Alpha Release Validation Test Procedures

James E. Francis

MIT 64-30130.12

Rev. A

October 27, 1993

Contents

1 Overview

1.1 Scope

This document describes the set of tests and procedures that are used to validate the Alpha Release of the XTE-EDS Flight Software. The tests and procedures defined by this document satisfy the requirements described in the **Alpha Release Validation Test Requirements**.

1.1.1 References

MIT 64-02006.01 .... "XTE EDS Hardware Functional Description and Requirements" Rev. E
MIT 64-30102 .... "Flight Software Global Design" Rev. F
MIT 64-02019.1001 .... "Alpha Release Validation Test Requirements" Rev. A
MIT 64-30130 .... "Flight Software Executable Image"

1.2 Identification

All components of the Alpha Release Validation Tests, including tools, scripts and reports are either provided in or are identified by this document.

1.3 Overview of Tests

The Alpha Release Validation are grouped as follows:

a. Burst Trigger Signal Tests

   1. Verify that Call-Subroutine can properly invoke FREGset_bit() and FREGclr_bit(), including argument passing. Ensure that FREGset_bit() and FREGclr_bit() can properly assert and deassert the High and Low Priority Trigger signals using this invocation mechanism.

   2. Verify that Call-Subroutine can properly invoke TIMINGtrigger_hexte() and that the TIMING Module properly synthesizes a HEXTE Output Trigger pulse under 100ms and greater than 31ms.

   3. Verify that Burst Catcher Mode properly detects the High Priority, Low Priority and HEXTE Input Burst Trigger Signals and correctly indicates the detection via the telemetry stream.

b. PCA Transparent Configuration Test

  1. Verify the configurations for Transparent Modes 1, 2 and 3 and the corresponding event path from the PCA simulator to the telemetry stream.

c. PCA Binned Mode Performance Test

  1. Verify that Binned Mode correctly process all input event patterns.
  2. Verify that Binned Mode's transient event processing rate exceeds 500K events/second for up to 32ms.
  3. Verify that Binned Mode's continuous event processing rate exceeds 400K events/second.

d. ASM Data Test

  1. Verify that ASM Event by Event Mode correctly processes all input event patterns from each of the three SSCs.

e. ASM Performance Test

  1. Verify that Position Histogram Mode correctly processes all input event patterns.
  2. Verify that Position Histogram Mode continuous event processing rate exceeds 30K events/second.
  3. Verify that the ASM Housekeeper Thread properly obtains and telemeters ASM Housekeeping information corresponding to each dwell.

1.4 Environment

The Alpha Release Validation Tests are performed using the following equipment and software.

1.4.1 Hardware

The Alpha Release Validation Tests require the following hardware. The test report must contain the specific versions used during any given run of the tests (see Appendix A):

a. Event Analyzer Subgroup, including:

  1. 1 System Manager configured to be Side-B
  2. 3 PCA EAs

   3. 1 ASM EA

b. 1 MIT Spacecraft Simulator

c. 1 MIT PCA Event Simulator

d. 1 MIT ASM Instrument Simulator

e. 1 DecStation 5000 running Ultrix 4.3 or later and populated and configured with the following:

   1. DCT DMA Board (used to acquire telemetry from S/C Simulator) accessed via /dev/dct0

   2. TC-100 Parallel Interface Board (used to command PCA Event Simulator) accessed via /dev/pp0

   3. 2 RS-232 Serial Ports accessed via /dev/tty00 and /dev/tty01

f. 1 Oscilloscope, capable of measuring Burst Trigger Pulse widths

1.4.2 Software

The Alpha Release Validation Tests require the following software. The test report must contain the specific versions used during any given run of the tests (see Appendix A):

a. Alpha Release of the Flight Software (provided via the System Manager ROMs)

b. Flight Software Boot ROMs Makefile Version 1.7 or later (provided in each of the EAs Boot ROMs)

c. MIT GSE Command Server and Client

   1. command_server

   2. emit_command

d. MIT GSE Telemetry Program

   1. mit_telemetry

e. MIT GSE PCA Event Simulator Command Program

   1. pca_sim

f. MIT GSE ASM Event Simulator Command Program

   1. asm_sim

g. Command Builder Programs
   1. Command Packet Builder - fscmd
   2. Time-Delay Inserter - tpkt
h. Telemetry Parser Programs
   1. General Purpose Telemetry Monitor - watch_data
   2. Science Data Extraction - unpackScienceData
   3. Event Encoded Mode Event Count Program - eventSum
i. Command/Telemetry Aliases
   1. Command Utilities
      (a) Send Command - easend
      (b) Get Telemetry - eaget
   2. EA Discretes
      (a) Turn EAs Off - ea0off, ea1off, ea2off, ea3off
      (b) Uplink Boot EAs - ea0up, ea1up, ea2up, ea3up
      (c) ROM Boot EAs - ea0on, ea1on, ea2on, ea3on
   3. Serial Commands
      (a) Terminate - ea0term, ea1term, ea2term, ea3term
j. Alpha Release Flight Software Memory Map File
   1. fsrom.map

2 Burst Trigger Signal Tests

2.1 Scope

The Burst Trigger Signal Tests verify that the Flight Software has the necessary functionality needed to test the Burst Trigger signals during Hardware Verification Testing.

2.2 Identification

The procedure for these tests are completely identified and defined in this section. The results of these tests are reported in Appendix A.1.

2.3 Overview of Test

This section describes three tests:

a. Low-Priority Trigger Test - Verify that CallSub can be used to pulse low-priority trigger signal and that Burst Catcher Mode detects and indicates the signal.

b. High-Priority Trigger Test - Verify that CallSub can be used to pulse the high-priority trigger signal and that Burst Catcher Mode detects and indicates the signal.

c. HEXTE Output Trigger Test - Verify that CallSub can be used to pulse the HEXTE Output Trigger signal.

d. HEXTE Input Trigger Test - Verify that Burst Catcher Mode detects the HEXTE Input Burst Trigger signal and properly indicates the detection of the signal.

2.4 Global Setup

a. Setup hardware (seek assistance if needed)

b. Attach oscilloscope to the HEXTE Output Burst Trigger line and setup the scope to trigger on assertion of the line and to measure the width of the pulse. Expect the pulse-width to be greater than 10ms and less than 100ms.

c. Turn on power to Spacecraft Simulator and reset simulator

d. Start command server

   1. stty 9600 -even -odd pass8 raw > /dev/tty01

   2. command_server >& /dev/null&

e. Turn on power to System Manager

f. Zero PRAM Version and flag PRAM as invalid

   1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0 0 SEND" | easend

g. Turn all EAs on and waiting for uplink to boot (i.e. On but Quiet)

   1. ea0up; ea1up; ea2up; ea3up

h. In a window reserved for the telemetry monitor, start telemetry capture and monitor

   1. eaget | tee raw.pkts | watch_data -commands

i. Turn On an EA 1

   1. ea1on

j. Wait for Startup Message from the telemetry monitor

   1. [69 STARTUP :: PRAM Version = 0x00]

k. If the message above does not appear within 1/2 a minute, there is something wrong with the setup. Seek assistance.

l. Stop telemetry from EA 1

   1. ea1up

2.5 Test Case - Low Priority Trigger

2.5.1 Objective

The objective of this test is to ensure that the Flight Software Alpha Release can generate a pulse on the Low-Priority Inter-EA Burst Trigger Signal and that Burst Catcher properly detects and indicates the signal.

2.5.2 Estimate

This test takes under 30 minutes to run.

2.5.3 Test Execution

a. Find the flat address of Burst Catcher Configuration in the map file

   1. Search fsrom.map for "_romLibCatcherBinned". The first value after the symbol is the flat address of the configuration (note: it may not be on the same line). For the purposes of this script, we'll label this value <catcher>.

b. Find the flat address for FREGset_bit() and FREGclr_bit in the map file

   1. Search fsrom.map for "_FREGset_bit" and "_FREGclr_bit". The first value after the symbol is the flat address of the function (note: it may not be on the same line). For the purposes of this script, we'll label the value for "_FREGset_bit", <set_bit>, and the value for "_FREGclr_bit", <clr_bit>.

c. Configure EA 1 for NULL configuration and EA 2 for Binned Burst Catcher

   1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0x012345678 0x055aa6955 0 0 FLAT 0 0 0 0 FLAT 0 0 0x0<catcher> 0 SEND" | easend.

d. Start EAs 1 and 2

   1. ea1on; ea2on

e. Wait for 2 Startup Message Packets to appear in the telemetry monitor window in either order

   1. [69 STARTUP :: PRAM Version = 0x012345678]

   2. [73 STARTUP :: PRAM Version = 0x012345678]

f. If either message does not appear, the test has failed. Record "FAIL" in the "Startup PASS/FAIL" section of Appendix A.1.2. If both messages appear, record "PASS" in the same section.

g. Wait for at least 1/2 a minute after the configuration dump from EA 2. No configuration dump appears from EA 1. The last packet of the dump is indicated as follows:

   1. [73 DUMP :: Burst Catcher 0]

h. If either the dump from EA 2 is incorrect, or a dump was received from EA 1 ([69 DUMP...], the test has failed. Record "FAIL" in the "Dump PASS/FAIL" section of Appendix A.1.2. If both messages appear, record "PASS" in the same section.

i. Pulse the Low-Priority Burst Trigger Signal via two Call-Subroutine invocations

   1. echo "PCA1:Serial:Side-B Super CallSub VAL16 0 FLAT 0x0<set_bit> VAL16 0x0200 SEND PCA1:Serial:Side-B Super CallSub VAL16 0 FLAT 0x0<clr_bit> VAL16 0x0200 SEND" | easend

j. Wait a few seconds for the command acknowledgements

   1. [69 CMDACK :: Supervisor CallSub :: Ok]
      Command (via Ack)::
      Supervisor CallSub
      Address = 0x0<set_bit (segment:offset)>
      Return Type = Integer
      Arg[00] = 0x00200
      Returned Value = <ignore>

   2. [69 CMDACK :: Supervisor CallSub :: Ok]
      Command (via Ack)::
      Supervisor CallSub
      Address = 0x0<clr_bit (segment:offset)>
      Return Type = Integer
      Arg[00] = 0x00200
      Returned Value = <ignore>

k. If either of the acknowledgements from EA 1 are missing or incorrect, or if partition data appeared from EA 2 ([73 PARTHDR...] the test has failed. Record "FAIL" in the "CallSub PASS/FAIL" section of Appendix A.1.2. If both messages appear, record "PASS" in the same section.

l. Terminate the Burst Catcher Observation and verify acknowledgment

   1. ea2term

   2. [73 CMDACK :: Science Terminate :: Ok]

m. If the acknowledgement from EA 2 is missing or incorrect. the test has failed. Record "FAIL" in the "Terminate PASS/FAIL" section of Appendix A.1.2. If both messages appear, record "PASS" in the same section.

n. Wait for 4 Partitions of captured data

    1. [73 PARTHDR :: # = <constant + 0> Len = 0x02000]
    2. [73 PARTDATA :: # = 00]
    3. [73 PARTHDR :: # = <constant + 1> Len = 0x02000]
    4. [73 PARTDATA :: # = 00]
    5. [73 PARTHDR :: # = <constant + 2> Len = 0x02000]
    6. [73 PARTDATA :: # = 00]
    7. [73 PARTHDR :: # = <constant + 3> Len = 0x02000]
    8. [73 PARTDATA :: # = 00]

o. If the Partitions from EA 2 are missing or incorrect. the test has failed. Record "FAIL" in the "Capture PASS/FAIL" section of Appendix A.1.2. If both messages appear, record "PASS" in the same section.

p. Wait for Observation Coda

    1. [73 OBSCODA :: PartCnt = 00000000]

q. If the Coda from EA 2 is missing or incorrect. the test has failed. Record "FAIL" in the "Coda PASS/FAIL" section of Appendix A.1.2. If both messages appear, record "PASS" in the same section.

2.6 Test Case - High Priority Trigger

2.6.1 Objective

The objective of this test is to ensure that the Flight Software Alpha Release can generate a pulse on the High-Priority Inter-EA Burst Trigger Signal and that Burst Catcher Mode properly detects and indicates the presence of the signal.

2.6.2 Estimate

This test takes under 30 minutes to run.

2.6.3 Test Execution

a. Find the flat address of Burst Catcher Configuration in the map file

   1. Search fsrom.map for "_romLibCatcherBinned". The first value after the symbol is the flat address of the configuration (note: it may not be on the same line). For the purposes of this script, we'll label this value <catcher>.

b. Find the flat address for FREGset_bit() and FREGclr_bit in the map file

   1. Search fsrom.map for "_FREGset_bit" and "_FREGclr_bit". The first value after the symbol is the flat address of the function (note: it may not be on the same line). For the purposes of this script, we'll label the value for "_FREGset_bit", <set_bit>, and the value for "_FREGclr_bit", <clr_bit>.

c. Configure EA 1 for NULL configuration and EA 2 for Binned Burst Catcher

   1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0x012345678 0x055aa6955 0 0 FLAT 0 0 0 0 FLAT 0 0 0x0<catcher> 0 SEND" | easend.

d. Start EAs 1 and 2

   1. ea1on; ea2on

e. Wait for 2 Startup Message Packets to appear in the telemetry monitor window in either order

   1. [69 STARTUP :: PRAM Version = 0x012345678]

   2. [73 STARTUP :: PRAM Version = 0x012345678]

f. If either message does not appear, the test has failed. Record "FAIL" in the "Startup PASS/FAIL" section of Appendix A.1.3. If both messages appear, record "PASS" in the same section.

g. Wait for at least 1/2 a minute after the configuration dump from EA 2. No configuration dump appears from EA 1. The last packet of the dump is indicated as follows:

   1. [73 DUMP :: Burst Catcher 0]

h. If either the dump from EA 2 is incorrect, or a dump was received from EA 1 ([69 DUMP...], the test has failed. Record "FAIL" in the "Dump PASS/FAIL" section of Appendix A.1.3. If both messages appear, record "PASS" in the same section.

i. Pulse the High-Priority Burst Trigger Signal via two Call-Subroutine invocations

   1. echo "PCA1:Serial:Side-B Super CallSub VAL16 0 FLAT 0x0<set_bit> VAL16 0x0200 SEND PCA1:Serial:Side-B Super CallSub VAL16 0 FLAT 0x0<clr_bit> VAL16 0x0400 SEND" | easend

j. Wait a few seconds for the command acknowledgements and the Partition Data.

   1. [69 CMDACK :: Supervisor CallSub :: Ok]
      Command (via Ack)::
      Supervisor CallSub
      Address = 0x0<set_bit (segment:offset)>
      Return Type = Integer
      Arg[00] = 0x00400
      Returned Value = <ignore>

   2. [69 CMDACK :: Supervisor CallSub :: Ok]
      Command (via Ack)::
      Supervisor CallSub
      Address = 0x0<clr_bit (segment:offset)>
      Return Type = Integer
      Arg[00] = 0x00400
      Returned Value = <ignore>

   3. [73 PARTHDR :: # = <constant + 0> Len = 0x02000]

   4. [73 PARTDATA :: # = 00]

   5. [73 PARTHDR :: # = <constant + 1> Len = 0x02000]

   6. [73 PARTDATA :: # = 00]

   7. [73 PARTHDR :: # = <constant + 2> Len = 0x02000]

   8. [73 PARTDATA :: # = 00]

9. [73 PARTHDR :: # = <constant + 3> Len = 0x02000]

10. [73 PARTDATA :: # = 00]

k. If either of the acknowledgements from EA 1 are missing or incorrect, or if partition data from EA 2 is missing or incorrect he test has failed. Record "FAIL" in the "Capture PASS/FAIL" section of Appendix A.1.3. If both messages appear, record "PASS" in the same section.

l. Terminate the Burst Catcher Observation and verify acknowledgment

1. ea2term

2. [73 CMDACK :: Science Terminate :: Ok]

m. If the acknowledgement from EA 2 is missing or incorrect. the test has failed. Record "FAIL" in the "Terminate PASS/FAIL" section of Appendix A.1.3. If both messages appear, record "PASS" in the same section.

n. Wait for Observation Coda

1. [73 OBSCODA :: PartCnt = 00000000]

o. If the Coda from EA 2 is missing or incorrect. the test has failed. Record "FAIL" in the "Coda PASS/FAIL" section of Appendix A.1.3. If both messages appear, record "PASS" in the same section.

2.7 Test Case - HEXTE Output Trigger

2.7.1 Objective

The objective of this test is to ensure that the Flight Software Alpha Release can generate a pulse on the HEXTE Output Burst Trigger Signal.

2.7.2 Estimate

This test takes under 30 minutes to run.

2.7.3 Test Execution

a. Reset the HEXTE Output Trigger LED on the MIT Spacecraft Simulator

b. Find the flat address for TIMINGtrigger_hexte() in the map file

    1. Search fsrom.map for "_TIMINGtrigger_hexte". The first value after the symbol is the flat address of the function (note: it may not be on the same line). For the purposes of this script, we'll label the value, <trigger_hexte>.

c. Configure EA 1 for NULL configuration

    1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0x012345678 0x055aa6955 0 0 FLAT 0 0 0 0 FLAT 0 0 0 0 0 SEND" | easend.

d. Start EA 1

    1. ea1on

e. Wait for a Startup Message Packet to appear in the telemetry monitor window

    1. [69 STARTUP :: PRAM Version = 0x012345678]

f. If the message does not appear, the test has failed. Record "FAIL" in the "Startup PASS/FAIL" section of Appendix A.1.4. If both messages appear, record "PASS" in the same section.

g. Pulse the HEXTE Output Burst Trigger Signal via a Call-Subroutine invocations and examine the pulse width measured on the oscilloscope. Also examine the HEXTE Output Trigger LED on the MIT Spacecraft Simulator.

    1. echo "PCA1:Serial:Side-B Super CallSub VAL16 0 FLAT 0x0<trigger_hexte> SEND" | easend

h. If the measured pulse width is less than 31ms or greater than 100ms, or if the trigger LED on the Simulator failed is not On, the test failed. Record "FAIL" in the "HEXTE Pulse PASS/FAIL" section of Appendix A.1.4. If the pulse width is valid, record "PASS" in the same section.

i. Wait a few seconds for the command acknowledgements

    1. [69 CMDACK :: Supervisor CallSub :: Ok]
       Command (via Ack)::
       Supervisor CallSub

Address	= 0x0<trigger_hexte (segment:offset)>
Return Type	= Integer
Returned Value	= <ignore>

j. If the acknowledgement from EA 1 is missing or incorrect, the test has failed. Record "FAIL" in the "CallSub PASS/FAIL" section of Appendix A.1.4. If both messages appear, record "PASS" in the same section.

2.8 Test Case - HEXTE Input Trigger

2.8.1 Objective

The objective of this test is to ensure that the Flight Software Alpha Release can run a Burst Catcher Mode which properly detects and indicates the presence of the signal.

2.8.2 Estimate

This test takes under 30 minutes to run.

2.8.3 Test Execution

a. Find the flat address of Burst Catcher Configuration in the map file

    1. Search fsrom.map for "_romLibCatcherBinned". The first value after the symbol is the flat address of the configuration (note: it may not be on the same line). For the purposes of this script, we'll label this value <catcher>.

b. Configure EA 2 for Binned Burst Catcher

    1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0x012345678 0x055aa6955 0 0 FLAT 0 0 0 0 FLAT 0 0 0x0<catcher> 0 SEND" | easend.

c. Start EA 2

    1. ea2on

d. Wait for a Startup Message Packet to appear in the telemetry monitor window

    1. [73 STARTUP :: PRAM Version = 0x012345678]

e. If the message does not appear, the test has failed. Record "FAIL" in the "Startup PASS/FAIL" section of Appendix A.1.5. If message appears, record "PASS" in the same section.

f. Wait for at least 1/2 a minute after the configuration dump from EA 2. The last packet of the dump is indicated as follows:

    1. [73 DUMP :: Burst Catcher 0]

g. If the dump from EA 2 is incorrect, or if partition data appears, the test has failed. Record "FAIL" in the "Dump PASS/FAIL" section of Appendix A.1.5. If the message appears, record "PASS" in the same section.

h. Push the HEXTE Input Trigger signal on the MIT Spacecraft Simulator

i. Wait a few seconds for the Partition Data.

    1. [73 PARTHDR :: # = <constant + 0> Len = 0x02000]
    2. [73 PARTDATA :: # = 00]
    3. [73 PARTHDR :: # = <constant + 1> Len = 0x02000]
    4. [73 PARTDATA :: # = 00]
    5. [73 PARTHDR :: # = <constant + 2> Len = 0x02000]
    6. [73 PARTDATA :: # = 00]
    7. [73 PARTHDR :: # = <constant + 3> Len = 0x02000]
    8. [73 PARTDATA :: # = 00]

j. If the partition data from EA 2 is missing or incorrect he test has failed. Record "FAIL" in the "Capture PASS/FAIL" section of Appendix A.1.5. If both messages appear, record "PASS" in the same section.

k. Terminate the Burst Catcher Observation and verify acknowledgment

    1. ea2term
    2. [73 CMDACK :: Science Terminate :: Ok]

l. If the acknowledgement from EA 2 is missing or incorrect. the test has failed. Record "FAIL" in the "Terminate PASS/FAIL" section of Appendix A.1.5. If both messages appear, record "PASS" in the same section.

m. Wait for Observation Coda

    1. [73 OBSCODA :: PartCnt = 00000000]

n. If the Coda from EA 2 is missing or incorrect. the test has failed. Record "FAIL" in the "Coda PASS/FAIL" section of Appendix A.1.5. If both messages appear, record "PASS" in the same section.

3 PCA Transparent Configuration Tests

3.1 Scope

The PCA Transparent Configuration Tests verify that Event Encoded Mode, when configured using Transparent Mode Configurations 1, 2 and 3, records all PCA event bits, time-stamped within 1 microsecond, and accurately telemeters the obtained recorded events.

3.2 Identification

The procedure for these tests are completely identified and defined in this section. The results of these tests are reported in Appendix A.2.

3.3 Overview of Test

This section describes one test:

a. Event Capture Test - Verify that all bits are passed through Transparent Mode and that the events are time-stamped to within 1 microsecond.

3.4 Global Setup

a. Setup hardware (seek assistance if needed)

b. Turn on power to Spacecraft Simulator and reset simulator

c. Start command server

   1. stty 9600 -even -odd pass8 raw > /dev/tty01

   2. command_server >& /dev/null&

d. Turn on power to System Manager

e. Zero PRAM Version and flag PRAM as invalid

   1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0 0 SEND" | easend

f. Turn all EAs on and waiting for uplink to boot (i.e. On but Quiet)

   1. ea0up; ea1up; ea2up; ea3up

g. In a window reserved for the telemetry monitor, start telemetry capture and monitor

   1. eaget | tee raw.pkts | watch_data -data

h. Turn On an EA 1

   1. ea1on

i. Wait for Startup Message from the telemetry monitor

   1. [69 STARTUP :: PRAM Version = 0x00]

j. If the message above does not appear within 1/2 a minute, there is something wrong with the setup. Seek assistance.

k. Stop telemetry from EA 1

   1. ea1up

3.5 Test Case - Event Capture

3.5.1 Objective

The objective of this test is to ensure that the Flight Software Alpha Release can reliably acquire, time-stamp, and telemeter all PCA event patterns.

3.5.2 Estimate

This test takes under 30 minutes to run.

3.5.3 Test Execution

a. Find the flat addresses of Transparent Modes 1, 2 and 3 in the map file

   1. Search fsrom.map for "_science_Transparent1", "_science_Transparent2" and "_scienceTransparent3". The first value after the symbol is the flat address of the configuration (note: it may not be on the same line). For the purposes of this script, we'll label these values <trans1>, <trans2> and <trans3>.

b. Configure EA 1 for Transparent Mode 1, EA 2 for Transparent Mode 2, and EA 3 for Transparent Mode 3.

1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0x012345678 0x055aa6955 0 0 FLAT 0 0 0 0 FLAT 0 0x0<trans1> 0x0<trans2> 0x0<trans3> SEND" | easend.

c. Load the PCA Simulator with a known event sequence which asserts and deasserts all PCU bits.

   1. echo "stop" | pca_sim > /dev/pp0

   2. echo "disable 0" | pca_sim > /dev/pp0

   3. For each PCU type: pca_sim > /dev/pp0, followed by

      (a) load <pcu#>

      (b) e:h 0:d 16384

      (c) e:h 255:x 3:v:l1:l2:l3:r1:r2:r3:p:c:d 16384

      (d) Ctrl-D

d. Start EA 1, 2, and 3

   1. ea1on; ea2on; ea3on

e. Wait for 3 Startup Message Packets to appear in the telemetry monitor window in any order

   1. [69 STARTUP :: PRAM Version = 0x012345678]

   2. [73 STARTUP :: PRAM Version = 0x012345678]

   3. [77 STARTUP :: PRAM Version = 0x012345678]

f. If either message does not appear, the test has failed. Record "FAIL" in the "Startup PASS/FAIL" section of Appendix A.2.2. If both messages appear, record "PASS" in the same section.

g. Wait for at least 1/2 a minute after the configuration dump from the three EAs.

h. If either the dumps from the EAs are incorrect, the test has failed. Record "FAIL" in the "Dump PASS/FAIL" section of Appendix A.2.2. If both messages appear, record "PASS" in the same section.

i. Wait for the second partition of data from each of the 3 EAs. Verify that each EA sends 129 words of partition data. If the data does not arrive, or if any of the words are incorrect, the failed. Record "FAIL" in the "Markers PASS/FAIL" section of Appendix A.2.2. If both messages appear, record "PASS" in the same section.

1. The first word should be 0x07ffc (for the 1st partition, this value is 0x07ff8)

2. The next 127 words should be 0x07ff8

3. The last word should be 0x07f7f (end-of-data marker)

j. Enable events from the simulator and verify the events against those loaded into the simulator.

   1. echo "enable 0" | pca_sim > /dev/pp0

   2. echo "run single" | pca_sim > /dev/pp0

k. If any of the events are incorrect, or have an invalid time-stamp, the test has failed. Record "FAIL" in the "Capture PASS/FAIL" section of Appendix A.2.2. If both messages appear, record "PASS" in the same section.

   1. A total of 10 events should be captured from each EA, with 2 events from each PCU (per EA).

   2. The first event from each PCU should have 0 for all bits but the time-stamp and PCU Id.

   3. The second event from each PCU should have all 1's for all bits except the time-stamp and PCU Id.

   4. The time between the two events from a given PCU should be 16384 micro-seconds (or about 17179.8 microticks).

4 PCA Binned Mode Performance Tests

4.1 Scope

The PCA Binned Mode Performance Tests verify the event processing performance of Binned Mode.

4.2 Identification

The procedure for these tests are completely identified and defined in this section. The results of these tests are reported in Appendix A.3.

4.3 Overview of Test

This section describes a single test:

a. Binned Performance - Verify event processing performance of Binned Mode. This test verifies that over 468K events can be processed each second. Since the EAs have 1024 event FIFO, this rate ensures the Binned Mode can process 500K events/second for up to 32ms (i.e. 1024/(500,000 - 468,000) == 32ms).

4.4 Global Setup

a. Setup hardware (seek assistance if needed)

b. Turn on power to Spacecraft Simulator and reset simulator

c. Start command server

    1. stty 9600 -even -odd pass8 raw > /dev/tty01

    2. command_server >& /dev/null&

d. Turn on power to System Manager

e. Zero PRAM Version and flag PRAM as invalid

    1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0 0 SEND" | easend

f. Turn all EAs on and waiting for uplink to boot (i.e. On but Quiet)

    1. ea0up; ea1up; ea2up; ea3up

g. In a window reserved for the telemetry monitor, start telemetry capture and monitor

    1. eaget | tee raw.pkts | watch_data -sum

h. Turn On an EA 1

    1. ea1on

i. Wait for Startup Message from the telemetry monitor

    1. [69 STARTUP :: PRAM Version = 0x00]

j. If the message above does not appear within 1/2 a minute, there is something wrong with the setup. Seek assistance.

k. Stop telemetry from EA 1

    1. ea1up

4.5 Test Case - Binned Performance

4.5.1 Objective

The objective of this test is to ensure that the Flight Software Alpha Release can process over 480K events per second.

4.5.2 Estimate

This test takes under 30 minutes to run.

4.5.3 Test Execution

a. Find the flat addresses of Binned TimeOnly in the map file

    1. Search fsrom.map for "_romLibBinnedTimeOnly". The first value after the symbol is the flat address of the configuration (note: it may not be on the same line). For the purposes of this script, we'll label this value <binned>.

b. Configure EA 1 for Binned TimeOnly Mode.

    1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0x012345678 0x055aa6955 0 0 FLAT 0 0 0 0 FLAT 0 0x0<binned> 0 0 SEND" | easend.

c. Load the PCA Simulator with a known event sequence which asserts and deasserts all PCU bits and supplies over 500K events per second.

   1. echo "stop" | pca_sim > /dev/pp0

   2. echo "disable 0" | pca_sim > /dev/pp0

   3. For each PCU type: pca_sim > /dev/pp0, followed by

      (a) load <pcu#>

      (b) e:h 0:d 10

      (c) e:h 255:x 3:v:l1:l2:l3:r1:r2:r3:p:c:d 10

      (d) Ctrl-D

   4. For PCU 1, run at a slower rate: pca_sim > /dev/pp0, followed by

      (a) load 1

      (b) e:h 0:d 11

      (c) e:h 255:x 3:v:l1:l2:l3:r1:r2:r3:p:c:d 11

      (d) Ctrl-D

d. Start EA 1

   1. ea1on

e. Wait for a Startup Message Packets to appear.

   1. [69 STARTUP :: PRAM Version = 0x012345678]

f. If the message does not appear, the test has failed. Record "FAIL" in the "Startup PASS/FAIL" section of Appendix A.3.2. If both messages appear, record "PASS" in the same section.

g. Wait for at least 1/2 a minute after the configuration dump from the EAs.

h. If the dump is incorrect, the test has failed. Record "FAIL" in the "Dump PASS/FAIL" section of Appendix A.3.2. If both messages appear, record "PASS" in the same section.

i. Enable events from the simulator and verify the event processed event rate.

   1. echo "enable 0" | pca_sim > /dev/pp0

2. echo "run repeat" | pca_sim > /dev/pp0

3. The sum of the bins in each partition should be about 490,909 counts/partition.

j. If the telemetry indicates spillage or have an invalid event count, the test has failed. Record "FAIL" in the "Count PASS/FAIL" section of Appendix A.3.2. If both messages appear, record "PASS" in the same section.

5 ASM Data Tests

5.1 Scope

The ASM Data Tests verify that ASM Event by Event Mod records all ASM event bits and accurately telemeters the obtained recorded events.

5.2 Identification

The procedure for these tests are completely identified and defined in this section. The results of these tests are reported in Appendix A.4.

5.3 Overview of Test

This section describes one test:

a. ASM Event Capture Test - Verify that all bits from each of the SSCs are passed through. This test relies on a couple of "back-door" tricks to reconfigure ASM Event Mode's DSP parameters for each SSC, and restart the Flight Software without reloading ROM and, therefore, overwrite the reconfigured parameters.

5.4 Global Setup

a. Setup hardware (seek assistance if needed)

b. Turn on power to Spacecraft Simulator and reset simulator

c. Turn on power to ASM Simulator and reset simulator

d. Start command server

   1. stty 9600 -even -odd pass8 raw > /dev/tty01

   2. command_server >& /dev/null&

e. Turn on power to System Manager

f. Zero PRAM Version and flag PRAM as invalid

1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0 0 SEND" | easend

g. Turn all EAs on and waiting for uplink to boot (i.e. On but Quiet)

   1. ea0up; ea1up; ea2up; ea3up

h. In a window reserved for the telemetry monitor, start telemetry capture and monitor

   1. eaget | tee raw.pkts | watch_data -data -modedata

i. Turn On an EA 1

   1. ea1on

j. Wait for Startup Message from the telemetry monitor

   1. [69 STARTUP :: PRAM Version = 0x00]

k. If the message above does not appear within 1/2 a minute, there is something wrong with the setup. Seek assistance.

l. Stop telemetry from EA 1

   1. ea1up

5.5 Test Case - ASM Event Capture

5.5.1 Objective

The objective of this test is to ensure that the Flight Software Alpha Release can capture all ASM event information.

5.5.2 Estimate

This test takes under 30 minutes to run.

5.5.3 Test Execution

a. Find the flat address of ASM Event by Event in the map file

   1. Search fsrom.map for "_romLibAsmeventMode". The first value after the symbol is the flat address of the configuration (note: it may not be on the same line). For the purposes of this script, we'll label this value <asmevent>.

b. Find the flat address of the SSC selection parameter in DSP Parameters for ASM Event by Event mode.

1. Search fsrom.map for "_romLibAsmeventModeDSP". Add 8 to the obtained value. The result points to the SSC Selection parameter within the DSP's parameter block. For the purposes of this script, we'll label this value <ssc-select>.

c. Find the flat address of the initialization code for the Flight Software

   1. Search fsrom.map for "_STARTUPinit". For the purposes of this script, we'll label this value <restart>.

d. Configure EA 0 for ASM Event by Event Mode.

   1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0x012345678 0x055aa6955 0 0 FLAT 0 0 0 0 FLAT 0x0<asmevent> 0 0 0 SEND" | easend.

e. Load and start the ASM Simulator with a known event sequence which asserts and deasserts all ASM bits.

   1. asm_sim, followed by

      (a) s 1:w 0:a 0:b 0:d 32767

      (b) s 1:w 15:a 4095:b 4095:d 32767

      (c) s 2:w 0:a 0:b 0:d 32767

      (d) s 2:w 15:a 4095:b 4095:d 32767

      (e) s 3:w 0:a 0:b 0:d 32767

      (f) s 3:w 15:a 4095:b 4095:d 32767

      (g) g

      (h) Ctrl-D

f. Start EA 0

   1. ea0on

g. Wait for a Startup Message Packets to appear.

   1. [69 STARTUP :: PRAM Version = 0x012345678]

h. If the message does not appear, the test has failed. Record "FAIL" in the "Startup PASS/FAIL" section of Appendix A.4.2. If both messages appear, record "PASS" in the same section.

i. Wait for at least 1/2 a minute after the configuration dump from the EAs.

j. If the dump is incorrect, the test has failed. Record "FAIL" in the "Dump PASS/FAIL" section of Appendix A.4.2. If both messages appear, record "PASS" in the same section.

k. Verify pairs of events from the default SSC 1

   1. First event should have all 0's except for the time-stamp field

   2. The second event should have all 1's except for the time-stamp field

l. If the telemetry indicates spillage or have invalid events, the test has failed. Record "FAIL" in the "SSC 1 Capture PASS/FAIL" section of Appendix A.4.2. If both messages appear, record "PASS" in the same section.

m. Set the SSC Selection Parameter in the DSP Load Block to acquire events from SSC 2 and restart the EA without reloading ROM

   1. echo "ASM:Serial:Side-B Super WriteMem FLAT 0x0<sscselect> VAL16 2 VAL16 1 SEND" | easend

   2. echo "ASM:Serial:Side-B Super CallSub VAL16 0 FLAT 0x0<restart> SEND" | easend

n. The EA should then restart, issuing a new startup message and parameter dump. If not, the test has failed. Record "FAIL" in the "Restart 1 PASS/FAIL" section of Appendix A.4.2. If both messages appear, record "PASS" in the same section.

o. Verify pairs of events from the SSC 2

   1. First event should have all 0's except for the time-stamp field

   2. The second event should have all 1's except for the time-stamp field

p. If the telemetry indicates spillage or have invalid events, the test has failed. Record "FAIL" in the "SSC 2 Capture PASS/FAIL" section of Appendix A.4.2. If both messages appear, record "PASS" in the same section.

q. Set the SSC Selection Parameter in the DSP Load Block to acquire events from SSC 3 and restart the EA without reloading ROM

   1. echo "ASM:Serial:Side-B Super WriteMem FLAT 0x0<sscselect> VAL16 2 VAL16 2 SEND" | easend

   2. echo "ASM:Serial:Side-B Super CallSub VAL16 0 FLAT 0x0<restart> SEND" | easend

r. The EA should then restart, issuing a new startup message and parameter dump. If not, the test has failed. Record "FAIL" in the "Restart 2 PASS/FAIL" section of Appendix A.4.2. If both messages appear, record "PASS" in the same section.

s. Verify pairs of events from the SSC 3

   1. First event should have all 0’s except for the time-stamp field

   2. The second event should have all 1’s except for the time-stamp field

t. If the telemetry indicates spillage or have invalid events, the test has failed. Record "FAIL" in the "SSC 3 Capture PASS/FAIL" section of Appendix A.4.2. If both messages appear, record "PASS" in the same section.

6 ASM Performance Tests

6.1 Scope

The ASM Position Histogram Mode Performance Tests verify the event processing performance of Position Histogram Mode.

6.2 Identification

The procedure for these tests are completely identified and defined in this section. The results of these tests are reported in Appendix A.5.

6.3 Overview of Test

This section describes a single test:

a. Position Histogram Performance - Verify event processing performance of Position Histogram Mode. This test verifies that over 30K events can be processed each second. This test relies on a couple of "back-door" tricks to reconfigure ASM Position Histogram Mode's parameters to shorten the dwell duration from 64 seconds to 5 seconds and restart the Flight Software without reloading ROM and, therefore, overwrite the reconfigured parameters.

6.4 Global Setup

a. Setup hardware (seek assistance if needed)

b. Turn on power to Spacecraft Simulator and reset simulator

c. Turn on power to ASM Simulator and reset simulator

d. Start command server

    1. stty 9600 -even -odd pass8 raw > /dev/tty01

    2. command_server >& /dev/null&

e. Turn on power to System Manager

f. Zero PRAM Version and flag PRAM as invalid

    1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0 0 SEND" | easend

g. Turn all EAs on and waiting for uplink to boot (i.e. On but Quiet)

    1. ea0up; ea1up; ea2up; ea3up

h. In a window reserved for the telemetry monitor, start telemetry capture and monitor

    1. eaget | tee raw.pkts | watch_data -sum

i. Turn On an EA 1

    1. ea1on

j. Wait for Startup Message from the telemetry monitor

    1. [69 STARTUP :: PRAM Version = 0x00]

k. If the message above does not appear within 1/2 a minute, there is something wrong with the setup. Seek assistance.

l. Stop telemetry from EA 1

    1. ea1up

6.5 Test Case - ASM Performance

6.5.1 Objective

The objective of this test is to ensure that the Flight Software Alpha Release of Position Histogram Mode can process over 30K events per second.

6.5.2 Estimate

This test takes under 30 minutes to run.

6.5.3 Test Execution

a. Find the flat address of ASM Position Histogram Mode in the map file

    1. Search fsrom.map for "_romLibPositionMode". The first value after the symbol is the flat address of the configuration (note: it may not be on the same line). For the purposes of this script, we'll label this value <position>.

b. Compute the flat address of the dwell duration parameter in parameter block by adding 26 to <position>. For the purposes of this script, we'll label this value <dwell>.

c. Find the flat address of the initialization code for the Flight Software

    1. Search fsrom.map for "_STARTUPinit". For the purposes of this script, we'll label this value <restart>.

d. Configure EA 0 for ASM Position Histogram Mode

    1. echo "PRAM:Serial:Side-B VAL16 0 VAL32 0x012345678 0x055aa6955 0 0 FLAT 0 0 0 0 FLAT 0x0<asmevent> 0 0 0 SEND" | easend.

e. Load and start the ASM Simulator with a known event sequence which generates over 30K non-background events/second.

    1. asm_sim, followed by

        (a) s 1:w 1:a 10:b 10:d 0

        (b) s 2:w 1:a 10:b 10d 0

        (c) s 3:w 1:a 10:b 10:d 3

        (d) g

        (e) Ctrl-D

f. Start EA 0 to load PRAM and the configuration pointer

    1. ea0on

g. Wait for a Startup Message Packets to appear.

    1. [69 STARTUP :: PRAM Version = 0x012345678]

h. If the message does not appear, the test has failed. Record "FAIL" in the "Startup PASS/FAIL" section of Appendix A.5.2. If both messages appear, record "PASS" in the same section.

i. Wait for at least 1/2 a minute after the configuration dump from the EAs.

j. If the dump is incorrect, the test has failed. Record "FAIL" in the "Dump PASS/FAIL" section of Appendix A.5.2. If both messages appear, record "PASS" in the same section.

k. Patch the dwell duration to use 5 second dwells instead of 64 second dwells and restart the EA

    1. echo "ASM:Serial:Side-B Super WriteMem FLAT 0x0<dwell> VAL16 2 VAL16 5 SEND" | easend

    2. echo "ASM:Serial:Side-B Super CallSub VAL16 0 FLAT 0x0<restart> SEND" | easend

l. The EA should then restart, issuing a new startup message and parameter dump. If not, the test has failed. Record "FAIL" in the "Restart PASS/FAIL" section of Appendix A.5.2. If both messages appear, record "PASS" in the same section.

m. Verify the event processed event rate.

    1. The sum of the "good-event" bins in each partition should be about 31,250 counts/partition.

n. If the telemetry indicates spillage or have an invalid event count, the test has failed. Record "FAIL" in the "Count PASS/FAIL" section of Appendix A.5.2. If both messages appear, record "PASS" in the same section.

o. Examine the ASM Housekeeping Telemetry packets with each dwell.

p. If an ASM Housekeeping packet is missing, or the information within the packet is incorrect, the test has failed. Record "FAIL" in the "House PASS/FAIL" section of Appendix A.5.2. If the Housekeeping is valid, record "PASS" in the same section.

Appendix A - Test Reports

A.1 Burst Trigger Signal Test Report

A.1.1 Overview

A.1.1.1 Test Run

Tester	Date	Time

A.1.1.2 Hardware Versions

SubSystem	Version
EA Subgroup	
MIT Spacecraft Simulator	
ASM Instrument Simulator	
PCA Event Simulator	

A.1.1.3 Software Versions

SubSystem	Program	Version
Command Uplink	command_server	
	emit_command	
Telemetry Acquisition	mit_telemetry	
Command Packet Builder	fscmd	
	tpkt	
Telemetry Parsers	watch_data	
	unpackScienceData	
	eventSum	
Command Aliases	ea[0-3]off	
	ea[0-3]up	
	ea[0-3]on	
	ea[0-3]term	

A.1.2 Test Case - Low-Priority Trigger

Stage	Pass/Fail
Startup PASS/FAIL	
Dump PASS/FAIL	
CallSub PASS/FAIL	
Terminate PASS/FAIL	
Capture PASS/FAIL	
Coda PASS/FAIL	
Low-Priority Trigger PASS/FAIL	
NOTES:	

A.1.3 Test Case - High-Priority Trigger

Stage	Pass/Fail
Startup PASS/FAIL	
Dump PASS/FAIL	
Capture PASS/FAIL	
Terminate PASS/FAIL	
Coda PASS/FAIL	
High-Priority Trigger PASS/FAIL	
NOTES:	

A.1.4 Test Case - HEXTE Output Trigger

Stage	Pass/Fail
Startup PASS/FAIL	
CallSub PASS/FAIL	
Pulse PASS/FAIL	
HEXTE Output Trigger PASS/FAIL	
NOTES:	

A.1.5 Test Case - HEXTE Input Trigger

Stage	Pass/Fail
Startup PASS/FAIL	
Dump PASS/FAIL	
Capture PASS/FAIL	
Terminate PASS/FAIL	
Coda PASS/FAIL	
HEXTE Input Trigger PASS/FAIL	
NOTES:	

A.2 PCA Transparent Configuration Test Report

A.2.1 Overview

A.2.1.1 Test Run

Tester	Date	Time

A.2.1.2 Hardware Versions

SubSystem	Version
EA Subgroup	
MIT Spacecraft Simulator	
ASM Instrument Simulator	
PCA Event Simulator	

A.2.1.3 Software Versions

SubSystem	Program	Version
Command Uplink	command_server	
	emit_command	
Telemetry Acquisition	mit_telemetry	
Command Packet Builder	fscmd	
	tpkt	
Telemetry Parsers	watch_data	
	unpackScienceData	
	eventSum	
Command Aliases	ea[0-3]off	
	ea[0-3]up	
	ea[0-3]on	
	ea[0-3]term	

A.2.2 Test Case - Event Capture

Stage	Pass/Fail
Startup PASS/FAIL	
Dump PASS/FAIL	
Markers PASS/FAIL	
Capture PASS/FAIL	
Event Capture PASS/FAIL	
NOTES:	

A.3 PCA Binned Performance Test Report

A.3.1 Overview

A.3.1.1 Test Run

Tester	Date	Time

A.3.1.2 Hardware Versions

SubSystem	Version
EA Subgroup	
MIT Spacecraft Simulator	
ASM Instrument Simulator	
PCA Event Simulator	

A.3.1.3 Software Versions

SubSystem	Program	Version
Command Uplink	command_server	
	emit_command	
Telemetry Acquisition	mit_telemetry	
Command Packet Builder	fscmd	
	tpkt	
Telemetry Parsers	watch_data	
	unpackScienceData	
	eventSum	
Command Aliases	ea[0-3]off	
	ea[0-3]up	
	ea[0-3]on	
	ea[0-3]term	

A.3.2 Test Case - Binned Performance

Stage	Pass/Fail
Startup PASS/FAIL	
Dump PASS/FAIL	
Count PASS/FAIL	
Binned Performance PASS/FAIL	
NOTES:	

A.4 ASM Data Test Report

A.4.1 Overview

A.4.1.1 Test Run

Tester	Date	Time

A.4.1.2 Hardware Versions

SubSystem	Version
EA Subgroup	
MIT Spacecraft Simulator	
ASM Instrument Simulator	
PCA Event Simulator	

A.4.1.3 Software Versions

SubSystem	Program	Version
Command Uplink	command_server	
	emit_command	
Telemetry Acquisition	mit_telemetry	
Command Packet Builder	fscmd	
	tpkt	
Telemetry Parsers	watch_data	
	unpackScienceData	
	eventSum	
Command Aliases	ea[0-3]off	
	ea[0-3]up	
	ea[0-3]on	
	ea[0-3]term	

A.4.2 Test Case - ASM Event Capture

Stage	Pass/Fail
Startup PASS/FAIL	
Dump PASS/FAIL	
SSC 1 Capture PASS/FAIL	
Restart 1 PASS/FAIL	
SSC 2 Capture PASS/FAIL	
Restart 2 PASS/FAIL	
SSC 3 Capture PASS/FAIL	
ASM Event Capture PASS/FAIL	
NOTES:	

A.5 ASM Performance Test Report

A.5.1 Overview

A.5.1.1 Test Run

Tester	Date	Time

A.5.1.2 Hardware Versions

SubSystem	Version
EA Subgroup	
MIT Spacecraft Simulator	
ASM Instrument Simulator	
PCA Event Simulator	

A.5.1.3 Software Versions

SubSystem	Program	Version
Command Uplink	command_server	
	emit_command	
Telemetry Acquisition	mit_telemetry	
Command Packet Builder	fscmd	
	tpkt	
Telemetry Parsers	watch_data	
	unpackScienceData	
	eventSum	
Command Aliases	ea[0-3]off	
	ea[0-3]up	
	ea[0-3]on	
	ea[0-3]term	

A.5.2 Test Case - ASM Performance

Stage	Pass/Fail
Startup PASS/FAIL	
Dump PASS/FAIL	
Restart PASS/FAIL	
Count PASS/FAIL	
House PASS/FAIL	
ASM Performance PASS/FAIL	
NOTES:	

Appendix B - Command Aliases

B.1 I/O

The following is a list of the "csh" aliases used to perform I/O:

a. easend - alias easend 'fscmd | tpkt | emit_command'

b. eaget - alias eaget 'mit_telemetry'

B.2 Discrete Commands

The following is a list of the "csh" aliases used to issue discrete commands to the EDS:

a. ea0off - alias ea0off 'echo "ASM:Discrete:Side-B VAL16 0 SEND" | easend'

b. ea1off - alias ea1off 'echo "PCA1:Discrete:Side-B VAL16 0 SEND" | easend'

c. ea2off - alias ea1off 'echo "PCA2:Discrete:Side-B VAL16 0 SEND" | easend'

d. ea3off - alias ea1off 'echo "PCA3:Discrete:Side-B VAL16 0 SEND" | easend'

e. ea0up - alias ea0up 'echo "ASM:Discrete:Side-B VAL16 7 SEND" | easend'

f. ea1up - alias ea1up 'echo "PCA1:Discrete:Side-B VAL16 7 SEND" | easend'

g. ea2up - alias ea2up 'echo "PCA2:Discrete:Side-B VAL16 7 SEND" | easend'

h. ea3up - alias ea3up 'echo "PCA3:Discrete:Side-B VAL16 7 SEND" | easend'

i. ea0on - alias ea0on 'echo "ASM:Discrete:Side-B VAL16 3 SEND" | easend'

j. ea1on - alias ea1on 'echo "PCA1:Discrete:Side-B VAL16 3 SEND" | easend'

k. ea2on - alias ea2on 'echo "PCA2:Discrete:Side-B VAL16 3 SEND" | easend'

l. ea3on - alias ea3on 'echo "PCA3:Discrete:Side-B VAL16 3 SEND" | easend'

B.3 Serial Commands

The following is a list of the "csh" aliases used to issue serial commands to the EDS:

a. ea0term - alias ea0term ’echo "ASM:Serial:Side-B Science Terminate SEND" | easend’

b. ea1term - alias ea1term ’echo "PCA1:Serial:Side-B Science Terminate SEND" | easend’

c. ea2term - alias ea1term ’echo "PCA2:Serial:Side-B Science Terminate SEND" | easend’

d. ea3term - alias ea1term ’echo "PCA3:Serial:Side-B Science Terminate SEND" | easend’