GLAST LAT Flight Software Demonstration

SLAC Campus
Building 84, Central Lab Annex
Building 80, Central Lab

February 27, 2004
10:30 AM

1 Demonstration Overview

1.1 Agenda for the Demonstrations

The demonstrations will take place in the Central Laboratory (Building 80) and the Central Laboratory Annex (Building 84).

Demo Agenda Item	Presenter(s)
1. Introductions	NA
2. Overview of the Demonstration	Curt Brune
3. LCB Package Test Demo	Sergio Maldonado
4. TKR and CAL Demos	Ramakrishnan P. Chirayathumadom
5. Watchdog Function Demo	Steve Mazzoni
6. 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 16.

1.2 Goals of the Demonstrations

The demonstration will achieve the following goals:

- The LCB Package Test Demonstration shows that the LCB software package runs on a CPU crate and interrogates the PCI hardware of a LAT Communications Board in that crate. This demo also shows that test scripts for the LCB package have been written and execute successfully on the LAT Flight Software Test Executive (LTX) system.
- The TKR and CAL Demonstrations show that the DEM software package, used to configure and capture event data from these LAT subsystems, is complete and operational. The software successfully writes and reads registers on a Tower Electronics Module (TEM). It can also be used to instruct the TEM to turn off selected TKR and CAL contributions to event data sets.
- 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.

1.3 Expected Outputs of the Demonstrations; Verification of Requirements

This section summarizes 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	Expected Output of the Demonstration
LCB Package Test Demo	5.3.4.6.1, 5.3.4.6.2	For a full description of the outputs of the LCB Package Test Demo, see Section 1.3.1 on page 3. For the demonstration procedure and the demo's software and hardware context, see Section 2.1 on page 11.
		Monitor Signature: _____
TKR and CAL Demos: **1) Register Write/Read Tests** **2) Event Tests**	5.3.4.6.1, 5.3.4.6.2	For a full description of the outputs of the TKR and CAL Demos, see Section 1.3.2 beginning on page 5. For the demonstration procedure and the demo's software and hardware context, see Section 2.2 on page 12. The Register Write/Read Test description begins in Section 2.2.1 on page 12. The Event Test description begins in Section 2.2.2 on page on page 13. These tests are detailed in document LAT-TD-02485.
		Monitor Signature: _____
Watchdog Function	5.2.1.2, 5.3.2.1	For a full description of the outputs of the Watchdog Function Demo, see Section 1.3.3 on page 7. For the demonstration procedure and the demo's software and hardware context, see Section 2.3 on page 15.
		Monitor Signature: _____

1.3.1 Output of the LCB Package Test Demo

The LCB Package Test Demo produces two kinds of output:

- Log file output that is displayed in terminal windows. 3 separate logs (an LTX session log, host log, and an LTX system log) are displayed.
- A complete LTX test record that can be displayed and reviewed in the LTX GIU, then saved to an LTX testing archive.

1.3.1.1 Expected Output of LTX Session Log for LCB Package Test

The LTX session log contains the most detailed output for the test, recording all initialization and loading steps and the final notice of test success or failure. In the figure below, records of initialization and loading are omitted for brevity.

```
.
.
.

LTXSYS>t_MemReg( lcb, 50000)

Starting Memory Space register read/write test.  50000 iterations ...


  LCB_TEST OK.


value = 0 = 0x0
.
.
.
```

Figure 1: Expected Output of LTX Session Log for LCB Package Test (Abbreviated)

1.3.1.2 Expected LTX Test Record for LCB Package Test Displayed in LTX GUI

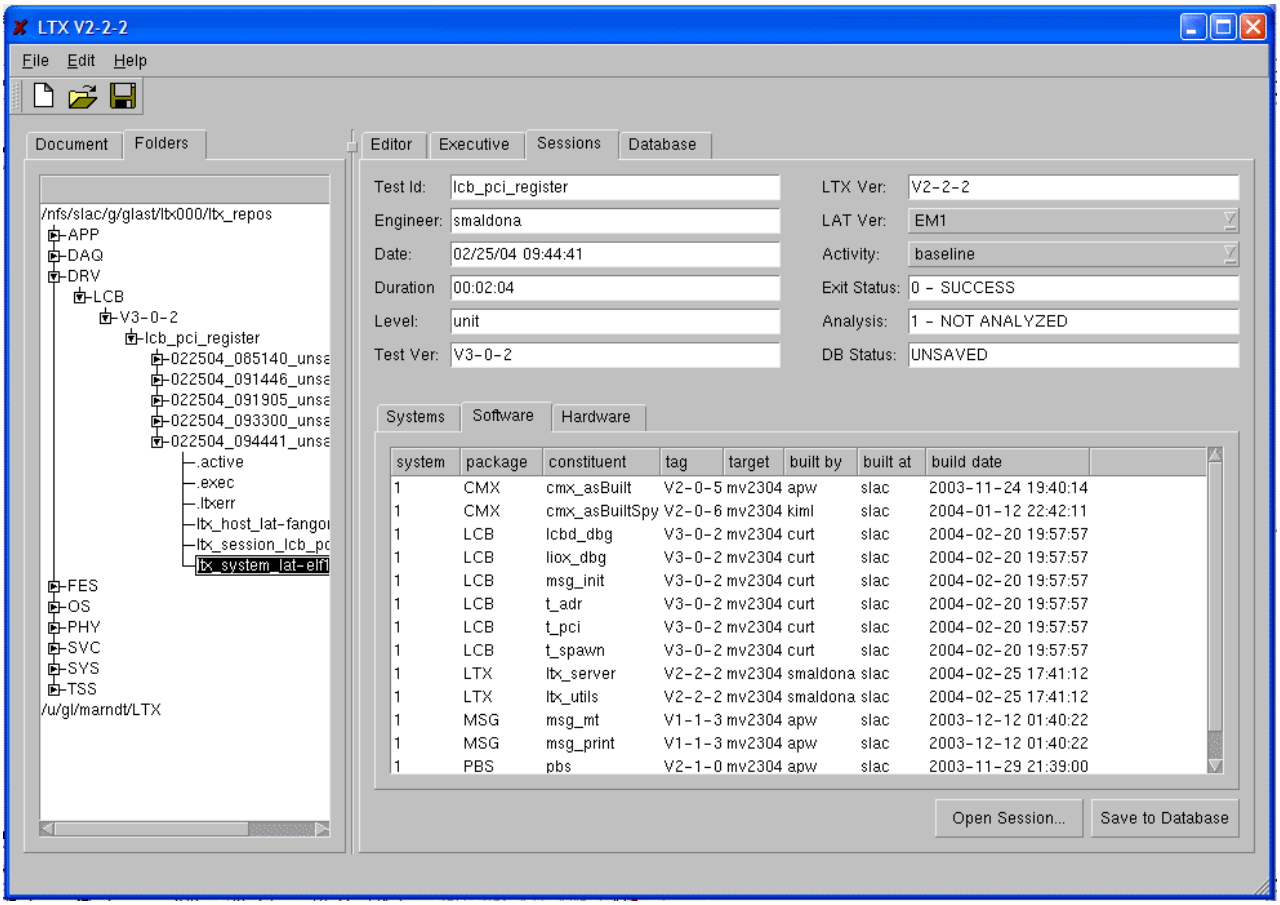


Figure 2: Expected LTX Test Record for LCB Package Test Displayed in LTX GUI

1.3.2 Output of the TKR and CAL Demos

1.3.2.1 Output of TKR and CAL Register Write/Read Demo

When the test of register reads and writes is complete, the LATTE test suite displays a dialog indicating either that (1) all registers were written and read successfully or (2) some read or write operations were not successful. LATTE also returns a report as shown in Figure 3 below.

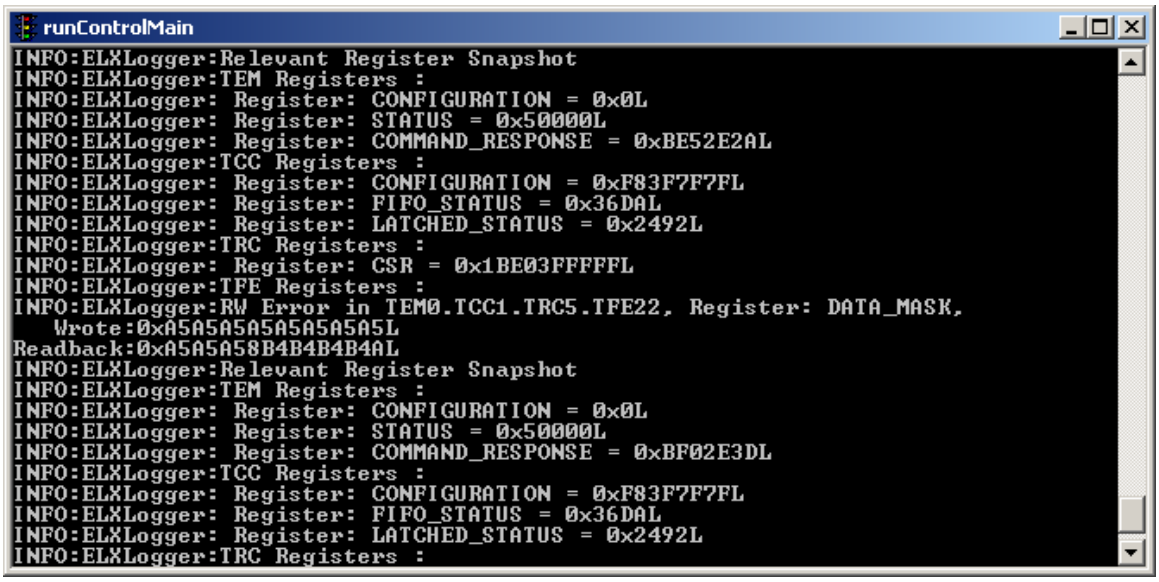


Figure 3: Expected LATTE Test Report for TKR and CAL Register Write/Read Demo

1.3.2.2 Output of TKR Event Data Demo

Using the LATTE test suite, the TEM is configured to accept data only from a subset of TCCs at a time and then triggered. The subsets are chosen from the list of TCCs present, as the TEM can only enable or disable an entire TCC at a time.

The event data is then parsed to ensure that all the correct strips of data and only all the correct strips of data are present. This event test verifies that the TEM control over tracker contribution is fully functional.

When an error occurs, the LATTE test suite displays a warning dialog and relevant status registers are logged for reference and error analysis. See Figure 4 and Figure 5 below. If there are no errors, a success dialog is displayed and status registers are logged for reference.

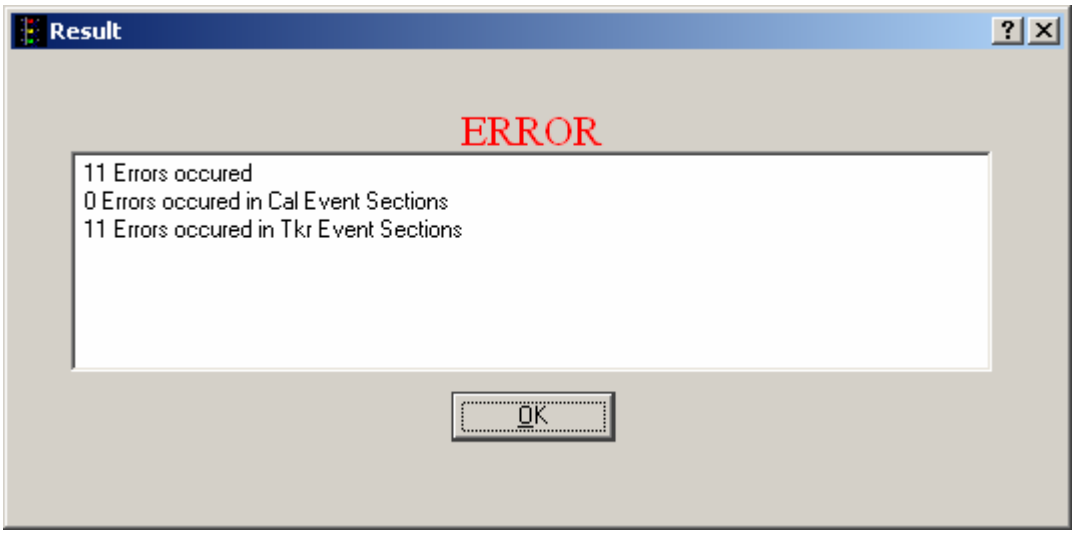


Figure 4: LATTE Error Dialog Used to Signal Errors During the Event Data Test

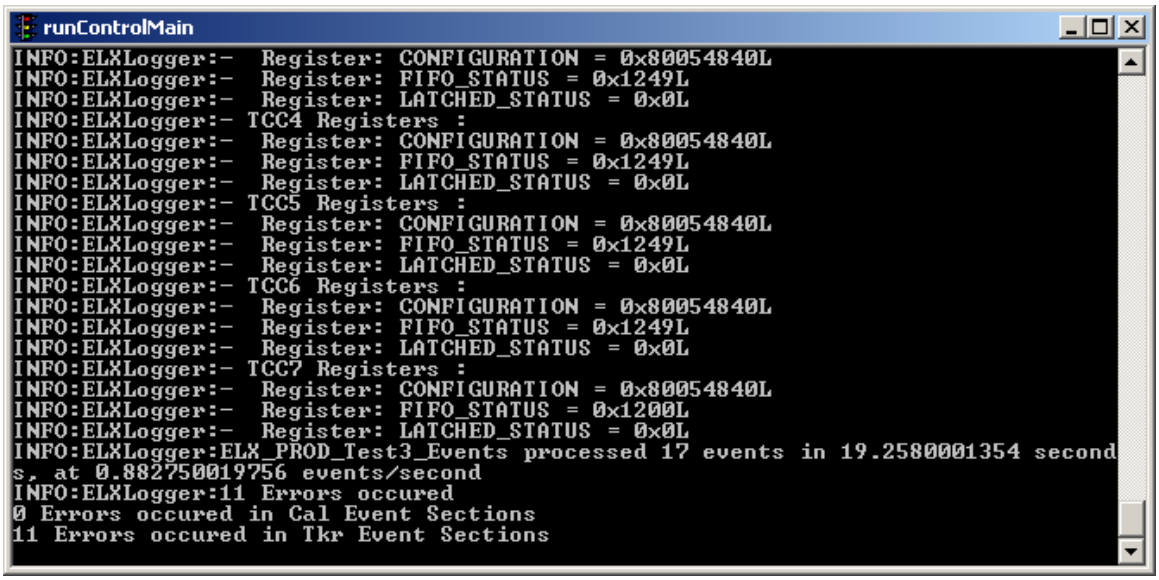


Figure 5: LATTE Log Records Written During Event Data Test

1.3.2.3 Output of CAL Event Data Demo

As with the TKR event test, following instrument configuration using the LATTE test suite, a trigger is sent. When an event returns, it is checked for event errors. The CAL contribution to the event is then checked for the presence of and only the presence of the selectively enabled log-ends. An error is flagged and reported if any data is either missing or unexpectedly found.

The test reports the number of events taken and the number of errors found.

When an error occurs, the LATTE test suite displays a warning dialog and relevant status registers are logged for reference and error analysis. Test data for both the TKR and CAL tests are generated together, thus the dialogs and reports are identical to those shown in Figure 4 and Figure 5 above.

1.3.3 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.

1.3.3.1 Expected Watchdog Test Output While Test Tasks are Not Hung

```
.
.
.
TestMessageInterface: LSWQ=0x76bc90 q=0x76bca8

Test tasks (1p through 4p) report normal progress
<1p....w1>
<3p....w3>
<3p....w3>
<1p....w1>
<3p....w3>
<2p...**********.w2>
<0p....w0>
<3p....w3>
<1p....w1>
<3p....w3>
<3p....w3>
<1p....w1>
<3p....w3>
<2p...**********.w2>
<0p....w0>
<3p....w3>
<1p....w1>
<3p....w3>
<3p....w3>
<1p....w1>
<3p....w3>
<2p...**********.w2>
<0p....w0>
<3p....w3>
<1p....w1>
<3p....w3>
<3p....w3>
.
.
.
```

```
    .
    .
    .<1p....w1>
    <3p....w3>
    <2p...**********.w2>
    <0p....w0>

    LSW_C: n=4

    After a preset interval, the test task callbacks report timer
    values

    [0-0:tc=8830108803 ts=0 d=8830108803]
    [1-1:tc=8830149663 ts=0 d=8830149663]
    [2-2:tc=8830185843 ts=0 d=8830185843]
    [3-3:tc=8830221663 ts=0 d=8830221663]

    The Watchdog control task resets the hardware watchdog to
    120000 msecs
    LSW_T: noProg=0  procMode=1
    LSW_RESET_WDT: tMsecs=120000
    .
    .
    .
```

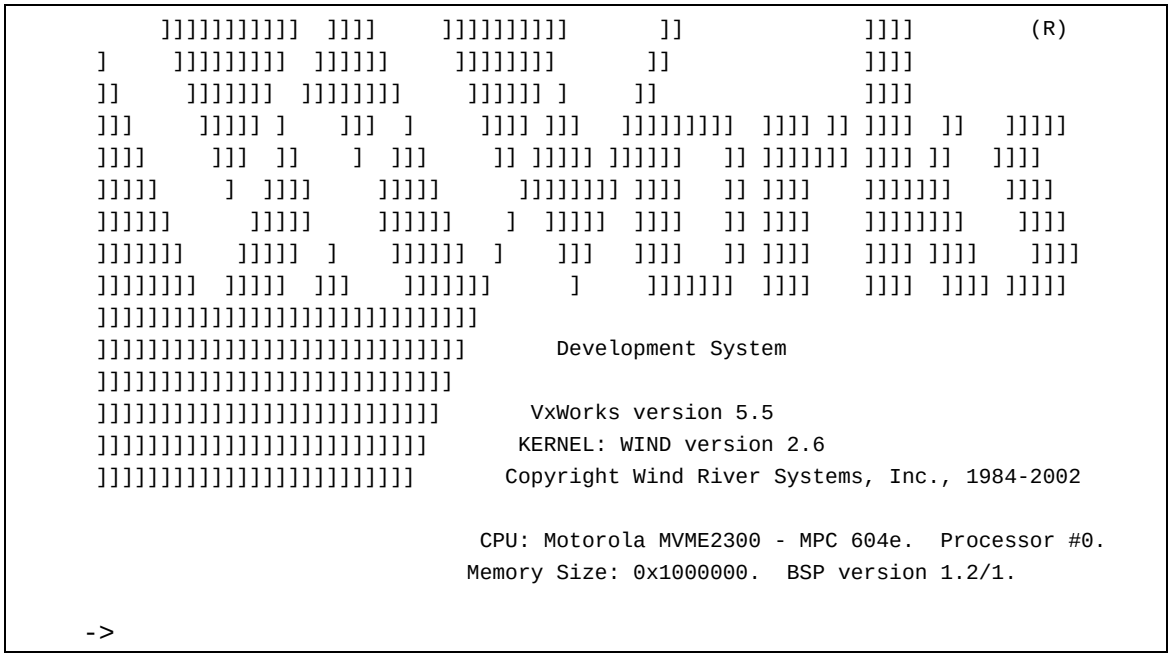
1.3.3.2 Expected Watchdog Test Output When Test Task 2 is Deliberately Hung

```
    .
    .
    .
    The test script sends a signal to hang Task 2

    HangTask: tn=2  ln=3

    Task 2 becomes stuck in a loop; it outputs asterisks
    "****************"to terminal

    <3p....w3>
    <2p...**********************************************************
************************************
    **********************************************************************
************************************
    **********************************************************************
************************************
    **********************************************************************
************************************
    ************************
    <0p.************.**.******************.
    LSW_C: n=w0>******
    <3p4.**************.
    <1p*.
```

```
    *..*********..w3>*[0-
*.w1>******0******************:******************tc=****************
32891940602***
    ****** ts=****************0******************
d=*****************32891940602*********]
    ******************[1-
*****************1******************:******************tc=***********
******33100047
    418********* ts=****************0******************
d=*****************33100047418*********]
    ******************[2-
*****************2******************:******************tc=***********
******
    <3p33308374434.****. ts=*.*************.32744585285w3>*****
d=*****************563789149***********]
    ******************[3-
*****************3******************:******************tc=***********
******33521913
    230********* ts=****************0******************
d=*****************33521913230*********]
    ******************
    LSW_T: noProg=*****0******************
procMode=*********1*******************
    LSW_RESET_WDT: tMs*ecs=****************120000**************
    *******************
    <1p
    LSW_T: rbtm=0x.*0*.****************.
    <3p*.  chkNs=*..*w1>*.8000000000*.w3>*
    LSW_T:
chktmo=0x***9c6676608***************************************************
***************************
    ***************************************************************
************************************
    ***************************************************************
************************************
    ***************************************************************
************************************
    ************************************
    <3p.***************.******************.******************.w3>****
************************************
    ***************************************************************
************************************
    ***************************************************************
************************************
    ****************************************
    <0p.************.**.******************.w0>**********************
*******************************
    <1p.***************.******************.******************.w1>****
************
    .
    .
    .
```

1.3.3.3 Expected Watchdog Test Output When the Watchdog Fails to Reset the Hardware Watchdog and the System Is Rebooted

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

    *****************[1-
****************1******************:******************tc=***********
******42246738
    175********* ts=****************
    <3p0.**************. d=*.**************.
    <1pw3>*.422467381*.75*.***************.]
    w1>**************[2-
****************2******************:******************tc=

    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…

    Press any key to stop auto-boot...
     0
    auto-booting...


    boot device          : dc
    unit number          : 0
    processor number     : 0
    host name            : lat-fangorn
    file name            : /afs/slac/g/glast/flight/oslink/lat-
elf8/vxw_bsp
    inet on ethernet (e) : 134.79.95.39:fffffc00
    host inet (h)        : 134.79.95.36
    gateway inet (g)     : 134.79.95.1
    user (u)             : lat_rsh
    flags (f)            : 0xa
    target name (tn)     : lat-elf8

    Attached TCP/IP interface to dc0.
    Attaching network interface lo0... done.
    Loading... 1059664
    Starting at 0x100000...

    Attached TCP/IP interface to dc unit 0
    Attaching network interface lo0... done.
    Unable to add route to 134.79.92.0; errno = 0xffffffff.

    Adding 1145 symbols for standalone.


     ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
     ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
     ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
```



2 Demonstration Procedure

2.1 LCB Package Test Demo

The LCB Package Test Demo has three goals:

- To show that tests of the LCB software package have been designed and scripted; these tests can be run in the LAT Flight Software Test Executive (LTX) system.
- To show that the LCB package can successfully control PCI register reads and writes on a functioning LAT Communications Board.
- To show the capabilities of the LAT Flight Software Test Executive.

2.1.1 LACB Package Test Demonstration Procedures

- The VME crate and TEM are powered up. When the VME crate is powered up, several software packages are automatically loaded (see below for package list).
- The test engineer logs on to a Linux host, from which the LTX testing system is run.

- The tester issues the **cmx start** command to launch the code management tool and thus gain access to the test scripts and FSW software libraries.
- The tester issues the **ltxui** command to launch the LTX GUI application.
- The tester uses the LTX GUI to load the **lcb_pci_register.py** test script.
- Using the LTX GUI, the tester configures LTX to use serial line communications to the MV2304.
- The tester starts the test.

2.1.2 Context of the LCB Package Test

- The LCB package, which controls the functioning of the LAT Communications Board, is installed on a VME crate running an MV2304 processor. This crate contains a functioning LAT Communications Board.
- In addition to the LCB package, the DEM, DAB, PBS, MSG, GNAT, and GGLT packages are installed on the MV2304.
- The VME crate is connected through a LAT Communications Board to a production laboratory version of a TEM.
- The MV2304 CPU is running a developer edition of VxWorks that provides terminal emulation and other features.

2.2 TKR and CAL Demos

The TKR and CAL demo will be run at a demonstration terminal in Building 80.

The TKR and CAL Demonstrations have three goals:

- To show the ability of the DEM package to communicate with the TKR and CAL over the command/response communications network fabric.
- To show that DEM correctly constructs bit strings for write operations to the TKR and CAL, and correctly decodes bit strings from read operations from the TKR and CAL.
- To show that, using the DEM package to write registers, engineers can configure TEM (TKR and CAL) contributions to the event data set.

The purpose, interfaces, implementation approach, and modes of operation for these tests are described in full in document LAT-TD-02485.

2.2.1 TKR and CAL Register Read/Write Test

The Read/Write Test exhaustively exercises each readable/writeable register on each TKR or CAL function block selected, both on-board and off-board the TEM. The test allows the selection of several patterns and applies these patterns to test each register. Further, the test follows this routine over several iterations to determine failures. The test's strength is its comprehensive design. The test is fairly long and categorically determines the functioning of all subcomponents. This test is used to determine the presence of any register address line anomalies and command-response failures, if any. The register test also provides detailed status information for error analysis in the event of a failure. All status and configuration registers on the object and all its

parents are displayed as and when an error occurs. This test is also written to work with multiple TEMs, though for today's demonstration, a single TEM is used.

2.2.1.1 TKR and CAL Register Read/Write Test Procedures

- The VME crate and TEM are powered up. When the VME crate is powered up, several software packages are automatically loaded (see below for package list).
- A Python test script (**ELX_PROD_Test1_RegRW.py**) is loaded onto a Wintel PC running the LATTE test suite.
- Next, the LATTE test suite presents a configuration dialog for the read/write test; if the test engineer wishes to override elements from the test script, he or she specifies which registers on the TKR and CAL are to be tested, the kinds of bit strings send these registers, the number of test iterations, and a readback mode.

2.2.1.2 Context of the TKR and CAL Register Read/Write Demonstrations

- The DEM package, which handles communications with the TKR and CAL and processes read and write bit strings, executes on a VME crate running an MV2304 processor.
- Aside from the DEM package, the DAB, PBS, MSG, LCB, GNAT, and GGLT packages are installed on the MV2304.
- The VME crate is connected through a LAT Communications Board to a production laboratory version of a TEM.
- The LATTE test suite runs on a Wintel PC connected to the MV2304 via Ethernet.
- The MV2304 CPU is running a developer edition of VxWorks that provides terminal emulation and other features.

2.2.2 TKR and CAL Event Tests

There are two main event tests, the first corresponding to the Calorimeter and the second to the Tracker. These two tests are combined into a single test for a unified event test that takes events with contributions from each subsystem simultaneously. The Event Tests use the Calibration Strobe and Trigger Acknowledge (TACK) to test the hardware with solicited events.

The **CAL portion** of the test is used to examine the TEM 's control over event settings for the Calorimeter contribution. It is used to ensure that event data is collected correctly when subsets of the available set of calorimeter components are selectively turned off at the TEM. This test is extensive in that it allows for testing several different permutations of the log-ends present.

The **TKR portion** of the test is used to examine the TEM 's control over event settings for the Tracker contribution. It is used to ensure that event data is collected correctly when subsets of the available set of tracker components are selectively turned off at the TEM. As the TEM can only turn on and off an entire cable controller at a time, this is the granularity at which subsets of tracker components are selected in this test.

2.2.2.1 TKR and CAL Event Test Procedures

- The VME crate and TEM are powered up. When the VME crate is powered up, several software packages are automatically loaded (see below for package list).

- A Python test script (**ELX_PROD_Test3_Events.py**) is loaded onto a Wintel PC running the LATTE test suite.
- The LATTE test suite presents a configuration dialog for the event test; if the test engineer wishes to override elements from the test script, he or she specifies Calorimeter and Tracker configuration options and other test options as shown in Figure 6 below.

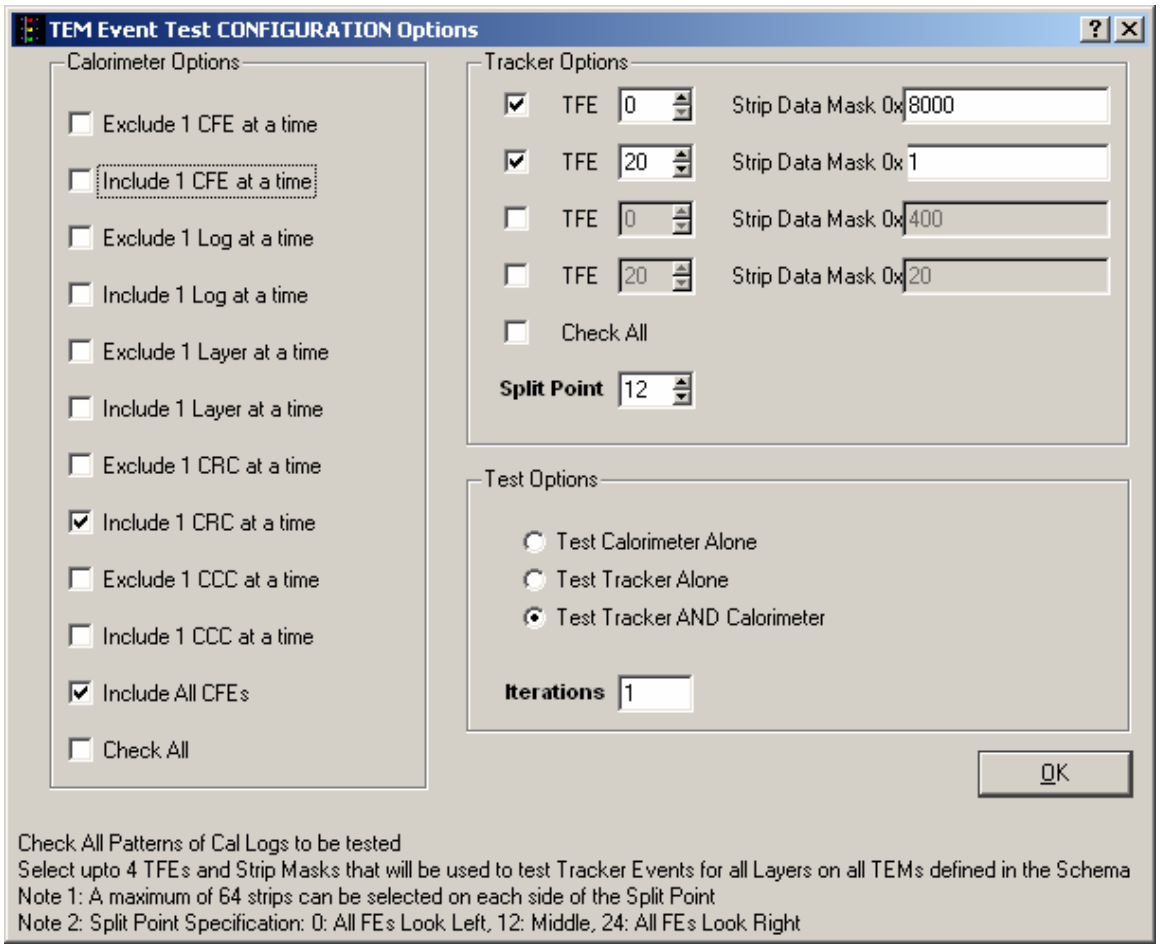


Figure 6: LATTE Configuration Dialog for CAL and TKR Event Test

2.2.2.2 Context of the TKR and CAL Event Demonstrations

- The DEM, DAB, PBS, MSG, LCB, GNAT, and GGLT packages are installed on the MV2304.
- The LATTE test suite runs on a Wintel PC connected to the MV2304 via Ethernet.
- The VME crate is connected through a LAT Communications Board to a production laboratory version of a TEM.
- The MV2304 CPU is running a developer edition of VxWorks that provides terminal emulation and other features.

2.3 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 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.

2.3.1 Watchdog Demo Procedure

- The RAD750 crate is powered up and VxWorks is booted.
- 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 below for the exact constituents).
- The **lswTest** test suite is launched at the VxWorks prompt and the Watchdog demo begins.

2.3.2 Context of the Watchdog Demo

- The Watchdog function demo runs on a 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.
- Constituents of the CMX package (cmx_asBuiltSpy), VXW package (vxw_extra), LSW package (lsw, lswTest), and simple print routines are installed on the RAD750.

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 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.