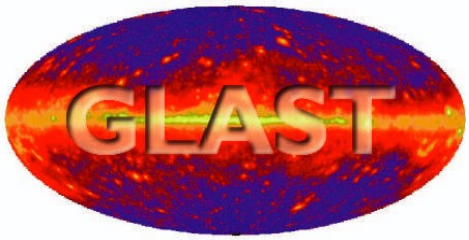

LAT Flight Software

LAT Charge Injection Calibration Software User Guide

Number: V1-2-2
Subsystem: Data Acquisition/Flight Software
Supersedes: V1-2-1
Type: User Manual
Author: J. Swain & Owen Saxton
Created: February 18, 2005
Updated: August 6, 2008
Printed: June 18, 2007

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
1.0.0	10 June 2005	Draft Release	J. Swain
1.1.0	26 Sept 2006	Brought up-to-date; added more detail	Owen Saxton
1.2.0	18 June 2007	Added new configuration elements	Owen Saxton
1.2.1	29 June 2007	Fixed typos in new CAL configuration element names	Owen Saxton
1.2.2	6 August 2008	Changed auto_range to four_range	Owen Saxton

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 saxton@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(void)
```

This function uses the LCI configuration database to supply needed parameters. One of these of interest is the capacity of a calibration, 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 three commands that LCI accepts, *calibrate, abort* and *noop*.

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 that 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 received 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.

3.2 Noop

This command is included as a way of testing whether the calibration task is responsive. Regardless of the calibration state, it is executed as soon as possible, but will be delayed until the end of the current cycle if the state is CALIBRATING.

3.2.0 Arguments

None.

4 Configuration

The "real" user interface is provided by the LCI configuration XML files. These files 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. A configuration file may contain multiple configurations, which are executed in succession. The first such configuration establishes a base and should specify all the available calibration elements. The common elements must be specified, as no defaults are taken. The sub-system specific elements may be omitted, and default to the values established by the LAT configuration. Subsequent configurations need only specify incremental changes from the previous configuration, and can be null.

The parser takes two or three arguments: an optional string identifying the subsystem configuration being specified ("ACD", "CAL" or "TKR"), the XML file name, and the output binary file name.

The contents of a binary configuration file may be displayed using the `LCI_report` executable. It takes one or two arguments: the name of the binary file and the optional name of the output file. If the latter is not specified, the output is directed to the terminal.

4.0 XML File Structure

It is recommended, but not required, that the first two lines of a configuration XML file be an XML declaration followed by a document type declaration. These lines have the form:

```
<?xml version='1.0' standalone='yes' ?>

<!DOCTYPE LCI_XXX_XML SYSTEM
"/afs/slac/g/glast/flight/ICS/source/LCI/prod/LCI/xxx.dtd">
```

in which “XXX” and “xxx” are to be replaced by the name of the subsystem being calibrated.

The last part of the document type declaration is the name of the document type definition (DTD) file defining the structure of the XML file. In this case it is not actually used by the parser, but is included for documentation purposes.

The document type name, “LCI_XXX_XML”, normally specifies the subsystem being calibrated, but if the subsystem name is supplied to the `LCI_parser` command, it has precedence, and the name in the document type declaration, if present, must match.

The remainder of an XML file is enclosed within the root tag pair, whose name must be the same as the document type, i.e. “LCI_ACD_XML”, “LCI_CAL_XML”, or “LCI_TKR_XML”.

Comments may be placed anywhere within an XML file. These are delimited by the strings “<!--“ and “ -->” (note the presence of the space in each).

4.1 XML Vocabulary

Within the root element, there are one or more configurations, each contained in a <**configuration**> element. As previously stated, the first configuration must contain all the common elements, and it is good practice to specify all the elements pertaining to the given calibration type.

Element (tag) names must be supplied exactly as shown.

Element values are either numbers or keywords. Numbers may be specified in either decimal or case-insensitive hexadecimal (leading “0x”) notation. Keywords are also case-insensitive. With one exception, all numbers used must be unsigned. The parser checks that each value is either a recognized keyword or a valid number, and such a number must lie within an acceptable range.

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 the parameter. Other acceptable keywords are described below where applicable. The value of the <**delta**> element is the only one which may be negative. Note that the <**count**> value is the number of steps from the initial value; hence the total number of values is one greater than this.

When more that one quantity is being iterated over, the iteration order is fixed by the LCI code. This order is the same as the order in which the iterated quantities are presented in the following descriptions, with the first presented being the first one iterated over (i.e. In the innermost loop).

4.1.0 Common Elements

The following elements are used in all configuration files.

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

<**period**> : Clock ticks (in 50 nsec units) between calibration pulses.

DEFAULT gives a value of 20000 for a pulse rate of 1 Khz.

If a value less than 20000 is entered then 20000 will be used.

<**delay**> : Delay (in 1/100 secs) inserted after the configuration phase of the calibration cycle.

Large changes in DAC value can require a settling period.

<**latc_delay**> : Additional delay (in 1/100 secs) when a LATC configuration is required.

Performing a LATC configuration generally requires even more settling time afterwards.

<**strobe**> : ON or OFF.

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 TAM **should** have the calstrobe bit set.

<**zero_suppress**> : ON or OFF.

ON enables zero-suppression, OFF disables zero-suppression.

4.1.1 Calorimeter Elements

The follow elements are used in calorimeter configuration files.

<**four_range**> : ON or OFF.

If ON, calorimeter four-range readout is used. If OFF, only a single data range is read.

<**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.

<**gain**> : Gain selection

<**low**> : Low energy gain selection. Three bit field.

<**high**> : High energy gain selection. Four bit field.

<**range_enab**> : Range enable bits.

<**low**> : Low energy range enable. ON or OFF.

<**high**> : High energy range enable. ON or OFF.

<**trig_enab**> : Trigger enable bits.

<**low**> : Low energy trigger enable. ON or OFF.

<**high**> : High energy trigger enable. ON or OFF.

<**calib_enab**> : Calibration enable bits.

<**low**> : Low energy calibration enable. ON or OFF.

<**high**> : High energy calibration enable. ON or OFF.

<**calib_gain**> : Calibration gain to use. HIGH or LOW.

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

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

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

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

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

<**trigger**> : Trigger threshold

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

The range select bit is just treated as the MSB.

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

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

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

4.1.2 Tracker Elements

The following elements are used in tracker configuration files.

<**split**> : The splits values to use.

<**low**> : Low side split value, i.e. the number of low-talking GTFEs, 0 - 24.

<**high**> : High side split value, i.e. the number of high-talking GTFEs, 0 - 24.

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

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

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

<**channel**> : *Iterate* : Tracker channel selection, 0 - 1535.

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

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

4.1.3 ACD Elements

The following elements are used in ACD configuration files.

<**range**> : Range setting. HIGH or LOW.

This has an effect only if the LAT AFE configuration sets the manual_gain_range field of config_reg to 1.

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

<**hld**> : *Iterate* : High level discriminator. Six bit DAC value.

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

<**veto_vernier**> : *Iterate* : Veto vernier. Six bit DAC value.

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

<**bias**> : *Iterate* : Bias voltage. Six bit DAC value.

<**hold_delay**> : *Iterate* : Hold delay. Seven bit DAC value.

<**hitmap_delay**> : *Iterate* : Hitmap delay. Five bit DAC value.

<**channel**> : *Iterate* : ACD channel selection, 0 - 215.

FOREACH indicates that LCI should iterate over all the channels, enabling one at a time.

ALL indicates that all channels are enabled.

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.