GLAST LAT Flight Software Demonstration

SLAC Campus
Building 84, Central Lab Annex
Room B-101
January 30

EM2 Peer Review
January 29-30, 2004

1 Demonstration Overview

1.1 Agenda for the Demonstrations

The demonstrations will take place in the Central Laboratory Annex (Building 84), Room B-101.

Hour One			
Demo Agenda Item	**Presenter(s)**	**Relevant FSW Specs**	**Output of Demonstration**
1. Introductions	NA	NA	NA
2. Overview of the Demonstration	Terry Schalk JJ Russell	NA	NA
3. Software/Algorithm Demonstrations	JJ Russell	5.2.2.4	Test suite output for 3 core software packages and the data filter package ported to VxWorks and running on a RAD750 processor: ____ PBS package: Test suites demonstrate process control, interprocess communication, thread control, and wakeup timers. ____ MSG package: Test suites demonstrate messaging capabilities. ____ CMS package: Tests demonstrate software versioning control on board the RAD750. ____ Filter package: Tests demonstrate filtering of simulated data.
4. Programmable Input/Output Device Demonstrations	JJ Russell	5.3.4.3.4, 5.3.1.2, 5.3.1.1	Demonstrate use of PID input and output channels to communicate the spacecraft and perform timekeeping functions: ____ PID Demo One: RAD750 uses PID channels 5 and 6 to communicate with a spacecraft simulator. ____ PID Demo Two: Spacecraft uses PID channels 23, 24, 25 to communicate with the RAD750. ____ PID Demo Three: PID channels 8 and 9 are used to provide clock data.
5. Questions from Attendees			

Hour Two (after 5 minute break)			
6. Storage and Interface Board Demonstrations	Dan Wood	5.3.1.3, 5.3.4.2.4	____ SIB Demo One: The SIB will execute a primary power bootstrap operation to power first the PDU, and through switches in the PDU, the GASU. ____ SIB Demo Two: Simple commands are used to access the SIB's on-board EEPROM, list files stored there, and dump the contents of the files to the screen. ____ SIB Demo Three: Commands and outputs at the demonstration terminal will show that 1553-based messages have been sent successfully between two single board computers, one a COTs board acting a Bus Controller, the other a RAD750 acting as a Remote Terminal.
7. Questions from Attendees and Wrapup			

As indicated above, the agenda allows time for questions from the attendees. Feel free to jot questions and comments down in the margins of this document or in the space provided on page 24.

1.2 Overview of Flight Software (FSW) Subsystems and How They Function In Data Acquisition

Flight Software (FSW), which is formally part of the LAT's data acquisition (DAQ) subsystem, plays the critical role of capturing data from the instrument, processing and filtering that data, and outputting the processed data for subsequent transmission to the ground.

As its name suggests, the Flight Software system is comprised of software algorithms, but also of custom computing and electronics hardware modules which serve as the platform on which this software is executed, and which are responsible for programmable input and output of data, intermediate storage of data, and the interface between the LAT and the spacecraft. The major FSW subsystems are:

- **Crates.** "Crates" are the computational core of the Flight Software system. Microprocessors and customized input/output boards are packaged ("crated") into standalone units, each containing 3 main components: a Storage and Interface Board (SIB), a customized RAD750 CPU, and a LAT Communication Board (LCB). Crates come in 2 flavors, *Spacecraft Interface Units* (SIUs), which provide control of the LAT and integrate it with the spacecraft and *Event-Processor Units* (EPUs), which handle data recorded by the LAT. The final design includes a primary SIU, multiple EPUs, and multiple backups for each.

- **Global-Trigger/ACD-EM/Signal-Distribution Unit (GASU)**. The GASU serves as the major hardware interface between data acquisition electronics on the LAT and the hardware and electronics that make up the FSW hardware package.
- **Power Distribution Unit (PDU).** PDU hardware manages power distribution from the spacecraft and monitors the health of other FSW hardware.
- **Code Library**. The complete library of FSW code is currently organized into multiple "projects", each of which contains code packages that serve either on-board flight functions or assist with development. Flight Software projects include DAQ (code for data acquisition), DRV (important low-level code used in communication), SVC (basic services), and OS (including boot code that manages hardware powerup via the PDU and the handoff of control to the operating system during the boot process). These flight code libraries are designed to run under the VxWorks operating system, the native crate operating environment.

The capabilities of many of these FSW software and hardware subsystems will be demonstrated today.

The Flight Software software/hardware package receives instrument data after it is captured by the 4 major subcomponents of the telescope itself: the Anticoincidence Detector (ACD), Tracker (TKR), Calorimeter (CAL), and Trigger and Dataflow System (T&DF). Of these telescope subsystems, the Trigger and Dataflow system is the one most closely tied (through DAQ electronics modules on the instrument) to the Flight Software system. The relationship among these systems is shown diagrammatically in Figure 1 below.

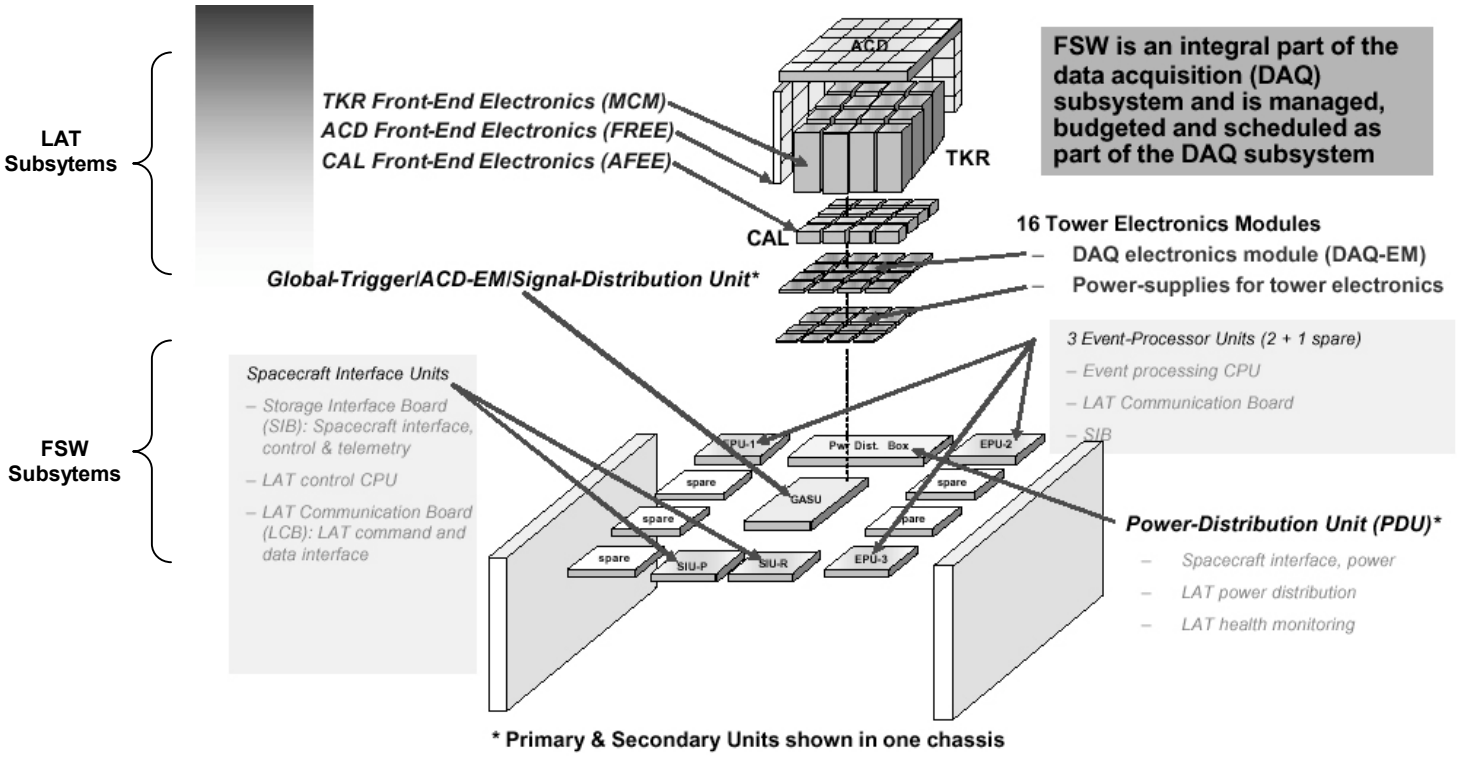


Figure 1: Overview of LAT Subsystems and Flight Software Subsytems

Thus, the algorithms and hardware of the Flight Software system capture raw data from the telescope, filter and verify that data, and pass the data on to the spacecraft for communication with researchers back on the ground.

1.3 Brief Overview of Subsystems to be Demonstrated

Today's demonstration covers several important deliverables of the Flight Software system, most importantly, functioning crates built around RAD750 processors. Generally speaking, the demo will work "inside out", starting from the RAD750 as an isolated system within a crate, then moving outward to show how the crates can operate and communicate in a fully functioning hardware context that includes a simple "spacecraft", power management hardware (a working PDU), and an interface to the LAT (a working GASU):

- First are the **Algorithm/Software Demos** showing several tools from the Code Library run on an SIU crate.
- Next are the **Programmable Input/Output Demos**. These demonstrations show programmable crate communications hardware in operation.
- Finally, the **Storage and Interface Board Demos** show hardware that has a working non-volatile memory capability and the ability to use the 1553 communications bus.

1.4 Goals of the Demonstrations

The demonstration will achieve the following goals:

- Overall, the demo will show that a fully functioning crate, incorporating a RAD750 CPU and SIB, has been successfully built.
- The demo will show that the crate contains a working Storage and Interface Board (SIB) that communicates with the RAD750 within the crate, and with 1553-compliant devices outside the crate.
- The demo will show that the crates have operating backplanes and therefore function in a larger hardware context that includes a spacecraft, GASU, and PDU.
- The Algorithm/Software Demonstration will show that several critical, low-level systems have been fully ported to VxWorks, the RAD750 native environment.
- The Programmable Input/Output Device Demos will show that RAD750-based crates have a functioning backplane of I/O devices and communicate with the spacecraft in customizable ways.
- The Storage and Interface Board Demos will show that the crates have fully functional, readable and writeable non-volatile memory; that through a working backplane, the crates are control the PDU to execute power-up operations; and that they are able to utilize the 1553 communications bus.

2 The Software Demonstrations

The Code Library contains 9 major projects and dozens of associated packages. Figure 2 below shows a matrix that maps FSW packages onto major areas of functionality.

Function	CCSDS	CLI	CMX	CTDB	DAB	DEM	EXPAT	FILE	GGLT	GNAT	GRL	LATC	LCB	MSG	OCS	OES	PBC	PBI	PBS	VXW	XLX	ZLIB
Configure TKR and CAL front end electronics			Y			Y							Y	Y				Y	Y	Y		
Configure ACD front end electronics			Y			Y							Y	Y				Y	Y	Y		
Configure GASU (CRU, GEM, EBM, AEM)			Y		Y								Y	Y				Y	Y	Y		
Configure PDU			Y		Y								Y	Y				Y	Y	Y		
Configure XBRD			Y						Y	Y			Y	Y				Y	Y	Y		
Configure by compressed file			Y				Y					Y		Y				Y	Y	Y	Y	Y
Real event delivery (instrument to CPU)			Y		Y	Y			Y				Y	Y	Y	Y		Y	Y	Y		
Housekeeping data stream		Y	Y		Y	Y			Y		Y		Y	Y				Y	Y	Y		
RAD750 boot and crate initialization	Y	Y	Y	Y				Y			Y			Y			Y	Y	Y	Y		Y
1553 bus communications	Y	Y	Y								Y			Y				Y	Y	Y		
Telecommand/telemetry database and services			Y																			
Emulated event delivery (to science data interface)			Y																			
CPU internal communications/task frameworks			Y																			
Software watchdog			Y																			
Wall clock time services (GPS)			Y																			

Figure 2: Function/Package Mapping for FSW Software Packages

The Algorithm/Software Demos will run test suites for three important core operating packages from the SVC (basic services) project:

- **PBS Package.** This package provides routines for computing in real time, including routines for process/task/thread control, byteswapping, timers and processor clocks, memory allocation, and resource control. It and the MSG package are the lowest-level FSW packages, functioning just above the kernel itself, and serve as the foundation for all other FSW code. PBS capabilities shown during this demonstration are defined on page 7. Output of the PBS package tests are defined on page 8.
- **MSG Package.** This package provides low-level messaging and error handling capabilities. It and the PBS package are the lowest-level FSW packages, functioning just above the kernel itself, and serve as the foundation for all other FSW code. The capabilities shown during this demonstration are defined on page 11. Output of the MSG package tests are defined on page 12.
- **CMX Package.** This package provides routines across the functionality map shown above. The capabilities shown during this demonstration are defined on page 13. Output of the CMX package tests are defined on page 14.

Also demonstrated from the DAQ (data acquisition) project:

- **Filter Package.** This package provides filtering functions that allow the FSW system to access data from the Global Trigger, ACD, CAL, and TKR systems on the instrument, then use the data as inputs for trigger descriminator tests, energy-level tests, and track-projection

tests on raw instrument data. The capabilities shown during this demonstration are defined on page 14. Output of the filter package tests are defined on page 15.

The Algorithm/Software Demos will achieve the following goals:

- Show that code that has been under development on other platforms is fully portable to the RAD750 and important packages have already been ported.
- Show that the demonstrated software performs stably in the native RAD750 VxWorks environment.
- Show that software that supports real-time computing on the RAD750 is complete.
- Show that software that supports messaging on the RAD750 is complete.
- Show that software that supports in-flight software versioning control on the RAD750 is complete.
- Show that software is able to successfully and reproducibly filter simulated instrument data according to dimensions of interest (trigger discrimination, energy level, and track projection) and at required performance levels.

2.1 The PBS Package Demo

Three scripts from the PBS test suite will be run at the demonstration terminal in Room B-101:

- **test_fork:** This script forks a process and echoes communications between the parent and child process.
- **test_task:** This script demonstrates thread control: it spawns multiple threads from a process, synchronizes a parent thread to an existing child thread, and demonstrates that the threads can be successfully assigned priorities, can be paused, be suspended, and resumed.
- **test_wut:** This script demonstrates wakeup timers, fine-grain timers useful on the scale of 50 µsecs or greater. These PBS timers replace the VxWorks timer facility, which is fairly accurate to roughly 1 msec, but is usually used for intervals of 20 msecs. This test script also establishes the absolute time scale. It is timed against the wall clock, proving that the processor's concept of 1 second is correct.

Context of the PBS Demonstration:

- The PBS test scripts are run on the RAD750 processor in a crate configured as an SIU.
- The RAD750 is running a developer edition of VxWorks that provides terminal emulation and other features that will not be installed with the RAD750 in the flight configuration.
- The PBS, MSG, CMX, filter packages, and print routines are installed on the RAD750.
- The crate is connected to a simulated spacecraft, to a COTs test crate, to the GASU, to the Tower Electronics Module, and to the PDU.

2.1.1 Sample Output of Scripts from the PBS Test Suite

Sample test_fork output

```
PBS initialization
------------------
Number of wut tmrs: 256
Keepalive   period: 20000000
Update      period: 1000000000
PTS_initialize    : 2 cpus
Frequency         : 1390710000.000 Hz
Nseconds / ptu    :         0.719 nsecs

PARENT: Queing the done message...
PARENT: Waiting till child is done
CHILD : Name is F0
CHILD : Message from getChild priority: 0
CHILD : Message from specific callback: test_fork
CHILD : Message from default  callback: test_fork
CHILD : Message from quit     callback: Winding down child process
PARENT: Child is done
        Parent join status = 00000000 (expect 0x00000000)
        Child  exit status = ffffffff (expect 0xffffffff)
PARENT: Disabling the FORK que
PARENT: Queing the messages
PARENT: Waiting two seconds before renabling the que
PARENT: Reenabling the que
PARENT: Queing the done message...
PARENT: Waiting till child is done
CHILD : Name is F1
CHILD : Message from getChild priority: 0
CHILD : Message from specific callback: test_fork
CHILD : Message from default  callback: test_fork
CHILD : Message from quit     callback: Winding down child process
PARENT: Child is done
        Parent join status = 00000000 (expect 0x00000000)
        Child  exit status = ffffffff (expect 0xffffffff)
```

Sample test_task output

```
PBS initialization
------------------
Number of wut tmrs: 256
Keepalive   period: 20000000
Update      period: 1000000000

The initial priority of the parent task is: 0
The new     priority of the parent task is: 10
The priority of the spawned task will be  : 4
TASK_spawn  status is                     : 0

       Into the spawned thread
       The file name of this executable is: test_task
       The name     of the spawned task is: T1
       The priority of the spawned task is: 4
       The identity of the spawned task is: PARENT

       TASK_pause test      Waiting 2 secs:   96575004.451047000
       ............................ done:   96575006.470477000
       TASK_suspend test    Waiting 2 secs:   96575006.470517000
       Message pointer at user location  x:   00026510
       Message from parent to child       :   BE GOOD
       TASK_resume called... and completed:   96575008.490944000

       The identity of the current task is: PARENT

TASK_join status is                       : 0
The spawn  task's exit status is          : deadbeef

----------------------------------------

TASK_create   status is                   : 0
TASK_activate status is                   : 0

       Into the spawned thread
       The file name of this executable is: test_task
       The name     of the spawned task is: T3
       The priority of the spawned task is: 4
       The identity of the spawned task is: PARENT

       TASK_pause test      Waiting 2 secs:   96575008.491698000
       ............................ done:   96575010.500427000
       TASK_suspend test    Waiting 2 secs:   96575010.500466000
       Message pointer at user location  x:   00026510
       Message from parent to child       :   BE GOOD
       TASK_resume called... and completed:   96575012.521005000

       The identity of the current task is: PARENT

TASK_join     status is                   : 0
The activated task's exit status is       : abadcafe
```

Sample test_wut output

```
PBS initialization
------------------
Number of wut tmrs: 256
Keepalive   period: 20000000
Update      period: 1000000000

TOV = 4616583994962717008 1074882223.125938000
TOV = 4616583999257684304 1074882224.125938000
Wait  = 1000000000
  Address  Count TOV   secs.nsecs     TIM   secs.nsecs
 -------- ------ ----------.--------- ----------.---------
  Timeout                                 96575023.390667000
  Timeout                                 96575023.650537000
  Timeout                                 96575023.910555000
 000262b8      1 1074882224.126077000   96575024.130562000
  Timeout                                 96575024.390554000
  Timeout                                 96575024.650551000
  Timeout                                 96575024.910565000
 000262d8      2 1074882225.126077000   96575025.130570000
  Timeout                                 96575025.390577000
  Timeout                                 96575025.650572000
  Timeout                                 96575025.910605000
 000262f8      3 1074882226.126077000   96575026.130560000
  Timeout                                 96575026.390577000
  Timeout                                 96575026.650581000
  Timeout                                 96575026.910580000
 00026318      4 1074882227.126077000   96575027.130583000
  Timeout                                 96575027.390599000
  Timeout                                 96575027.650590000
  Timeout                                 96575027.910590000
 00026338      5 1074882228.126077000   96575028.130600000
  Timeout                                 96575028.390600000
  Timeout                                 96575028.650672000
  Timeout                                 96575028.910589000
 00026358      6 1074882229.126077000   96575029.130596000
  Timeout                                 96575029.390593000
  Timeout                                 96575029.650592000
  Timeout                                 96575029.910601000
 00026378      7 1074882230.126077000   96575030.130575000
  Timeout                                 96575030.390596000
  Timeout                                 96575030.650678000
  Timeout                                 96575030.910629000
 00026398      8 1074882231.126077000   96575031.130617000
  Timeout                                 96575031.390615000
  Timeout                                 96575031.650812000
  Timeout                                 96575031.910633000
 000263b8      9 1074882232.126077000   96575032.130632000
  Timeout                                 96575032.390672000
  Timeout                                 96575032.650654000
  Timeout                                 96575032.910684000
.
.
.
```

2.2 MSG Package Demo

The MSG test suite will be run at the demonstration terminal in Room B-101. The test suite uses the MSG package to run tests of the message system during various state transitions, start and stop the message system, configure the message system, test messages in various data formats, test how the system handles nonsense messages, and test buffering of messages. The suite ends by testing the capacity of the message system. As currently configured and run on the RAD750, MSG is capable of processing up to 30,000 messages per second.

Context of the MSG Demonstration:

- The MSG test suite runs on the RAD750 processor in a crate configured as an SIU.
- The RAD750 is running a developer edition of VxWorks that provides terminal emulation and other features that will not be installed with the RAD750 in the flight configuration.
- The PBS, MSG, CMX, filter packages, and print routines are installed on the RAD750.
- The crate is connected to a simulated spacecraft, to a COTs test crate, to the GASU, to the Tower Electronics Module, and to the PDU.

2.2.1 Sample Output of the MSG Test Suite

```
Begin MSG shotgun test
========================================================================
Begin configuration of the MSG system
------------------------------------------------------------------------------
Action: Check initial state of MSG system
  Test:                            control state:0x00000000 == 0x00000000 pass
------------------------------------------------------------------------------
Action: Initialize the MSG system
  Test:                             status value:0x7f0333a8 == 0x7f0333a8 pass
------------------------------------------------------------------------------
Action: Attach specialized output processor (testShotgun_cb)
  Test:                             status value:0x7f0333a8 == 0x7f0333a8 pass
------------------------------------------------------------------------------
Action: Start the MSG system
  Test:                             status value:0x7f0333a8 == 0x7f0333a8 pass
==============================================================================
End configuration of the MSG system                                      pass
==============================================================================
Begin shotgun accumulation phase (first parameter set)
==============================================================================
End shotgun test accumulation phase                                      pass
==============================================================================
Begin rundown of MSG system
------------------------------------------------------------------------------
Action: Stop the MSG system
  Test:                             status value:0x7f0333a8 == 0x7f0333a8 pass
------------------------------------------------------------------------------
Action: Shut down the MSG system
  Test:                             status value:0x7f0333a8 == 0x7f0333a8 pass
==============================================================================
End rundown of MSG system                                                pass
==============================================================================
Begin analysis: task/pkts/time/tmin/tmax 8/16/10/50000/100000
------------------------------------------------------------------------------
Action: Correlate statistics
  Test:                Messages Tx = Rx (task  0):0x00000301 == 0x00000301 pass
  Test:                Messages Tx = Rx (task  1):0x000002fd == 0x000002fd pass
  Test:                Messages Tx = Rx (task  2):0x000002e6 == 0x000002e6 pass
  Test:                Messages Tx = Rx (task  3):0x0000030f == 0x0000030f pass
  Test:                Messages Tx = Rx (task  4):0x0000030f == 0x0000030f pass
  Test:                Messages Tx = Rx (task  5):0x00000316 == 0x00000316 pass
  Test:                Messages Tx = Rx (task  6):0x000002d5 == 0x000002d5 pass
  Test:                Messages Tx = Rx (task  7):0x00000314 == 0x00000314 pass
  Test:             Blackout begin = Blackout end:0x00000000 == 0x00000000 pass
  Test:             Sum Tx dropped  = Rx measured:0x00000000 == 0x00000000 pass
  Test:             Sum Tx blackout = Rx measured:0x00000000 == 0x00000000 pass
  Test:                             No anomalies:0x00000000 == 0x00000000 pass
==============================================================================
End analysis of shotgun test output                                      pass
==============================================================================
Begin configuration of MSG system
------------------------------------------------------------------------------
Action: Check initial state of MSG system
```

```
  Test:                              control state:0x00000000 == 0x00000000 pass
-------------------------------------------------------------------------------
Action: Initialize the MSG system
  Test:                               status value:0x7f0333a8 == 0x7f0333a8 pass
-------------------------------------------------------------------------------
Action: Attach specialized output processor (testShotgun_cb)
  Test:                               status value:0x7f0333a8 == 0x7f0333a8 pass
-------------------------------------------------------------------------------
Action: Start the MSG system
  Test:                               status value:0x7f0333a8 == 0x7f0333a8 pass
===============================================================================
End configuration of the MSG system                                        pass
===============================================================================
Begin shotgun accumulation phase (second parameter set)
===============================================================================
End shotgun test accumulation phase                                        pass
===============================================================================
Begin rundown of MSG system
-------------------------------------------------------------------------------
Action: Stop the MSG system
  Test:                               status value:0x7f0333a8 == 0x7f0333a8 pass
-------------------------------------------------------------------------------
Action: Shut down the MSG system
  Test:                               status value:0x7f0333a8 == 0x7f0333a8 pass
===============================================================================
End rundown of MSG system                                                  pass
===============================================================================
Begin analysis: task/pkts/time/tmin/tmax 8/8/10/50000/100000
-------------------------------------------------------------------------------
Action: Correlate statistics
  Test:                Messages Tx = Rx (task  0):0x0000021d == 0x0000021d pass
  Test:                Messages Tx = Rx (task  1):0x00000298 == 0x00000298 pass
  Test:                Messages Tx = Rx (task  2):0x0000029e == 0x0000029e pass
  Test:                Messages Tx = Rx (task  3):0x00000295 == 0x00000295 pass
  Test:                Messages Tx = Rx (task  4):0x00000299 == 0x00000299 pass
  Test:                Messages Tx = Rx (task  5):0x00000292 == 0x00000292 pass
  Test:                Messages Tx = Rx (task  6):0x00000298 == 0x00000298 pass
  Test:                Messages Tx = Rx (task  7):0x0000029b == 0x0000029b pass
  Test:             Blackout begin = Blackout end:0x000001b7 == 0x000001b7 pass
  Test:             Sum Tx dropped  = Rx measured:0x000004a7 == 0x000004a7 pass
  Test:             Sum Tx blackout = Rx measured:0x000001b7 == 0x000001b7 pass
  Test:                               No anomalies:0x00000000 == 0x00000000 pass
===============================================================================
End analysis of shotgun test output                                        pass
*******************************************************************************
End MSG shotgun test                                                       pass
```

2.3 CMX Package Demo

The CMX test suite will be run at the demonstration terminal in Room B-101. Two constituents of the CMX packages are run on the RAD750, cmx_asBuilt and cmx_asBuiltSpy, to determine which packages are installed and reports package contents, control versioning system (CVS) tags, build authors, build locations, and build dates.

Context of the MSG Demonstration:

- The CMX test suite runs on the RAD750 processor in a crate configured as an SIU.
- The RAD750 is running a developer edition of VxWorks that provides terminal emulation and other features that will not be installed with the RAD750 in the flight configuration.
- The PBS, MSG, CMX, filter packages, and print routines are installed on the RAD750.
- The crate is connected to a simulated spacecraft, to a COTs test crate, to the GASU, to the Tower Electronics Module, and to the PDU.

2.3.1.1 Sample Output of the CMX Test Suite

```
> CMX_asBuiltPrint

Package     Constituent     CVS tag     Built by  Built at  Build Time (UTC)
----------- --------------- ----------- --------- --------- -------------------
CMX         cmx_asBuilt     V2-0-6      kiml      slac      2004-01-12 22:41:27
            cmx_asBuiltSpy  V2-0-6      kiml      slac      2004-01-12 22:41:27
PKGA        link_test       <test>      apw       slac      2003-06-13 19:17:25
            pkga            <test>      apw       slac      2003-06-13 19:17:25
PKGB        pkgb            <test>      apw       slac      2003-06-13 19:17:05
PKGC        pkgc            <test>      apw       slac      2003-06-13 19:16:43
```

2.4 Filter demo

The filter test suite will be run at the demonstration terminal in Room B-101.

Context of the Filter Demonstration:

- The filter test suite runs on the RAD750 processor in a crate configured as an SIU.
- The RAD750 is running a developer edition of VxWorks that provides terminal emulation and other features that will not be installed with the RAD750 in the flight configuration.
- The PBS, MSG, CMX, filter packages, and print routines are installed on the RAD750.
- The crate is connected to a simulated spacecraft, to a COTs test crate, to the GASU, and to the PDU.

The Simulated Event File. The filter demonstration takes a data file of 1000 simulated events as input generated using Monte Carlo simulations. The simulated data is presented to the filter program in the same format as the filter program expects to receive the data from the event builder in the real LAT, thus the filter package has no way of determining whether this is simulated or real data. Each event is composed of 18 contributors, presented in the following order:

- 1 from the GEM (Global Trigger Electronics Module),
- 16 from the 16 towers
- 1 from the ACD

The output of the filter program shows the data load (in bytes) of each. See Sample Output of the Filter Test Suite below.

2.4.1 Sample Output of the Filter Test Suite

```
-> filter 0,&EBD_11001_1000,0,0,1000,-1
value = 0 = 0x0
-> TMR_initialize
Measured Frequency:  7732632 Hz
Nominal  Frequency:  8333333 Hz
Nseconds / tick   :      120 nsecs
Input  file: (null)
Processing : 1000
Skipping   : None
Printing   : None
Geometry Id: Default
Veto Mask  : 7fff8000
  Id  Geo Type         CMT Tag     Created     Revised
 ---  --------  --------------  ----------  ----------
   2  GLEAM MC       v1/r13/p0   1/ 9/2004   1/ 9/2004
File Size  :   953072 bytes
Event Count:     1000

 Subsystem  Total Bytes Bytes/Event        CAL                TKR
 GEM              56000      56       Total   Bytes/      Total   Bytes/
 ACD              89664      89       Bytes   Event       Bytes   Event
 TEM[ 0]          37216      37        6888       6       30328      30
 TEM[ 1]          42784      42        7020       7       35764      35
 TEM[ 2]          39152      39        6800       6       32352      32
 TEM[ 3]          39840      39        7172       7       32668      32
 TEM[ 4]          37888      37        7212       7       30676      30
 TEM[ 5]          46128      46        8588       8       37540      37
 TEM[ 6]          42576      42        8628       8       33948      33
 TEM[ 7]          39792      39        7940       7       31852      31
 TEM[ 8]          40096      40        6992       6       33104      33
 TEM[ 9]          45520      45        9596       9       35924      35
 TEM[10]          45088      45        8380       8       36708      36
 TEM[11]          38320      38        7156       7       31164      31
 TEM[12]          38560      38        6124       6       32436      32
 TEM[13]          45696      45        7560       7       38136      38
 TEM[14]          42464      42        7924       7       34540      34
 TEM[15]          38288      38        6928       6       31360      31
 TEM[all]        659408     659      120908     120      538500     538
 Tot[noHdr]      805072     805
 Total           953072     953

TIMING      Elapsed Time   Nevts    Per Event
CAL0 Average:   3021.120 /  1000 =      3.000 usecs
ACD  Average:  15092.400 /   248 =     60.840 usecs
DIR  Average:   2972.280 /   199 =     14.880 usecs
ATF  Average:   1682.640 /   199 =      8.400 usecs
CAL1 Average:   4791.360 /   106 =     45.240 usecs
TKR0 Average:  15521.640 /    99 =    156.840 usecs
TOT  Average:  43081.440 /  1000 =     43.080 usecs
Elapsed Time:  45947.640 /  1000 =     45.960 usecs

Quantity(      All)   NONE <10Mev 300Mev 350Mev 500Mev  5 Gev > 5Gev Totals
------------------- ------ ------ ------ ------ ------ ------ ------ ------
```

ALL.....	477	43	287	13	26	122	32	1000
ACD Top Tile.....	285	25	161	3	7	48	18	547
ACD Side Tile.....	335	25	139	3	17	92	31	642
ACD Side Veto Tile	238	20	124	1	16	75	25	499
ACD >0 Veto Tiles..	417	39	248	4	17	96	27	848
ACD =0 Tiles Hit...	34	1	26	7	8	13	1	90
ACD =0 Veto Tiles..	60	4	39	9	9	26	5	152
Cal LO Trigger.....	.	.	32	10	18	115	32	207
CAL LO Trigger only	.	.	18	7	9	18	.	52
Cal Hi Trigger.....	.	.	.	.	.	2	18	20
Possible.....	1	.	5	.	.	8	2	16
Trigger.....	443	43	263	6	17	94	29	895
TKR Throttle.....	351	30	190	6	12	88	27	704
ALL.....	477	43	287	13	26	122	32	1000
NoCalLo + V Tile...	417	39	237	2	8	6	.	709
ACD Splash Veto 0..	.	.	2	.	2	30	9	43
ACD Evaluation....	60	4	48	11	16	86	23	248
Cal<350Mev + V Tile	.	.	9	2	.	.	.	11
Cal 0 Energy + Tile	26	3	.	.	.	.	.	29
ACD Splash Veto 1..	.	.	.	.	.	.	9	9
ATF Evaluation....	34	1	39	9	16	86	14	199
ACD Top Tile Veto.	.	.	.	.	.	15	.	15
ACD Side Tile Veto.	.	.	.	.	1	13	.	14
Zbottom.....	.	.	20	7	9	26	2	64
CAL LO only/ZBOT...	.	.	18	7	9	18	.	52
CAL1 Evaluation....	34	1	19	2	6	32	12	106
Cal E0/Etot < .01	.	1	3	.	.	.	.	4
Cal E0/Etot > .90	.	.	2	.	.	.	1	3
TKR Finding.....	34	.	14	2	6	32	11	99

Quantity(Survivors)	NONE	<10Mev	300Mev	350Mev	500Mev	5 Gev	> 5Gev	Totals
ALL.....	34	.	14	2	6	32	11	99
Cal LO Trigger.....	.	.	1	1	6	32	11	51
CAL LO Trigger only	.	.	.	.	.	.	.	.
Cal Hi Trigger.....	.	.	.	.	.	.	7	7
Possible.....	.	.	1	.	.	3	2	6
Trigger.....	34	.	13	2	6	29	9	93
TKR Throttle.....	.	.	7	2	2	22	8	41
ACD Top Tile.....	.	.	.	.	.	8	2	10
ACD Side Tile.....	.	.	7	2	5	27	10	51
ACD Side Veto Tile	.	.	.	.	5	20	5	30
ACD >0 Veto Tiles..	.	.	.	.	5	25	7	37
ACD =0 Tiles Hit...	34	.	7	.	1	2	1	45
ACD =0 Veto Tiles..	34	.	14	2	1	7	4	62
Tracks == 0.....	11	.	2	1	.	5	3	22

```
Tracks == 1........      9      .      4      .      2      1      2     18
Tracks >= 2........      4      .      5      1      2      8      6     26
Track/Top     Acd..      .      .      .      .      .      2      .      2
Track/Row 0,1 Acd..      .      .      .      .      2     11      .     13
Track/Row 2   Acd..      .      .      3      .      .      5      .      8
Skirt region.......     10      .      .      .      .      .      .     10
TKR < 2, E < 350Mev      9      .      4      .      .      .      .     13
Any veto...........     30      .      9      1      2     23      3     68
No  vetoes.........      4      .      5      1      4      9      8     31
```

3 The Programmable Input/Output Device Demonstrations

In flight configuration, the RAD750 has two interfaces:

- **PCI interface.**
    - PCI standard bus, 32 bits wide, 33 MHz
- **Programmable Input/Output Discretes**
    - Provided as a feature of the RAD750
    - Programmability includes
        - Input or output. Inputs can be configured to deliver interrupts.
        - Time/timer functions. Driven by internal or external clocks.

The programmable I/O discrete interface is the focus of this section of the demonstration program.

3.1 Overview of PIDs

For today's demonstration, the crate is configured as an SIU. PID assignments on the RAD750 are summarized in Figure 3 below. These assignments are shown for reference; not all of the

inputs/outputs are used in today's demonstration, but those that are demonstrated are highlighted in **bold text**.

#	Input Description	SIU	EPU	Input/Output
23-25	**SC Inputs 0-2**	**Y**		**Input**
21-22	GBM Interrupt Primary/Redundant	Y		Input
17	LCB PCI Interrupt	Y	Y	Input
16	SIB PCI Interrupt	Y		Input
15	Select Me	Y		Output
14	Clock Enable	Y		Output
11-13	External Clock – Primary	Y	Y	Input
8-10	**External Clock – Redundant**	**Y**	**Y**	**Input**
7	SC Primary/Redundant Selection	Y		Output
5-6	**SC Outputs 0-1**	**Y**		**Output**
2	LCB Interrupt – Alternate	Y	Y	Input
1	SIB Interrupt – Alternate	Y		Input

Figure 3: RAD750 PID Assignments

3.2 Summary of the PID Demo

The PID Demo is conducted in three parts, showing the following functionality:

- PID Demo One: Responding to a command issued from the demonstration terminal, the RAD750 will use PID channels 5 and 6 to communicate with a spacecraft simulator and turn an LED on the spacecraft on, then off. The output is described under 3.2.2.1 on page 19.
- PID Demo Two: After a switch on the front panel of the spacecraft simulator is thrown, the spacecraft will use PID channels 23, 24, 25 to communicate with the RAD750. The I/O from the spacecraft is hooked to an interrupt on the RAD750, which is monitored by a simple routine. When the interrupt is detected, the routine prints a message to the demonstration terminal. The output is described under 3.2.2.2 on page 20.
- PID Demo Three: Time hacks are sent through PID channel 9 while a 10 MHz clock derived by dividing the LAT 20 MHz system clock down drives PID channel 8. PID channel 8 is configured as an up counter, here driven by the 10 MHz clock. At each time hack, PID channel 9 captures the value of PID channel 8. The output is described under 3.2.2.3 on page 21.

3.2.1 Context of the PID Demo

The PID demonstration is run in the following software and hardware context:

- The RAD750 is running a developer edition of VxWorks that provides terminal emulation and other features that will not be installed with the RAD750 in the flight configuration.
- The PBS, MSG, CMX, filter packages, and print routines are installed on the RAD750.
- SU, to the

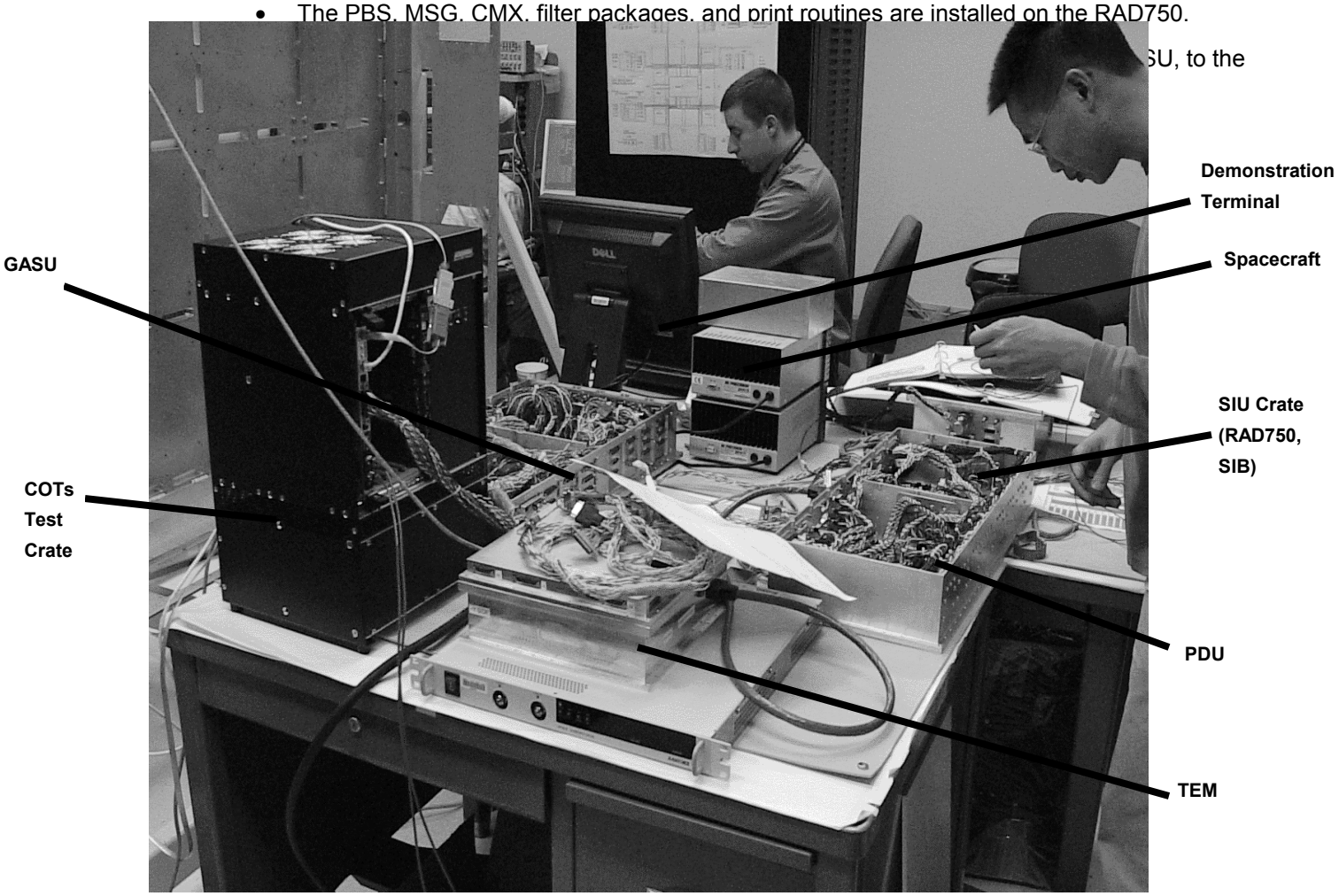


Figure 4: Full Hardware Setup for Flight Software Demonstration (Room B-101)

3.2.2 Output of the PID Demos

3.2.2.1 Output of PID Demo One

The RAD750 will use PIDs to communicate with the spacecraft and turn LEDs on and off. The "spacecraft" is shown in Figure 5 below. The backplane of the SIU crate is shown in Figure 6 below.

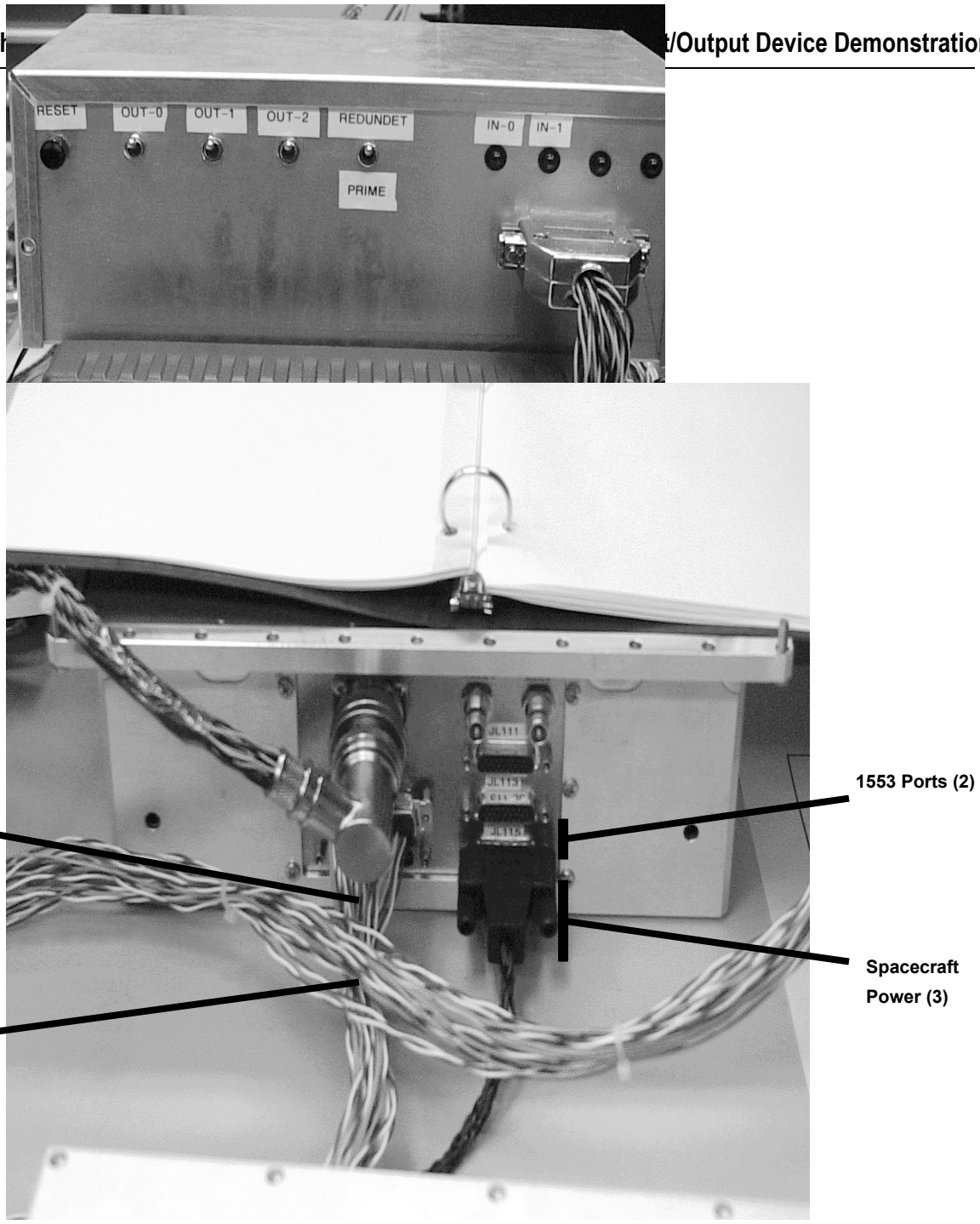


Figure 6: Backplane of the SIU Crate

3.2.2.2 Output of PID Demo Two

The Out-0, Out-1, and Out-2 switches on the spacecraft simulator will be thrown (see Figure 5 above). When these outputs trigger interrupts on the RAD750, a message is displayed at the demonstration terminal.

3.2.2.3 Output of PID Demo Three

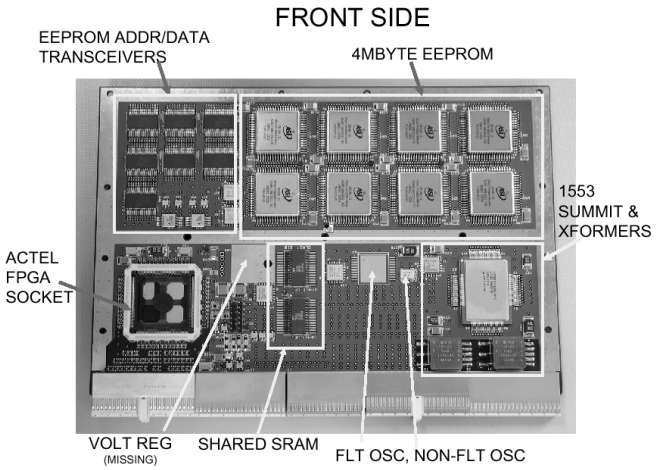
Time hacks are sent to the RAD750. Collated time information is displayed on the demonstration terminal.

4 The Storage and Interface Board Demonstrations

The Storage and Interface Board Demos will show that the SIU crates have fully functional non-volatile memory that can be read and written to; that through a working backplane, the crates are capable of controlling the PDU to execute power-up operations; and that they are able to utilize the 1553 communications bus.

4.1 Overview of the Storage and Interface Board (SIB)

The SIB integrates non-volatile memory (in the form of EEPROM), 1553 communications electronics, and interface electronics that talk to the GASU, and PDU. An SIB is shown in Figure 7 below.



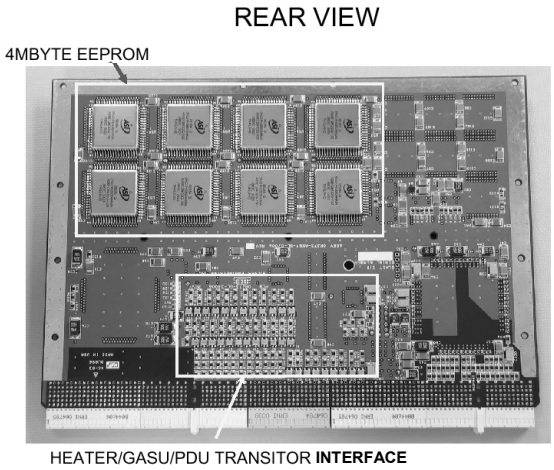
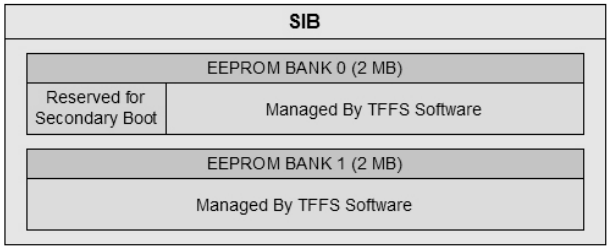


Figure 7: Storage and Interface Board (Front and Rear Views)

4.1.1 The File System for Non-Volatile Memory (EEPROM)

EEPROM is laid out on board the SIB as shown in Figure 8 below.



Layout In EEPROM Bank 0		
Address	**Use**	**Comment**
0x00000000	RTOS executable image (compressed)	An *absolutely* linked image
0x00060000	RTOS auxiliary module 0 (SSB0)	Second stage boot executable
0x00070000	RTOS auxiliary module 1 (SSB1)	Second stage boot script
0x00080000	Start of region managed by TFFS	Object & Configuration Files
0x00200000	End	

Figure 8: EEPROM Layout

The LAT will use a file system to read and write both RAM and, when data is ready to be committed to longer term storage, the SIB's EEPROM for storage of:

- Configuration files
- Startup & command scripts
- Object modules

This file system is managed by TFFS, a commercial product.

TFFS compensates for the fact that EEPROM is not infinitely writeable. It spreads the writes as evenly as possible, manages damaged memory using bad blocking techniques, and preserves logical continuity even if the file is not physically contiguous.

A 4 MB EEPROM with 1000 writes supports 4 GB lifetime. During a 5 year mission, can write ~2 MB/day.

Due to size restrictions of telecommand packets, file specifications are limited to 32-bits with sub-directory depth of 1.

4.1.2 1553 Communications

THE SIB implements the MIL_STD_1553B (1553) bus, which serves as the primary interface for exchanging information between SIU crates on the LAT and the spacecraft. Using this bus, the spacecraft can send commands and telemetry to the LAT, and can receive repoint requests from the LAT.

The spacecraft acts as bus controller (BC) node. Each SIU can act as remote terminal (RT) node. The bus protocol and schedule are under the control of the spacecraft.

All traffic over the bus will consist of CCSDS packets.

4.2 Summary of the SIB Demos

The SIB Demo is conducted in three parts, showing the following functionality:

- SIB Demo One: The SIB will execute a primary power bootstrap operation to power first the PDU, and through switches in the PDU, the GASU. The output is described under 4.4.1 on page 24.
- SIB Demo Two: From the demonstration terminal, basic “ls” and “cat” commands are used to access the SIB’s on-board EEPROM, list files stored there, and dump the contents of the files to the screen. The output is described under 4.4.2 on page 24.
- SIB Demo Three: Commands and outputs at the demonstration terminal will show that 1553-based messages have been sent successfully between two single board computers, one a COTs board acting a Bus Controller, the other a RAD750 acting as a Remote Terminal. In actual use, the bus controller is the Spacecraft or the Spacecraft Instrument Interface Simulator. The test proceeds as follows:

  1. Phase 1. Initialization occurs: the hardware is configured and checked by running a built-in test (BIS).

  2. Phase 2. The Bus Controller sends telecommand test packets to the Remote Terminal.

  3. Phase 3. The Remote Terminal posts telemetry and telecommand test packets to the Bus Controller.

  4. Phase 4. Finally, after the tests are completed, a summary of the statistics on the Remote Terminal is printed.

  The output is described under 4.4.3 on page 24.

4.3 Context of the SIB Demos

The SIB Demos are run in the following software and hardware context:

- The RAD750 is running a developer edition of VxWorks that provides terminal emulation and other features that will not be installed with the RAD750 in the flight configuration.

- The PBS, MSG, CMX, filter packages, and print routines are installed on the RAD750.
- The crate is connected to a simulated spacecraft, to a COTs test crate, to the GASU, to the Tower Electronics Module, and to the PDU, as shown in Figure 4 above.

4.4 Output of the SIB Demos

4.4.1 Output of SIB Demo One

Instruments are used to confirm that the PDU and GASU have powered up successfully based on instructions from the SIB.

4.4.2 Output of SIB Demo Two

After the ls command is issued, a directory listing of files stored on the EEPROM will appear on the demonstration terminal. After the cat command is issued, contents of one or more files in the directory listing will be dumped to the demonstration terminal.

4.4.3 Output of SIB Demo Three

The output for Phase 1 is display of the 1553 hardware registers The output for Phases 2 & 3 is a dump of the data packets by the receiving side. This is a pattern of incrementing 16 bit words. The output for Phase 4 is an ASCII display of the statistics block.

5 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.

_____

_____

_____

_____

_____

Again, to summarize the goals of the demonstration:

- Overall, the demo will show that a fully functioning crate, incorporating a RAD750 CPU and SIB, has been successfully built.
- The demo will show that the crate contains a working Storage and Interface Board (SIB) that communicates with the RAD750 within the crate, and with 1553-compliant devices outside the crate.
- The demo will show that the crates have operating backplanes and therefore function in a larger hardware context that includes a spacecraft, GASU, and PDU.
- The Algorithm/Software Demonstration will show that several critical, low-level systems have been fully ported to VxWorks, the RAD750 native environment.
- The Programmable Input/Output Device Demos will show that RAD750-based crates have a functioning backplane of I/O devices and communicate with the spacecraft in customizable ways.
- The Storage and Interface Board Demos will show that the crates have fully functional, readable and writeable non-volatile memory; that through a working backplane, the crates are control the PDU to execute power-up operations; and that they are able to utilize the 1553 communications bus.

6 Glossary

ACD (Anticoincidence Detector). LAT component that detects non-gamma ray events (energetic cosmic ray electrons and nuclei) for the purpose of removing these background events during observations. Flight Software uses data captured by this telescope subsystem to filter out noise and highlight events of interest.

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 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).

EPU (Event-Processor Units). A type of CPU crate in the FSW hardware suite that processes/filters instrument 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.

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 CPU crate in the FSW hardware suite that acts as an interface between the spacecraft and the LAT.

T&DF (Trigger and Dataflow System). LAT component 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.

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.