fep5VideoOffset[ONODE_A..ONODE_D]:= 90,86,79,94
      deaLoadOverride        := 0
      fepLoadOverride        := 0
   }
```

5.4.2 Initial Instrument State

This example assumes that both BEPs are powered, that the Side-A DEA Power supply has power, that all the FEPs have power (and were powered for over 7 minutes), but that the state of the video board power is unknown to the ground scheduling algorithm. This example also assumes that all of the hardware and software is operating properly, that the system configuration parameters (other than the video board power) are in a stable, known state, and that the focal plane temperature is at its normal operating temperature (i.e. about -120^{o}C).

5.4.3 Switch Video Board Power

The user first issues a command which powers off video boards not used for the run, and powers on the boards required for the run. In this example, CCDs I0 - I3 and S2 and S3 are to be used. This leads to the following “Change System Configuration” command (see Section 3.3.3):

```
   changeConfigSetting: CMDOP_CHANGE_SYS_ENTRY
   {
      entries[] =
      {
         itemId    =  SYSSET_DEA_POWER
         itemValue =  0x0cf
      }
   }
```

If all of the board power settings were changed by the command, the command will take about 11 seconds to execute. If all of the boards were already in their desired state, the command will take less than 1 second to execute.

5.4.4 Load Parameter Block

For this example, the parameter block will be loaded into Timed Exposure slot 3, by setting the *teBlockSlotIndex* in the parameter set shown in Section 5.4.1.8 to a value of 3, and issuing the command. Since no windows are being used, no window parameter block command is needed for the run. Upon receipt of the command, the block will be loaded in under 1 second (see Section 4.1.4.1). The instrument is now ready to start a run.

5.4.5 Start Run

The user starts the run using a “Start Science” command, specifying slot 3 (see Section 4.1.5):

```
startScience: CMDOP_START_TE
{
   blockSlotIndex   = 3
}
```

5.4.6 Setup Time

Given that 6 CCDs are being configured and the PRAM/SRAM load is a standard load, the system will take on the order of 2 minutes to configure the system, acquire and dump the DEA housekeeping information, and start clocking the CCDs:

```
setup time   := 30 seconds to load video registers +
                1 minute to acquire and dump DEA housekeeping +
                11 seconds to flush CCD charge +
                ~1 second of additional overhead
             ~= 1.7 minutes
```

5.4.7 Bias Calculation

Once the system is configured, the FEPs will be commanded to re-compute the bias. Assuming that the whole-frame algorithm keeps up with the incoming images, the bias computation will take about 3 minutes:

```
bias time := (30 ignored frames + 16 bias frames) * 3.3 seconds/frame
          := 2.53 minutes
```

5.4.8 Bias Telemetry

For this example, the bias maps are not telemetered.

5.4.9 Data Processing

After the bias maps have been computed, the instrument will stop the sequencers, prepare the FEPs for data processing, and re-start the CCD sequencers. The FEPs will automatically discard the first two exposures, and then start data processing. The first exposure records will appear about 10 seconds after data processing starts (3.3 seconds/frame * 2 ignore frames + 1 exposure frame). Every 3.3 seconds, each FEP will produce zero or more Faint-with-Bias data packets followed by 1 Faint-with-Bias exposure record. The data packets and exposure records from the various FEPs are interleaved in time. If the event rate climbs such that one or more of the FEPs gets backlogged, their data packets and exposure records may be sent later than those for FEPs which are not backlogged.

NOTE: If the system’s buffers fill, some FEPs may drop exposures, whereas others might not.

5.4.10 Stop Run

Once the desired observation time has elapsed, the user stops the run by issuing a “Stop Science” command packet:

```
stopScience: CMDOP_STOP_SCIENCE
{
   No additional parameters are required
}
```

After receiving the command, the system will command the FEPs to finish up their current exposures.

5.4.11 Draining Telemetry

Once the FEPs have been instructed to stop, the BEP will poll the FEPs until they’ve finished their current exposure, and drained their FEP to BEP ring-buffers. In this example, the event rate doesn’t exceed the telemetry rate, and therefore, the ring-buffers and telemetry buffers are fairly empty. If the last image had just started arriving at the FEPs when they were instructed to stop, the image transfer time will dominate the time to finish the run. For this example, this is about 3.3 seconds.

5.4.12 Run Complete

Once the run completes, the BEP will produce a “Science Report” telemetry packet. Assume for this example, that the observation was scheduled for about 4 hours, that nothing went wrong, and that telemetry was never saturated (i.e. no frames were dropped). Under these conditions, the “Science Report” might look as follows:

```
scienceReport: TTAG_SCI_REPORT
{
   runStartTime           = 0x054379a2 (~biasStartTime + 2.5 minutes)
   parameterBLockId       = 0x635231a2
   windowBlockId          = 0xffffffff
   biasStartTime          = 0x045bd8c2
   biasParameterId        = 0x635231a2 (same as parameterBlockId)
   exposuresProduced      = 4364 (largest exposure number @ ~4 hours @
                            3.3 seconds/frame)
   exposuresSent          = 26166 (first 2 exposures are never sent, 6
                            FEPs sending)
   biasErrorCount         = 0
   fepErrorCodes[6]       = FEP_CMD_NOERR, FEP_CMD_NOERR,
                            FEP_CMD_NOERR, FEP_CMD_NOERR,
                            FEP_CMD_NOERR, FEP_CMD_NOERR
   ccdErrorFlags[6]       = 0,0,0,0,0,0
   deaInterfaceErrorFlag  = 0
   terminationCode        = SMTERM_STOPCMD
}
```

Appendix A ACIS Power Table

The following table lists the power consumed by ACIS due to its various components. This table was derived (stolen) from information gathered by Bob Goeke from the results of a series of Short Form and Long Form tests performed at Lincoln Laboratory (NOTE: This is taken from a note dated 19 May 1997, with some corrections from Ellen Sen on Sept. 9, 97). All the numbers shown include losses in the PSMC (order of 20% total):

TABLE 17. ACIS Power Table

State	**Side-A**	**Side-B**	**A or B**	**Summary**
PSMC Overhead	3w	3w		Total Overhead := 6w
BEP On and Running	14w	10w		Total BEP := 24w
2 FEPs On	-	-	16w	
3 FEPs + 3 FEPs On	25w	25w		Total FEP := 50w
All BEPs and FEPs On (idle)				**Total Static DPA := 74w**
DEA 11th Board On	24w			
DEA 12th Board On		18w		
2 Video Boards On (idle)			7w	
6 Video Boards On (idle)			21w	
DEA 11th or 12th and 6 Video Boards On (idle)				**Total Static DEA := 45w**
Additional power with 3 + 3 FEPs processing images	1w	1w		
Additional power with 6 Video Boards Clocking			8w	
Total Additional Power while clocking 6 CCDs				**Extra for clocks := 10w**
Additional power of remaining 4 video boards are powered (idle)			15w	
Focal Plane Heater Operating at Nominal Cold (-120C)			2w/6w peak	
FP Heater Bakeout (30C)			52w peak	
Detector Housing Heater Operating at nominal cold case (-60C)			8w/72w peak	
Detector Housing Heater Bakeout (25C)			72w peak	
Totals:				
Normal, clocking ACIS				**145w**
2 CCD Calibration				**90w**

TABLE 17. ACIS Power Table

State	Side-A	Side-B	A or B	Summary
Standby while maintaining thermal control				**54w**
Memory Save Mode (no thermal control)				**20w**
Bakeout slewing to temperature				**168w**
Bakeout at stable temperature (approx.)				**128w**
All ACIS On (not useful, but possible)				**276w**

The following table indicates the total current draw by ACIS, from the perspective of the service from the spacecraft:

TABLE 18. Total ACIS Current Draw

Voltage	Current (nominal)	Current (max)
30VDC	4.8A	9.2A
22VDC	6.6A	12.5A

Note that 10.8A is the maximum one could draw from one service, A or B

Appendix B - Active Patch List

Table 19 lists the patches active at the time this document was last updated. The standard patches consist of bug fixes and systems changes which are needed for the smooth operation of the instrument. The developers strongly recommend that the standarad patches are always loaded into the instrument during normal science operations. Refer to the latest release of MIT 36-58010 for a list and description of these patches.

New features and modes are provided in the suite of optional patch loads. Each of these loads is verified in combination with the latest standard patch load, and some of the engineering patches needed to access the test hardware. By default, the optional patches are NOT tested in combination. Refer to the latest version of MIT 36-58020 for a list and description of the set of available optional patches.

In order to manage testing of combinations of optional patches, separate verification tests are run on requested combinations of optional patches (always with the standard patch load in place). These combinations are provided distinct part numbers. The development team has not officially tested unlisted combinations of optional patches.

In order to support testing, debugging, and prototypes of future patch concepts, there are also a collection of engineering/prototype patches. These patches are not indended for normal operations.

TABLE 19. Current Patches

Type	Patch	MIT Part #	Revision	Size (bytes)
Standard	biastiming	36-58030.04	A	Total Standard Load Size = 472
36-58010 Rev. A	corruptblock	36-58030.01	A	
	digestbiaserror	36-58030.02	A	
	histogramvar	36-58030.03	A	
	rquad	36-58030.14	A	
	histogrammean	36-58030.15	A	
	zap1expo	36-58030.16	A	
Optional	cc3x3	36-58030.06	A	4636
36-58020 Rev. A	eventhist	36-58030.05	A	6356
	teignore	36-58030.09	A	36
	ccignore	36-58030.10	A	36
Optional Combos	cc3x3 + eventhist	36-58021.01	A	4636 + 6356
Engineering/ Prototypes	hybrid	36-58030.13	03	6104
	tlmio	36-58030.07	02	10312
	printswhouse	36-58030.08	01	4668
	deaeng	36-58030.11	02	2604
	dearepl	36-58030.12	02	556

Appendix C - Issue Summary

Fusing Issue

System Startup Times

Science Run Startup Times

Bias Telemetry Time

Telemetry Buffer Drain Time @24Kbps

Telemetry Buffer Drain Time @500bps

Simultaneous Power to DEA Side-A and Side-B

FEP Power “OR-d” by BEPs

Overlapping Bias Trickle with Data Processing (t-plane lockup)

Bias Drift without Jitter DAC patch

Bias Drift with Jitter DAC patch

Science Run Stop-Stop Problem

Histogram Mode when using AC or BD clocking option

DEA Video Board DAC System Configuration Limits

DEA Housekeeping Reporting 0xffff during Science Setup

Unix-Only: Version Number in SW Housekeeping’s first packet

Missing Clocking Combinations

Incomplete Unit Tests

Out-dated Detailed Design

Incomplete User’s Guide

DEA Video ADC Latchup

Command Noise

DEA Video Board Housekeeping with fewer than 10 boards on

Items To Be Completed or Determined