GLAST LAT Flight Software Demonstration

SLAC Campus
Building 84, Central Lab Annex

April 2, 2004
Overview Presentation 10:00 AM (Room B-259)
Demonstrations 10:15 AM

1 Demonstration Overview

1.1 Agenda for the Demonstrations

The demonstrations will take place in the Central Laboratory Annex (Building 84).

Demo Agenda Item	Presenter(s)
1. Introductions (in Group C Conference Room, Room B-259)	NA
2. Overview of the Demonstrations (in Group C Conference Room, Room B-259)	J.J. Russell
3. CCSDS Packet Demo	Tony Waite
4. GASU Configuration Demo	James Swain
5. Single Event Display Demo	James Swain
6. Watchdog Function Demo	Steve Mazzoni
7. Questions from Attendees	NA

Feel free to jot questions and comments down in the margins of this document or in the space provided on page 18.

1.2 Goals of the Demonstrations

The demonstrations will achieve the following goals:

- The CCSDS Packet Demo shows that the CCSDS and CTDB software packages – required for use of the 1553 data bus and for transmission of CCSDS packets back and forth on that bus – are complete and operational. In the demo, this command and telemetry infrastructure is used to support transmission of CCSDS packets between a LAT and a Spacecraft simulated in software.
- The GASU Configuration Demo shows that the DAB and DEM software packages – required for configuring the Global trigger Electronics Module (GEM), Event Builder Module (EBM), ACD Electronics Module (AEM), and Command Response Unit (CRU) subsystems on the GASU by writing and reading their registers – are complete and operational. These two packages also depend on a functioning LAT Communications Board (LCB) driver package, which is also used in the demo. During the demo, these packages enable a CPU crate to configure the CRU, GEM, EBM, and AEM in preparation for event taking during the Single Event Display Demo.
- The Single Event Display Demo shows that a functioning software and hardware infrastructure for retrieving event data using the GEM, EBM, and AEM subsystems of the GASU exists. During the demonstration, a RAD750 crate is used to trigger the GEM manually; in response to the trigger command, the GEM and AEM send event data to the EBM, where it is

assembled and returned to the crate, where it is parsed and the event data and outputs the results to the demonstration terminal.

- The Watchdog Function Demonstration shows that a proof-of-concept design for the software Watchdog is complete, and a test implementation of this design runs on a RAD750. In the demo, the Watchdog function will detect the failure of a test task to make progress, then reboot the RAD750.

1.3 Expected Outputs of the Demonstrations; Verification of Requirements

This section cites the expected outputs of each of the systems tested today, and provides a space for the demonstration monitors to verify by signature that the demonstration ran successfully.

Demo Agenda Item	Relevant FSW Requirements (SRS Version 2)	Expected Output of the Demonstration
CCSDS Packet Demo	5.3.1.1.1, 5.3.1.1.3, 5.3.4.2.3	For a full description of the outputs of the CCSDS Packet Demo, see Section 1.3.1 on page 4. For the demonstration procedure and the demo's software and hardware context, see Section 2.1 on page 13.
		Monitor Signature: _____
GASU Configuration Demo	5.3.3.1, 5.3.4.6.1, 5.3.4.6.2	For a full description of the outputs of the GASU Configuration Demo, see 1.3.2 on page 5. For the demo's software and hardware context and the demonstration procedure, see Section 2.2 on page 14. For a diagram of the GASU and its relationship to LAT CPUs, see Attachment 1 on page 21.
		Monitor Signature: _____
Single Event Display Demo	5.2.1.3	For a full description of the outputs of the Single Event Display Demo, see Section 1.3.3 beginning on page 5. For the demo's software and hardware context and the demonstration procedure, see Section 2.3 on page 15. For a diagram of the GASU and its relationship to LAT CPUs, see Attachment 1 on page 21.
		Monitor Signature: _____

Demo Agenda Item	Relevant FSW Requirements (SRS Version 2)	Expected Output of the Demonstration
Watchdog Function Demo	5.2.1.2, 5.3.2.1	For a full description of the outputs of the Watchdog Function Demo, see Section 1.3.4 on page 9. For the demo's software and hardware context and the demonstration procedure, see Section 2.4 on page 16.
		Monitor Signature: _____

1.3.1 Output of the CCSDS Packet Demo

The CCSDS Packet Demo produces terminal window output displaying the contents of a boot command packet sent from the simulated spacecraft to the RAD750 crate after execution of the **SC_send_boot** command. The command packet instructs the RAD750 to continue from the primary boot process and move on to secondary boot code (SBC) execution without waiting for the normal 10 minute timeout period to elapse.

The demo concludes successfully when the RAD750 crate enters its secondary boot process on command. Other expected output is illustrated in the following figure.

```
->SC_send_boot 0,1,1,1

********************************************************************************
CCSDS header
--------------------------------------------------------------------------------
Offset  Literal    Bits  Value      Description      Interpretation
------- ---------- ----- ---------- ---------------- -------------------------
  0-  1 0x1e40      0- 2 0x00       CCSDS version    CCSDS version 1 packet
                    3- 3 0x01       Type             Telecommand
                    4- 4 0x01       Sec. header      Yes
                   5-15 0x0640      APID             1600 (decimal)
  2-  3 0xc000      0- 1 0x03       Seq. flags       Standalone packet
                    2-15 0x0000     Seq. count       0 (decimal)
  4-  5 0x0009                      CCSDS length     -> 6 byte payload
  6-  7 0x0003      0- 0 0x00       MBZ              Must be zero
                    1-15 0x0003     Function code    3 (decimal)

SC_send_boot payload
--------------------------------------------------------------------------------
Offset  Literal    Bits  Value      Description      Interpretation
------- ---------- ----- ---------- ---------------- -------------------------
  8-  9 0x0000      0- 3 0x00       LAT unit         SIU
                    4-15 0x0000     Padding          Unused
 10- 11 0x5400      0- 1 0x01       RTOS source      EEPROM-0
                    2- 3 0x01       SSB0 source      EEPROM-0
                    4- 5 0x01       SSB1 source      EEPROM-0
 12- 13 0x0000           0x0000     Reserved         EEPROM-0

CCSDS telecommand checksum
--------------------------------------------------------------------------------
```

```
Offset  Literal    Bits  Value      Description      Interpretation
------- ---------- ----- ---------- ---------------- -------------------------
 14- 15 0x017e                      Checksum         Should be 0x017e
*******************************************************************************


->
```

1.3.2 Output of the GASU Configuration Demo

Results of the GASU Configuration Demo are confirmed by the outputs of the Single Event Display Demo (see below). If event data is collected, assembled, and communicated as expected, and as recorded in the output of Single Event Display, then the GASU configuration is assumed to have been successful.

1.3.3 Output of the Single Event Display Demo

The Single Event Display Demo produces terminal window output. Working on a RAD750 crate, the tester sends a trigger command to the GEM, then the GEM issues a Trigger Accept Message. The AEM and the GEM then read back their event data, and forward it to the EBM, which assembles the data. The EBM returns the assembled data to the original crate, where it is parsed and displayed at the demonstration terminal. The report received at the terminal contains event data contributions from the 2 GASU subsystems used in the demonstration:

- **The AEM.** The AEM packetizes data received from ACD front end electronics boards and forwards it to the EBM. For this demo, since no ACD detector hardware is present, the packets are empty (though occasionally, in the current laboratory environment, noise or spikes can show up as data). However, the demo shows that AEM packet header fields (showing empty hit map and suppression map entries) and Pulse Height Analysis fields are present in the event data stream.

Expected output (for the AEM contribution) is shown below.

```
.
.
.
Decoding AEM Event Contribution

   Event Contribution Size (bytes): 96
   Event Contribution Source Addr : 0x13
   Event Contribution Summary Word: 0x8200000c
   ====================================
   Event Number-Tag    : 00006-00
   ====================================
   Error               : 0
   TRG Parity Error    : 0
   ====================================
   Diagnostic          : 0
   Cal Strobe          : 0
   Readout 4           : 0
   Zero supress        : 1
   TACK                : 0
   Marker              : 0
```

```
AEM Veto Hit Vectors
================================================================
-----   ----------------------------------------------------
Cable   Channel 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0   Raw
                7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
-----   ----------------------------------------------------
 00             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 01             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 02             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 03             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 04             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 05             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 06             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 07             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 08             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 09             1 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1   0x2a00f
-----   ----------------------------------------------------
 10             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 11             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------

AEM Accept Hit Vectors
================================================================
-----   ----------------------------------------------------
Cable   Channel 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0   Raw
                7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
-----   ----------------------------------------------------
 00             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 01             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 02             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 03             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 04             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 05             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 06             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 07             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
 08             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ----------------------------------------------------
```

```
 09             1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1   0x3ffff
-----   ---------------------------------------------------------
 10             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ---------------------------------------------------------
 11             0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0   0x00000
-----   ---------------------------------------------------------

AEM PHA Values
================================================================

----+-----+-------+------------+--------------------
Cbl   Rng    ADC    Parity Err    Raw
----+-----+-------+------------+--------------------
 09    1    0xfff      0         0x7ffe
 09    1    0xfff      0         0x7ffe
 09    1    0xe00      0         0x3c00
-------------------------------------------------
```

- **The GEM.** The GEM acts not only as the global trigger – it contributes event data as well. The GEM event data contribution contains:
  - Summaries of data from the ACD, TKR, and CAL front-end boards and their corresponding electronics modules. For the demo, since neither ACD nor TKR/CAL detector hardware are attached, the vector summaries contain no physics data.
  - A Trigger Summary listing all reasons for which the trigger was just fired.
  - A Veto List recording veto signals from the ACD. Again, in the demo, since no ACD detector hardware is attached, the Veto List has 0 values.
  - A sample of recent LAT Livetime (instrument non-busy time) counter data.
  - A sample of recent Busy counter data (showing the number of times that various LAT components were busy during recent trigger attempts).
  - A sample of recent Prescale counter data (showing the number of times that the instrument could have been triggered, but was not triggered because prescalers were used to reduce the importance of certain classes of event data).
  - A sample of recent Sent counter data (showing the number of times a Trigger Accept Message was sent).
  - The Trigger Time, a sampling of the stored GEM internal clock value.
  - 1-PPS Time, a sampling of the stored Spacecraft 1-PPS time signal.

Expected output (for the GEM contribution) is shown below.

```
GLAST/LAT FSW Event Contribution Decoder

Decoding GEM Event Contribution

Event Contribution Size (bytes): 64
Event Contribution Source Addr : 0x12
Event Contribution Summary Word: 0x8200000c
====================================
Event Number-Tag    : 00006-00
====================================
Error               : 0
```

```
TRG Parity Error    : 0
====================================
Diagnostic          : 0
Cal Strobe          : 0
Readout 4           : 0
Zero supress        : 1
TACK                : 0
Marker              : 0

GEM ROI Vector
=====================================================================
0x0000

GEM TKR Vector
=====================================================================
0x0000

GEM CAL HE Vector
=====================================================================
0x0000

GEM CAL LE Vector
=====================================================================
0x0000

GEM CNO Vector
=====================================================================
0x0000

GEM Trigger Summary
=====================================================================
0x0020

GEM ACD Veto (Raw)
=====================================================================
word 0:  0x00000000
word 1:  0x00000000
word 2:  0x00000000
word 3:  0x00000000

GEM Livetime
=====================================================================
0x00052201

GEM Busy
=====================================================================
0x0000007d

GEM Prescale
=====================================================================
0x00000000

GEM Sent
=====================================================================
0x00000018
```

```
   GEM Trigger Time
   ================================================================
   0x010553b2

   GEM 1-PPS Time
   ================================================================
   0x00000001
.
.
.
```

1.3.4 Output of the Watchdog Demo

Two constituents of the LSW package, **lsw** and **lswTest**, are run on a RAD750 CPU to monitor the functioning of 4 VxWorks test tasks.

- While these test tasks are making progress, as measured by their single cycle processing times, the Watchdog software function resets a register on the RAD750's hardware watchdog to prevent the CPU from rebooting.
- When one of the test tasks is deliberately halted and then fails to make progress (becomes "hung"), the Watchdog software function reboots the flight software system.

Expected output is illustrated in the following figures.

1.3.4.1 Expected Watchdog Demo Output During Initialization

```
LSW Test: Start!PBS initialization
------------------
Number of wut tmrs: 256
Keepalive   period: 20000000
Update      period: 1000000000


LSW Test: MSG_initialize OK

LSW Test: MSG_attachOutputRtn OK

LSW Test: MSG_startTask OK

*****LSW WOULD RESET WDT WITH 10000000*****

LSW Test: LSW_init done!
LSW Test: starting task 0 0

LSW Test: task 0 started
LSW Test: starting task 1 0

<0S>
LSW Test: task 1 started
<0: registered! 0>
LSW Test: starting task 2 0

<0R>w0>
```

```
LSW Test: task 2 started
LSW Test: starting task 3 0
.000000000 LSW_Register LSW-I-TASKREG TASK Task0 REGISTERED
.000000000 LSW_Register LSW-E-MLTRGERR TASK Task0 REGISTERED MORE THAN ONCE

<1S>
LSW Test: task 3 started
LSW Test: tasks start complete!
<1: registered! 1>
<1R>w1>.000000000 LSW_Register LSW-I-TASKREG TASK Task1 REGISTERED

<2S>
<3S>
<3: registered! 3>
<3R>
<2: registered! 2>
<2R>w3>w2>.000000000 LSW_Register LSW-I-TASKREG TASK Task3 REGISTERED
.000000000 LSW_Register LSW-I-TASKREG TASK Task2 REGISTERED

<3p..
LSW Registered Task List:
INDEX    NAME    CALLBACK       DATA
0        Task0   12134          243f8
1        Task1   12704          243fc
<1p
2        Task2   12c94          24400.
3        Task3   13220          24404
```

1.3.4.2 Expected Watchdog Demo Output When Tasks Are Making Normal Progress

```
.
.
.
TestMessageInterface: q=ff3a3280....w3>
<2p
<0p...w1>.
<3p....**********.w2>
<1p.....w3>.w0>.
[0-0:tc=101774448871615000 ts=0 d=101774448871615000]

[1-1:tc=101774448871691000 ts=101774447971681000 d=900010000]

[2-2:tc=101774448871701000 ts=0 d=101774448871701000]

[3-3:tc=101774448871709000 ts=0 d=101774448871709000]

*****LSW WOULD RESET WDT WITH 10000000*****

<3p..w1>..
<2p.
<1p...w3>.
<0p..******
<3p.**.*.*.w2>..w1>...w3>
```

```
<1p..w0>.
<3p.
<2p...
[0-0:tc=101774452881635000 ts=0 d=101774452881635000]

[1-1:tc=101774452881662000 ts=101774452181657000 d=700005000]

[2-2:tc=101774452881672000 ts=101774452841634000 d=40038000]

[3-3:tc=101774452881682000 ts=101774452531669000 d=350013000]

*****LSW WOULD RESET WDT WITH 10000000*****
..w1>..**********.w2>.w3>
<1p.
<0p.
<3p.......w1>..w3>
<2p..w0>.
<3p.
<1p....**********.w2>
[0-0:tc=101774456891654000 ts=0 d=101774456891654000]

[1-1:tc=101774456891681000 ts=101774456391664000 d=500017000]

[2-2:tc=101774456891691000 ts=0 d=101774456891691000]

[3-3:tc=101774456891700000 ts=101774456131640000 d=760060000]
.
.
.
```

1.3.4.3 Expected Watchdog Demo Output When Test Task 2 is Deliberately Hung

```
.
.
.
*****LSW WOULD RESET WDT WITH 10000000*****
.w0>..w3>.w1>
<3p..
<2p..
<1p...
<0p..w3>..*****.*****.w2>
<3p..w1>.

The test script sends a signal to hang Task 2
HangTask: tn=2  ln=3
...w0>
Task 2 becomes stuck in a loop; it outputs asterisks
"*****************"to terminal
<1p..w3>
[0-0:tc=101774496981876000 ts=0 d=101774496981876000]
[1-1:tc=101774496981894000 ts=101774496851878000 d=130016000]
[2-2:tc=101774496981904000 ts=0 d=101774496981904000]
[3-3:tc=101774496981913000 ts=0 d=101774496981913000]
```

```
     *****LSW WOULD RESET WDT WITH 10000000*****
     .
     <3p.
     <2p.....w1>..************************
     <0p..w3>**********.*
     <1p.*************.******
     <3p.********************.*****.****************.********.w1>****.*******************
.w0>******.w3>***************************
     [0-0:tc=101774500991913000 ts=0 d=101774500991913000]

     [1-1:tc=101774500991934000 ts=0 d=101774500991934000]

     [2-2:tc=101774500991943000 ts=101774497521907000 d=3470036000]

     [3-3:tc=101774500991952000 ts=0 d=101774500991952000]

     Task 2 (2p) continues to output in its hung state while
     seconds pass.

     *****LSW WOULD RESET WDT WITH 10000000*****
     ***
     <3p.**
     <1p.***********.*****.********************.*******.******************.w1>*********.
w3>*******
     <0p.************.*********
     <3p.********************.
     <1p*.***********.****************.***.*********.***************.w3>.w0>*********.w1>
**********************
     <3p.******************.**
     [0-0:tc=101774505001961000 ts=0 d=101774505001961000]

     [1-1:tc=101774505001985000 ts=0 d=101774505001985000]

     [2-2:tc=101774505001994000 ts=101774497521907000 d=7480087000]

     [3-3:tc=101774505002003000 ts=101774504641944000 d=360059000]

     *****LSW WOULD RESET WDT WITH 10000000*****
     ******************
     <1p.**********..************************.*w3>**.*********
     <0p.***********.*****.w1>***
     <3p.*****************.*****************************.*.*****
     <1p.************.*******.w3>****.w0>*****************.********
     <3p.****************.w1>****.************************
     [0-0:tc=101774509011930000 ts=0 d=101774509011930000]

     [1-1:tc=101774509012476000 ts=0 d=101774509012476000]

     [2-2:tc=101774509012489000 ts=101774497521907000 d=11490582000]

     [3-3:tc=101774509012541000 ts=101774508241921000 d=770620000]

     It takes 10 seconds before the task callback
     recognizes the task is hung and reports no progress. The Watchdog
```

```
function then orders a software reboot…

*****LSW WOULD REBOOT FSW*****
.000000000 LSW_CheckTaskProgress LSW-E-TASKHUNG TASK Task2 NOT MAKING PROGRESS
.000000000 LSW_RebootFSW LSW-E-REBOOT REBOOTING PROCESSOR
```

2 Demonstration Procedure

2.1 CCSDS Packet Demo

The CCSDS Packet Demo will be run at a demonstration terminal in the B-101 laboratory in Building 84.

The CCSDS Packet Demo shows that the CCSDS and CTDB software packages – required for use of the 1553 data bus and for transmission of CCSDS packets back and forth on that bus – are operational.

During the demo, command and telemetry message generation by the Spacecraft and LAT is simulated in software. However, real CCSDS packets are exchanged over a functioning 1553 communications hardware interface; and finalized, operational CCSDS and CTDB software that supports transmission of the packets between the simulated LAT and Spacecraft is used.

2.1.1 Context of the CCSDS Packet Demo

- Two crates are configured and connected together for this demonstration:
  - The first crate is configured with an mv2304 processor, a PMC 1553 communications board (operating as a Bus Controller), and a PMC LAT Communications Board (LCB). This crate serves as the Spacecraft.
  - The second crate is configured with a RAD750 processor, a 1553-capable Spacecraft Interface Board (SIB), and a PMC LCB.
- The CCSDS and CTDB software packages, which provide the infrastructure for the telecommand and telemetry system, are installed on the RAD750 crate.
- In addition to the CCSDS and CTDB packages, the PBS and MSG packages are installed on the RAD750 crate.
- Spacecraft simulator code is loaded on the mv2304 “Spacecraft” crate.

- The crates are running a developer edition of VxWorks that provides terminal emulation and other features not installed in the flight configuration.

2.1.2 CCSDS Packet Demo Procedure

- The “Spacecraft” crate is powered up before the demonstration session begins.
- The tester logs on to a Unix host at the demonstration terminal and opens a VxWorks shell to the “Spacecraft” crate.
- On the “Spacecraft” mv2304 crate, from VxWorks, the tester issues the **SC_start** command to start the Spacecraft simulator.
- The tester boots the RAD750 crate. The crate executes primary boot code. When primary boot is complete, the crate enters a 10 minute timeout period.
- On the Spacecraft crate, the tester issues the **SC_send_boot** command to send a command packet instructing the RAD750 crate to leave the 10 minute timeout period and proceed immediately to the secondary boot stage.

2.2 GASU Configuration Demo

The GASU Configuration Demo will be run at a demonstration terminal in the B-101 laboratory in Building 84. The GASU Configuration Demo and the Single Event Display Demo are run simultaneously, in a single step.

The GASU (Global-Trigger/ACD-EM/Signal-Distribution Unit) houses several critical pieces of the Trigger & Dataflow electronics system:

- The GEM (Global trigger Electronics Modules). A data acquisition subsystem that hosts the Global Trigger function. The Global Trigger decides – based on analysis of trigger primitive data from the TKR and CAL, and signals from the ACD – whether detectors on the LAT should be read out. The GEM also contributes event data, sending its contribution to the EBM.
- The AEM (ACD Electronics Module), which is responsible for assembling event data from ACD front-end electronics boards and passing the data forward in the data acquisition process. ACD data is critical for determining whether events are *not* caused by gamma rays; ACD data is used as veto input in the Global Trigger’s decision whether the instrument should be read out.
- The EBM (Event Builder Module), which is responsible for assembling event data from the TKR, CAL, and ACD electronics modules, as well as the GEM; determining when the capture process for an event is “complete”; and forwarding the fully-assembled event data set to CPUs for further processing. The EBM also passes event data among CPUs during processing or write data to the Solid State Recorder.
- The CRU (Command/Response Unit), which acts as a router for LAT command/response packet traffic.

Flight Software that commands and configures these hardware elements is correspondingly critical. The GASU Configuration Demo has three goals:

- Show that the DAB software package communicates with the GEM and EBM and the DEM package communicates with the AEM over the command/response network fabric through the CRU.

- Show that DAB and DEM correctly construct bit strings for write operations to GEM, AEM, and EBM registers.
- Show that the LCB (LAT Communications Board drive) package, required for all communications between CPUs and other data acquisition hardware, is operational.

2.2.1 Context of the GASU Configuration Demo

- A crate is configured with RAD750 processor, a cPCI LAT Communications Board (LCB) interface, a Spacecraft Interface Board (SIB), and a serial line. The crate is connected through its LCB to the CRU, and through the CRU, to the GEM, EBM, and AEM.
- The crate is running a developer edition of VxWorks that provides terminal emulation and other features not installed in the flight configuration.
- The DEM and DAB packages, which handle communications with the GASU (CRU, GEM, EBM, AEM) and read and write bit strings, run on the crate.
- Aside from the DEM and DAB packages, the LCB, DUTIL, EVUT, VXW, PBS, MSG, and CMX packages are installed on the crate.

2.2.2 GASU Configuration Demo Procedure

The configuration demo code exercises selected readable/writeable registers on the CRU, GEM, EBM, and AEM.

- The RAD750 crate is powered up before the demonstration session begins.
- The GASU is powered up by the Power Distribution Unit (PDU) in the GASU.
- The tester issues a switchOn command to power up the PDU and GASU.
- The tester runs a VxWorks script on the RAD750 that:
  - Loads FSW package libraries (see above for list) and library files for the demo.
  - Initializes the RAD750 crate, including its LCB.
  - Configures registers on the CRU, EBM, and GEM.
  - Registers an event handler with the LCB.
- The tester sends the trigger command.

2.3 Single Event Display Demo

The Single Event Display Demo will be run at a demonstration terminal in the B-101 laboratory in Building 84. The GASU Configuration Demo and the Single Event Display Demo are run simultaneously, in a single step.

The Single Event Display Demo has five goals:

- To show that commands can be sent from a RAD750 crate to the GEM over the LAT command/response network fabric. In this case, a trigger command is sent to the GEM.
- To show that the GEM, AEM, and EBM subsystems in the GASU function together and communicate over the LAT command/response and event network fabrics.

- To show that the GEM successfully transmits a Trigger Accept Message (TAM) to the AEM following the solicited trigger command.
- To show that the GEM and AEM both successfully read out their event data in response to the Trigger Accept Message and transmit event data packets to the EBM.
- To show that the EBM collects and assembles event data from the GEM and AEM, and can transmit the assembled data forward in the processing chain.

2.3.1 Context of the Single Event Display Demonstration

- The same RAD750 crate used in the GASU Configuration Demo is used for Single Event Display.
- The same GASU components (CRU, GEM, EBM, and AEM) used in the GASU Configuration Demo are used for Single Event Display.
- The demo code library and the DEM, DAB, LCB, DUTIL, VXW, PBS, MSG, and CMX packages are installed on the RAD750 crate.
- The crate is running a developer edition of VxWorks that provides terminal emulation and other features not installed in the flight configuration.

2.3.2 Single Event Display Demonstration Procedure

- The GASU Configuration Demo procedure described above includes the step (issuing the solicited trigger command using a VxWorks script) that begins event delivery for the Single Event Display Demo.

2.4 Watchdog Demo

The Watchdog function demo will be run at the demonstration terminal in Building 84, Room B-101. Two constituents of the LSW package, **lsw** and **lswTest**, are run on a RAD750 CPU to monitor the functioning of 4 VxWorks test tasks.

- While these test tasks are making progress, as measured by their single cycle processing times, the Watchdog software function resets a register on the RAD750's hardware watchdog to prevent the CPU from rebooting.
- When task 2 is deliberately halted and then fails to make progress (becomes "hung"), the Watchdog software function reboots the flight software system.

2.4.1 Context of the Watchdog Demo

- The Watchdog function demo runs on the same RAD750 crate used in the CCSDS Packet Demo.
- The RAD750 crate is running a developer edition of VxWorks that provides terminal emulation and other features that will not be installed in the flight configuration.
- Constituents of the CMX package (cmx_asBuiltSpy), VWX package (vmx_extra), LSW package (lsw, lswTest), PBS, MSG, and simple print routines are installed on the RAD750.

2.4.2 Watchdog Demo Procedure

- The RAD750 crate is powered up and VxWorks is loaded during the CCSDS Packet Demo, before the Watchdog demonstration begins.
- Tornado is launched to provide access to the VxWorks shell.
- The **ld_lsw** script is run at the VxWorks prompt. This script loads constituents of the CMX, VXW, and LSW packages (see above for the exact constituents).
- The **lswTest** test suite is launched at the VxWorks prompt and the Watchdog demo begins.

3 Demo Wrapup and Summary

The FSW Team thanks you for attending the demonstration and welcomes any questions about the software and hardware systems showcased. Comments or questions can be written in the following space.

4 Glossary

ACD (Anticoincidence Detector). LAT component that detects non-gamma ray events (energetic cosmic ray electrons and nuclei) for the purpose of vetoing data collection for such events.

AEM (ACD Electronics Module). A data acquisition subsystem in the GASU, which is responsible for assembling event data from ACD front-end electronics boards and passing the data forward in the DAQ process.

CAL (Calorimeter). LAT component that measures the energy of incident photons and background particles. These measurements, along with data collected by the TKR (see also), are used to reconstruct the energy of the incident photons.

Crate. Fond, generic term for development versions of Spacecraft Interface Units (SIUs) or Event-Processor Units (EPUs): custom-built, standalone on-board FSW processors and communications hardware units that control the LAT and communicate with the spacecraft (SIU), and process/filter instrument events (EPU). Crates are used for development purposes and will be replaced in the flight unit with single board computers with the same functionality.

EBM (Event Builder Module). A data acquisition subsystem in the GASU, which is responsible for assembling event data from the TKR, CAL, and ACD electronics modules, determining when the capture process for an event is “complete”, and forwarding the fully-assembled event data set to CPUs for further processing. The EBM also passes event data among CPUs during processing or writes data to the Solid State Recorder.

EPU (Event-Processor Units). A type of single board computer (SBC) in the FSW hardware suite that processes/filters event data.

GASU (Global-Trigger/ACD-EM/Signal-Distribution Unit). Portion of the FSW hardware suite that serves as the major hardware interface between data acquisition electronics on the LAT and other hardware and electronics that make up the FSW hardware package. The GASU contains the GEM, EBM, AEM, and CRU.

GEM (Global trigger Electronics Module). A data acquisition subsystem that hosts the Global Trigger function.

GLT (Global Trigger). A data acquisition subsystem that is responsible for deciding – based on analysis of trigger primitive data from the TKR and CAL, and signals from the ACD – whether detectors on the LAT should be read out.

PDU (Power Distribution Unit). Portion of the FSW hardware suite that manages power distribution from the spacecraft and monitors the health of other FSW hardware.

SIU (Spacecraft Interface Unit). A type of single board computer (SBC) in the FSW hardware suite that acts as an interface between the spacecraft and the LAT.

T&DF (Trigger and Dataflow System). Large LAT subsystem that provides gamma-ray identification, readout of the detector measurements, assembly of gamma-ray source location and energy measurements, and the streaming of data to the spacecraft. The T&DF subsystem contains the TKR, CAL, and ACD front-end electronics, Tower Electronics Modules (TEMs), ACD Electronics Modules (AEMs), Event Builder Module (EBM), Global Trigger (GLT), Global Trigger Electronics Module (GEM), and CPUs used for instrument configuration and data processing.

TEM (Tower Electronics Module). A data acquisition subsystem that is responsible for assembling event data from TKR and CAL front-end electronics boards and passing the data forward in the DAQ process. There are 16 TEMs in the final LAT configuration, one for each TKR/CAL tower.

TKR (Detector Tracker). LAT component that converts gamma rays to charged particles and measures with great precision the path of the charged particles through the tracker itself.

VxWorks. Computer operating system used on the board the RAD750 processor.

Attachment 1

The following diagram depicts the final, flight version of the GASU and the CPUs with which it communicates. For today's demonstrations, a subset of this hardware will be exercised:

- The Global trigger Electronics Module (GEM)
- The ACD Electronics Module (AEM)
- The Event Builder Module (EBM)
- The Command/Response Unit (CRU)
- A single RAD750 CPU crate configured with both SIU and EPU functionality (a special-purpose configuration that will never be used in the flight unit)