Timing-in Parameters and Procedures

Martin Kocian

April 21, 2008

Contents

1 Preface

1.1 Purpose

This documents the timing-in of the LAT.

1.2 Scope

Documentation

1.3 Applicable Documents

LAT-MD-01545 The GLT Electronics Module
Timing in during L&EO:
https://confluence.slac.stanford.edu/download/attachments/35026/timinginleo.pdf?version=2
Eduardo's Talk at the IA6 Workshop:
http://www-glast.slac.stanford.edu/IntegrationTest/SVAC/Instrument_Analysis/Workshop-6/Talks/EduardoIA6Feb2006.pdf

1.4 Definitions and Acronyms

Acronyms:

ACD Anti-Coincidence Detector Subsystem of the LAT

CAL Calorimeter Subsystem of the LAT

CAL-LO Low energy Calorimeter trigger, nominally 100 MeV

CAL-HI High energy Calorimeter trigger, nominally 1000 MeV

CNO Carbon, Nitrogen, and Oxygen cosmic rays

DGN Diagnostic filter

FHE CAL high threshold discriminator

FLE CAL low threshold discriminator

FSW Flight Software

GEM GLT Electronics Module

GAMMA Gamma ray filter

GLAST Gamma-Ray Large Area Space Telescope

GCCC Calorimeter Cable Controller

GTCC Tracker Cable Controller

HIP Heavy Ion Filter

LAT Large Area Telescope

L&EO Launch and Early Operations

MIP Minimum-Ionizing particle and also Minim-ionizing particle filter

MC Monte Carlo simulation

ROI Region of Interest, a set of ACD tiles grouped for trigger or veto

TACK Trigger acknowledge

TKR Tracker Subsystem of the LAT

TREQ Trigger request

2 Timing Parameters

2.1 General Introduction

The timing parameters control the delays for triggering and for latching the data. The timing delays are set through a number of registers. In addition to the delays set through the registers there are inherent delays in the system. For a complete schematic view of the timing see slide 6 of Eduardo's talk (link given in the "Applicable documents" section).
A complete description of the trigger and dataflow system can be found in TD-01545.

The timing registers can be divided into 4 groups:

- **TREQ:** Alignment of the trigger requests. The delays are applied in the frontends before the trigger request is sent to the GEM.
- **Trigger Window:** The window width is set in the GEM.
- **TACK:** Delays that control the latching of the event data. The delays are set in the TEMs/AEM.
- **Frontend Latching Delays:** There are additional registers in the frontends that control the latching and digitizing of the waveforms as well as the latching of AEM hitmap data.

2.2 TREQ Parameters

For normal data taking the TREQ delays should be set in a way that the trigger signals line up in time at the input of the GEM.

2.2.1 Calorimeter

The TREQ delay is set in the lower 4 bits of the GCCC register TRG_ALIGNMENT. This means that each cable controller could have its own delay. The alignment procedure does timing in by tower only so all GCCCs of one tower have the same delay. In the initial flight configuration all towers have the same delay as well. The Calorimeter is the slowest subsystem. In the initial configuration all delays are therefore set to 0. CAL high and CAL low share their delay registers so it is not possible to time in CAL high and CAL low independently.

2.2.2 Tracker

The TREQ delay is set in the lower 4 bits of the TCCC register TRG_ALIGNMENT. This means that each cable controller could have its own delay. The alignment procedure does timing in by tower only so all GTCCs of one tower have the same delay. The TKR is the second slowest subsystem. In the initial configuration it is delayed by 5 clock ticks to line up with the calorimeter.

2.2.3 ACD

The TREQ delay is set through the GARC veto_delay register. This means that each of the 12 GARCs controls its own delay. The TREQ alignment procedure measures the delay for each tile but in the end all tiles are set to the same delay. The ACD is the fastest subsystem. In the initial configuration it is delayed by 16 clock ticks with respect to the calorimeter. CNO shares the veto_delay registers with the veto tiles so it cannot be timed in separately. A compromise has to be adopted between the timing of the two subsystems.

2.3 Trigger Window

The trigger window is opened when a subsystem that is included in the window open mask sends a trigger request. All subsequent request from other subsystems are included as bits in the trigger word if they occur during the trigger window. Initially the trigger window is set to 12 ticks. The width is determined from the trigger jitter. The fast shaping CAL pulse has a rise time of about 500 ns (10 ticks) which is the largest contributor to the trigger jitter.

2.4 TACK

The TACK delays are set to latch the data at the peak of the signal waveform for optimal signal to noise ratio. There are separate delay registers for the subsystems.

2.4.1 Calorimeter

The TACK delay is set through the TACK delay field of the CAL_TRGSEQ register in the TEM. Each tower can set its TACK delay separately. In the timing in procedure the delay

is determined by tower. There is no separate delay for the high energy readout chain. A compromise between high energy and low energy readout timing has to be found.

2.4.2 Tracker

As in the CAL the TACK delay is set by tower through the TACK field of the TKR_TRGSEQ register in the TEM. In the timing in procedure the delays are set by tower.

2.4.3 ACD

For the ACD there is a single TACK delay, set through the TACK field of the TRGSEQ register in the AEM.

2.5 Latching Delays

2.5.1 TKR and CAL

There is a number of registers that delay the latching and digitizing of the data. These have been left at their default settings. No timing in is performed on these parameters.

2.5.2 ACD

The AEM hitmap has to have its parameters set so that it latches the hits in the ACD with the correct timing. Ideally the AEM hitmap should capture the hits over the same time interval as the GEM veto tile list for good consistency. The timing for the two hitmaps works in completely different ways which makes it rather tricky to set the delay parameters correctly. In addition the settings for the hitmap timing interfere with the delays for the latching of the ACD waveform data (a.k.a. PHA data) in that the hitmap timing is effected by the TACK delay. This makes it necessary in the end to modify the data latching register hold_delay to add up to the optimal total delay for data latching while setting the proper timing for the AEM hitmap.
Parameters involved:

- **TACK delay (TRGSEQ):** Introduces a delay for both data and hitmap latching.
- **hold_delay:** GARC register to delay the latching of the waveform data in the frontends.
- **hitmap_delay:** GARC register indicating the length of the pipeline in clock ticks that the hitmap is pushed through.
- **hitmap_width:** If a hitmap signal is above threshold shorter than hitmap_width+3 it will be lengthened to this value.
- **hitmap_deadtime:** This register increases the number of clock ticks that the hitmap signal is above threshold by value ticks.

- **window_width:** The width of the trigger window increases the overall delay of the latching (by the width of the window).

The hitmap is shifted through a pipeline. The input to the pipeline is the discriminator output of the ACD veto discriminator. This signal is then shifted through the pipleline for a fixed number of clock ticks determined by hitmap_delay. The output of the pipeline is latched as the hitmap with the event so the hitmap that is latched is the one a fixed number of clock ticks ago. With veto_delay=0, TACK delay=0, window_width=1 hitmap_delay would have to be set to 7 to latch the hitmap at trigger time which was verified experimentally. We set hitmap_width to 15 and hitmap_deadtime to 1. This way we get a minimum width of 19 ticks when the discriminator fires. With the default settings of veto_delay=16, TACK delay=4, window_width=12 hitmap_delay would thus have to be 38 to capture the first tick of the 19 tick wide hitmap and 20 to capture the last tick of the hitmap. The deterministic relationship is only true if ACD opens the window. If another subsystem opens the window than the delay introduced by window_width varies with trigger jitter. We set hitmap_delay to 27. This way we capture the hitmap if the ACD triggered between 7 ticks before the trigger window was opened and 11 ticks after the window was opened. This choice was made to synchronize the AEM hitmap with the GEM veto map. The veto signal has a width of 8 ticks and is latched if the veto signal was present in the trigger window, i.e. if the rising edge lay between 7 ticks before the start of the window (so that the 8th tick still lay in the window) and 11 ticks after the window opened since the window width is 12. hold_delay is set to 4 which captures the peak of the ACD pulse.
Remember to adjust the hitmap parameters whenever changing any of the parameters above.

3 Timing-in procedure

Timing-in is done through a number of special runs. There are 3 types of TREQ configurations. For the TACK part there are 2 sets of configurations, one set of 6 configs and another set of 3 configs.

3.1 Software

The analysis of the timing in runs is done offline using digi and recon files. The analysis code is set up as Root Tree Analysis. To run the code check out the calGenTRG package in the offline environment. “cd” into the cmt subdirectory and run make. “cd” into the workdir subdirectory. When you start up root it will automatically load the shared libraries that contain the analysis code that you built. To run the analysis code there are scripts that can be run interactively in root. They are typically called runXXX.C. Edit the scripts to set the digi and recon filenames of the run that you want to analyze, the number of events to run over (set to a very large number if you want all events), the names of the output root files, html files, text files (for TACK). To run the script type `.x runXXX.C` in root.

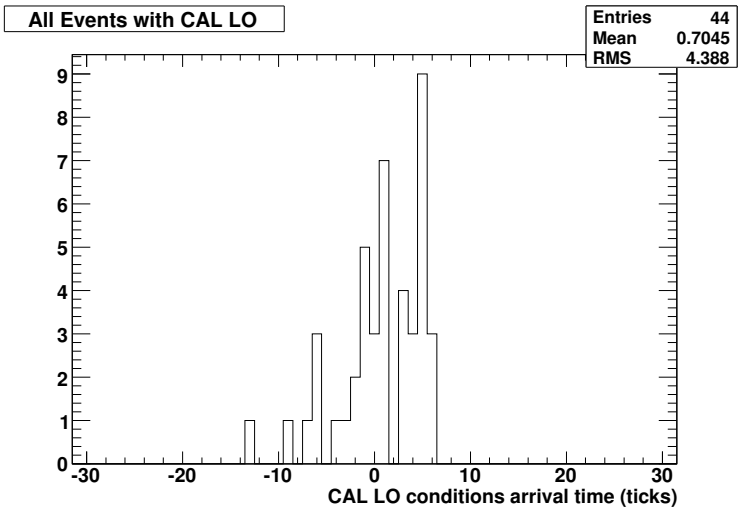

Figure 1: Timing plot for TKR vs. CAL

You can also run the analysis from the command line by typing `root -b -q runXXX.C` in the workdir directory. The same applies to running on LSF.

3.2 TREQ

There are 3 configurations for TREQ:

- **Treq_TkrCalGamma:** Times in TKR vs. CAL for photon-like events.
- **Treq_TkrCalMip:** Times in TKR vs. CAL for MIP events.
- **Treq_TkrAcd:** Times in TKR vs. ACD for MIPs.

For a discussion of the configurations see the document "Timing in during L&EO" listed in the "Applicable Documents" section.

3.2.1 treqCAL

Treq_TkrCalGamma and Treq_TkrCalMip use the treqCAL analysis. The input to the analysis is the digi and recon file for the run as well as the treq delays used in the config (0 for CAL and 5 for TKR in the original configuration). The output is a root file with the histograms, an html report and a number of plots used in the html report. There is an option to use a MIP selection which can be switched on and off in the steering file. This options should be used with Treq_TkrCalMip but not with Treq_TkrCalGamma. An example steering file is runTreqCALGamma.C.

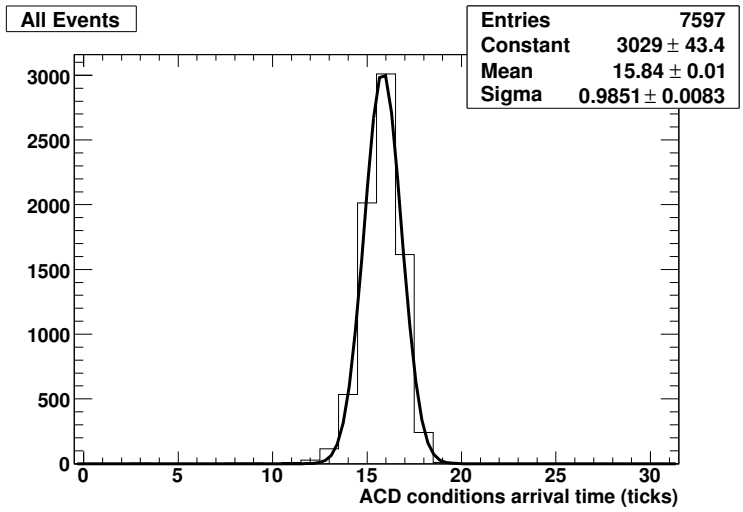

Figure 2: Timing plot for TKR vs. ACD

The quantity histogrammed is the condition arrival time of the CAL minus the condition arrival time of the TKR. An example plot is shown in figure 1. For each quantity the mean is calculated. The treq delays are subtracted and the result is then the delay of the CAL with respect to the TKR.
Treq_TkrCalGamma is the main configuration which should be used to set the TREQ delays. Treq_TkrCalMip is a cross-check to verify the timing for MIPs.

3.2.2 treqACD

treqACD is the analysis used for the Treq_TkrAcd configuration. The input to the analysis is the digi and recon file for the run as well as the treq delays used in the config (31 for ACD and 5 for TKR in the original configuration). The output is a root file with the histograms, an html report and a number of plots used in the html report. The analysis does a MIP selection and then histograms the ACD condition arrival time. The histograms are fit with a Gaussian as in the example plot in figure 2. The offset is subtracted. The result is the delay between ACD and TKR.

3.2.3 Interpreting the Results

The results from the 3 configurations are a large number of delays between the subsystems, including CALHI and CNO, and by tower and tile.
It should now be verified that the timing for towers and tiles is identical. Should a tower have a different timing then this would need to be investigated. If it is found that the timing difference is genuine the timing should be set separately for the tower which is possible.
If the timing between CALHI and CALLO should be found to differ then an informed decision

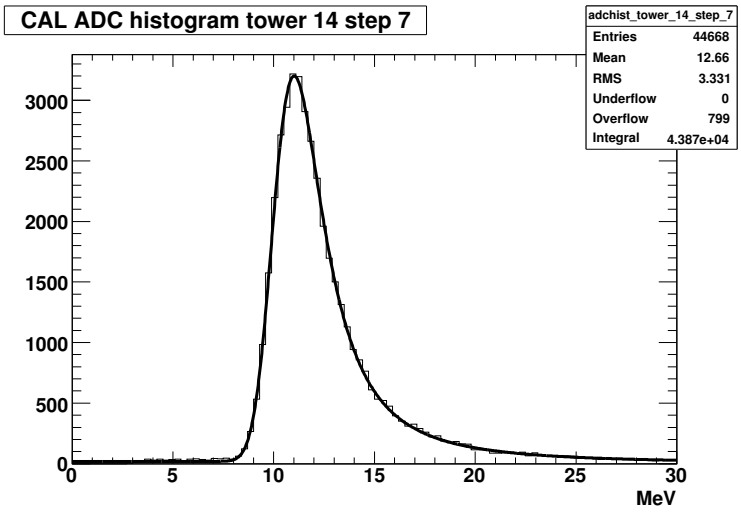

Figure 3: MIP peak for one scan point in tower 14

based on the physics has to be made. The same is true for ACD and CNO.
If MIP timing and photon timing differ then this difference should be taken into account for the TACK scan.
To put together the delays start set the CAL delay to 0 and the TKR delay to the measured delay between CAL and TKR. Then set the ACD delay to the difference between TKR and ACD plus the TKR delay from the previous step. Example: If CAL vs. TKR gives an optimal delay of 6 and TKR vs. TKR gives an optimal delay of 10, set CAL to 0, TKR to 6, ACD to 16.

3.3 TACK

3.3.1 treqACD

3.3.2 treqACD

The TACK scan is performed for a value of hold_delay of 0 in order to have access to a wide range around the peak. For regular physics running hold_delay is 20 because of the complications with the hitmap latching described above (the range of the hitmap_delay register is not wide enough to work with a hold_delay value of 0). To make sure that the switching from 0 to 20 does not affect the result of the ACD scan there is a verification suite of configs in addition to the main suite. Once it has been verified that hold_delay can be switched this way there is no need to rerun the "ver" suite of configs.
The TACK scan works in the following way: A MIP peak is recorded for a number of delay settings as shown in the example plot in figure 3. The MIP peak position is then plotted against the delay setting which reproduced the shape of the signal waveform as shown in figure 4. The delay that sits at the peak of the waveform is the optimal setting. Each configuration of a suite

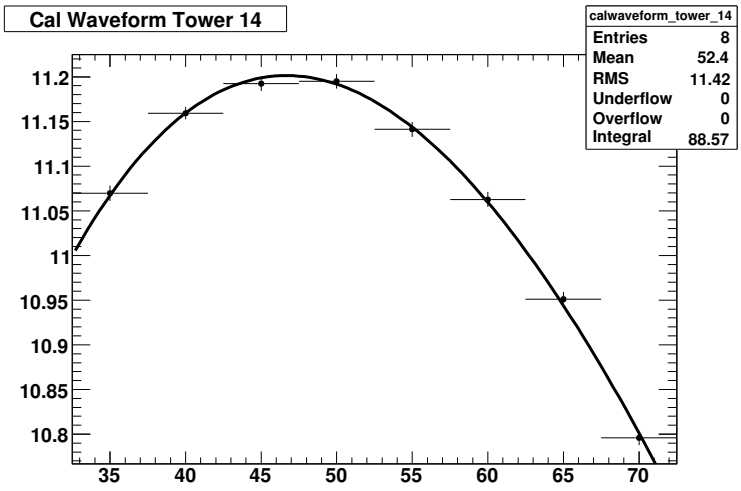

Figure 4: MIP peak for one scan point in tower 14

is one scan point. The configurations within a suite are identical except for the TACK delay settings for the 3 subsystems.
Main suite:

- **Tack_scan0**
- **Tack_scan1**
- **Tack_scan2**
- **Tack_scan3**
- **Tack_scan4**
- **Tack_scan5**

ACD hold_delay verification suite:

- **Tack_scan2**
- **Tack_scan3**
- **Tack_scan4**

The total delays for the ACD test runs are the same as for the regular runs of the same index.

3.3.3 Analysis

The analysis class used for the TACK Root Tree Analysis is takedata_mt. It selects MIPs and histograms CAL crystal energies per tower, ACD energy, and TKR hit efficiency. The TKR hit efficiency algorithm looks for the first and last hit on a track and then counts how many hits it found in between. Of course there may be dead material along the track so in the end the efficiency is not a strip efficiency but an actual overall hit efficiency. The absolute value is not of interest, it is the relative change in efficiency from scan point to scan point that counts.
For CAL the MIP histograms are made from the low energy readout chain. In order to get the timing for the high gain readout chain a trick is used: For heavy ion events which have four range readout the ratio between high range readout and low range readout is determined scan point by scan point. The low energy MIP peak position is then multiplied by this ratio to give the high energy MIP peak position.
Each run from the suites needs its own steering file. The inputs to the script are the digi and recon root files and the TACK delay settings, and the number of the scan point (0..5). The outputs are a root file and a text file. The names of the files can be set in the script but the text file must end on underscore scan point dot txt, for example testscanout_0.txt. An example script is runTakedata.C in the workdir directory.
Once you have run all the analyses for the scan points in the suite you can run the fit which takes the text and root files as an input, fits the MIP peaks using the CAL Langau function, then constructs the waveforms from the fit peak positions and in the end fits the waveforms. The output of the fit script is a root file with all histograms including fits and a report in html format. The input to the fit script is the first and last scan point index and the name of the ".txt" files with out the "_[index].txt" part. The script will then append the "_[index].txt" as it loops over the indices. An example fit script is testFit.C.

3.3.4 Interpretation of the results

For the TKR the expected result is an efficiency that falls off with increasing delay.
For ACD the result is one waveform whose peak position gives the optimal delay for the TACK.
For the CAL we get waveforms and peak positions by tower for the two readout chains. From tests on the ground we expect the timing to be the same for all towers but in principle the parameters can be set individually by tower. If the high energy readout chain which has not been tested on the ground should have a different optimal delay then a decision based on physics has to be made.
Note that the TACK delays depend on TREQ delays and window width so if there are any changes made to those parameters then the TACK delays have to be adjusted.

4 Other analyses

There are 2 more Root Tree Analyses in the package: **vetoeff** checks that the mechanism of vetoing coincidences between TKR and shadowing tile works. It was used extensively on the

ground for testing purposes.
mippeak is an analysis that generates an ACD MIP histogram from a run.