LAT Flight Software

LAT Charge Injection Calibration Software User Guide

Number:	V1-0-0
Subsystem:	Data Acquisition/Flight Software
Supersedes:	None
Type:	Document Template
Author:	J. Swain
Created:	February 18, 2005
Updated:	June 29, 2005
Printed:	March 28, 2005

This document describes the usage of the LAT Charge Injection Calibration utility (LCI).

Document Approval

Prepared By:			
	J.Swain	LAT Flight Software	Date
Approved By:			
	G.Haller	LAT Electronics Manager	Date
Approved By:			
	J.J.Russell	LAT Flight Software Manager	Date

Record Of Changes

Version	Date	Title Or Brief Description	Entered By
1	10 June 2005	Draft Release	J. Swain

Contents

0 Commencement

0.0 Scope

This document provides an overview of the LAT Charge Injection Calibration package from the perspective of a new user. Maintainers should also make themselves familiar with the *LAT Charge Injection Calibration Software Design* document.

0.1 References

1. *LAT Charge Injection Calibration Software Design*.
2. LAT-TD-00606 *LAT Inter-module Communications: A reference manual*.
3. LAT-TD-01380 *LAT Communication Board Driver: Software architecture and interfaces*.
4. LAT-TD-01547 *The Command/Response Unit: Programming ICD specification*.
5. LAT-TD-01546 *The Event Builder Module: Design specification*.
6. LAT-TD-01545 *The GLT Electronics Module: Programming ICD specification*.
7. LAT-TD-00639 *The ACD Electronics Module (AEM): A primer*.
8. LAT-SS-00363 *ACD-LAT ICD: Mechanical, thermal and electrical*.
9. LAT-SS-00363 *LAT Dataflow Specification: ACD-AEM interface*.
10. LAT-TF-00605 *The Tower Electronics Module (TEM): Programming ICD specification*.
11. *The GLAST acronym list*.

0.2 Request for Comments

If you have comments or corrections, please email jswain@slac.stanford.edu.

0.3 Acronyms

LCI - LAT Charge Injection

LAT – Large Area Telescope

SIU – Spacecraft Interface Unit

PDU – Power Distribution Unit

EPU – Event Processing Unit

TEM – Tower Electronics Module

CRU – Command Response Unit

ACD – Anti-Coincidence Detector

DAQ – Data AcQuisition

DAB – DAQ Board

LATp – LAT (communications) protocol

EEPROM – Electronic Erasable Programmable Read Only Memory

LCB – LAT Communications Board

GEM – Global trigger Electronics Module

EBM – Event Builder Module

AEM – ACD Electronics Module

FREE – FRont End Electronics

P/R – Primary/Redundant

C/R – Command/Response

1 Conspectus

The LAT electronics include circuitry to inject a known amount of charge into the front-ends for the purposes of diagnostics and calibration. The LAT Charge Injection Calibration software uses this circuitry to produce a dataset that contains information required to calibrate the front-end electronics. FSW does *not* provide tools to analyze such data.

The complete calibration consists of an initial LAT configuration, performed using LATC, followed by repeated cycles of configuration, collection, construction, compaction and consignment. The LCI configuration affects only a small subset of the LAT registers in the EBM, GEM and threshold registers scattered around the DAQ system.

2 Control

The number of LCI functions exposed for general use is extremely limited. Resource allocation, deallocation and task control are provided by the commands listed below.

```
unsigned LCI_initialise(unsigned cap)
```

The single argument to this function, *cap*, specifies the maximum number of events that a single calibration cycle can collect, compact and consign. The actual number requested is specified in the LCI configuration file. If the request is greater than the maximum then the control functions will preform repeated cycles until the requested number of events has been collected.

This function can fail, and will return a suitable message code when it does. Typically the failure will be a memory allocation error.

After this function has been called LCI will be in the INITIALISED state and can be either torn-down or started.

```
unsigned LCI_startTask(void);
```

The LCI task cannot be commanded to perform a calibration until it has been started. This function forks the task and waits until it is started before returning.

This function will return the LCI_STATE error if it is invoked when LCI is in any state other than INITIALISED. Other errors may be returned by the underlying ITC_startTaskW function.

After this function has been called LCI will be in the WAITING state and can either be commanded to calibrate or stop.

```
unsigned LCI_stopTask(void);
```

This function stops the LCI task. No resource deallocation is performed.

This function will return the LCI_STATE error if it is invoked when LCI is in any state other than WAITING. Other errors may be returned by the underlying ITC_stopTask function.

After this function has been called LCI will be in the INITIALISED state and can be either torn-down or started.

```
unsigned LCI_teardown(void);
```

Resource deallocation is performed by the LCI_teardown functions.

This function will return the LCI_STATE error if it is inoked when LCI is in any state other than INITIALISED. No other errors will be returned by this function.

3 Commanding

There are only two commands that LCI accepts, *calibrate* and *abort*.

3.0 Calibrate

The calibrate command triggers LCI to read in a configuration file and perform as many calibration cycles as necessary to collect the data requested.

The command will be rejected if LCI is not in the WAITING state.

The success or failure of the calibration will be reported by the transmission of an ITC packet to the raw socket of the initiating task, from the LCI task.

During calibration the LCI task will be in state CALIBRATING, and will revert to state WAITING once the calibration is complete and the status message has been sent.

3.0.0 Arguments

(a) LATC configuration master file ID used to configure the LAT.

(b) LCI configuration ID.

(c) Destination, either File ID or LATp address, of the calibration data.

3.1 Abort

The amount of time that the LAT is occupied with calibration is variable, depending on the number of cycles requested in the LCI configuration file and the number of events per cycle. In any case, it will be significant, which means than changing circumstances could make it be desirable to terminate the calibration early.

If LCI is not in the state CALIBRATING when the *abort* command is recieved then the command is a no-op. Otherwise the state is changed to ABORTING. At the start of the next cycle, the change of state will be detected and the calibration will terminate.

3.1.0 Arguments

None.

4 Configuration

The "real" user interface is provided by the LCI configuration XML files. These file are compiled into binary configuration files for upload to the LAT using the LCI_parser executable. The ID of this binary file is one of the parameters to the *calibrate* command. The parser is stateful so once a basic configuration has been established subsequent statements modify this configuration. The parser will take two arguments, a string identifying the subsystem configuration being specified ("ACD", "CAL" or "TKR") and the XML file name.

Many of the quantities specified in the configuration can be iterated over as indicated by the *iterate* comment next to the tag in the description below. In this case the XML element should contain either the elements <**initial**>, <**delta**> and <**count**>, or the single element <**constant**>, or a keyword. The keyword LATC is always acceptable, and indicates that LCI should not load any value for this parameter. Note that the LCI XML parser does not perform bounds checking on the parameters entered.

4.0 XML Vocabulary

Hierarchy of XML tags used to specify the LCI configuration.

4.0.0 Common

The following tags are used in all configuration files.

<**configuration**> : Tag encompassing changes to the configuration.

This tag can be empty, which will cause the parser to output another copy of the previous configuration.

<**number**> : Number of events to collect.

<**period**> : Clock ticks between calibration pulses.

DEFAULT gives values of 20000 for a pulse rate of 1kHz.

This is the one value that is subject to bounds checking. If a value less that 20000 is entered then 20000 will be used.

<**delay**> : Delay inserted after the configuration phase of the calibration cycle.

Large changes in DAC value can require a settling period.

<**strobe**> : It can be interesting to set up the LAT for calibration, but not pulse the front-end electronics, i.e. send a TACK only. The OFF keyword specifies that the TAM **should not** have the calstrobe bit set. The ON keyword specified that the TEM **should** have the calstrobe bit sent.

<**zero_suppress**> :

ON enables zero-suppression.

OFF disables zero-suppression.

4.0.1 Calorimeter

The follow tags are use in calorimeter configuration files.

<**inject**> : *Iterate* : Size of calibration charge. Twelve bit DAC value for the calorimeter.

<**trigger**> : Trigger threshold

<**low**> : *Iterate* : Low energy trigger threshold. Seven bit DAC

The range select bit is just treated as the MSB.

<**high**> : *Iterate* : High energy trigger threshold. Seven bit DAC

<**log_accept**> : *Iterate* : Log accept threshold. Seven bit DAC

<range_uld> : *Iterate* : Range upper level discriminator threshold. Seven bit DAC

<reference> : *Iterate* : Seven bit reference DAC

<**range_mask**> : Four bit mask indicating the calorimeter ranges to compact and consign.

AUTO will cause the calorimeter auto-ranging to be used and a single range returned.

<**first_range**> : Sets the state of the USE_FRST_RNG bit and FRST_RNG bits.

LATC indicates that LCI should not set these bits.

OFF Indicates that the USE_FRST_RNG bit should be cleared.

A number from 0 to 3 indicates the the USE_FRST_RNG bit should be set, and the FRST_RNG bits should be set to the value.

<**tack**> : *Iterate* : Eight bit trigger sequence TACK delay.

<**column**> : *Iterate* : Calorimeter column selection [0, 11].

FOREACH indicates that LCI should iterate over all the columns in the tower.

ALL indicates that LCI should enable all columns of the tower.

4.0.2 Tracker

The following tags are used in tracker configuration files.

<**threshold**> : *Iterate* : Tracker trigger threshold. Seven bit DAC value

<**inject**> : *Iterate* : Size of calibration charge. Seven bit DAC value for the tracker.

<**tack**> : *Iterate* : Eight bit trigger sequence TACK delay.

<**channel**> : *Iterate* : Tracker channel selection.

FOREACH indicates that LCI should iterate over all the channels in the tower with one channel enabled per layer.

ONEPERTFE indicates that LCI should iterate over all the channels on a TFE with one channel enabled per TFE.

4.0.3 ACD

The following tags are used in ACD configuration files.

<**inject**> : *Iterate* : Size of calibration charge. Six bit DAC value.

<**pha**> : *Iterate* : PHA threshold. Sixteen bit DAC value

<**veto**> : *Iterate* : Veto threshold. Six bit DAC value

<**veto_vernier**> : *Iterate* : Six bit DAC value

<**hld**> : *Iterate* : Six bit DAC value

5 Conclusion

After data collection, compaction and consignment are complete the calibration data will be down linked to the ground and will be processed off line. Before the analysis can be performed the calibration data must be expanded and put into a form acceptable to the off line process. A third LCI executable will be provided to perform this transformation.

The final format of the output data is still to be determined.