GLAST LAT TECHNICAL DOCUMENT	Document # LAT-SS-286	Date Effective Draft 29 July 01
	Prepared by(s) J.J.Russell	Supersedes None
	Subsystem/Office Electronics Subsystem	
Document Title **LAT Global Trigger Conceptual Design**		

Gamma-ray Large Area Space Telescope (GLAST)

Large Area Telescope (LAT)

LAT Global Trigger Conceptual Design

CHANGE HISTORY LOG

Revision	Effective Date	Description of Changes

1 Overview 4
2 A Functional Definition 4
3 Trigger Primitives 5
3.1 Tower Primitives 6
3.1.1 CAL Trigger Primitives 6
3.1.2 TRACKER 3-in-a-row Primitive 7
3.2 ACD Primitives 7
3.3 Transmission of Primitives from the Source to the Global Trigger 7
4 Construction of a Named Trigger Request 8
4.1 Reduction of the Trigger Primitives 8
4.2 Named Trigger Requests 8
5 Inhibits 8
5.1 Flow-Control 9
5.2 Prescalers 9
6 Trigger Parameterization 9
6.1 Readout Styles 9
6.2 Dispatch Classes 10
6.3 Arbitration of Simultaneous Named Trigger Requests 10
7 Trigger Request Message 10
7.1.1 Trigger Message Format 11
8 Control Functions 11
8.1 Marker Events 12
9 Trigger Live Time Monitoring 12
10 Trigger Data Contribution 13
11 Example 13

Table 1 Trigger Message Format 11
Table 2 Named Trigger Definitions 14
Table 3 Readout Type Definitions 14
Table 4 Dispatch Class Definitions 14

1 Overview

At the highest level, the GLAST global trigger (GLT) accepts a number of input signals and, in real time, decides whether to issue a L1T Accept signal, latching the front-end signals. An Algorithm Engine combines the input signals, called trigger primitives, to form specific named triggers. The named triggers, in turn, are transformed into a L1T Request. The transformation of a named trigger to a trigger request is denied if a dead time signal indicates the instrument cannot be readout at this time. Prescalars are associated with each trigger, allowing the leakage of events from high-rate triggers.

The global trigger also includes monitoring and configuration and control functions along with its own contribution to the event data.

2 A Functional Definition

Figure 1 shows a block diagram of the GLT which illustrates a possible implementation having all the desired properties. The *Algorithm Engine* logically transforms a collection of *trigger primitives* to a number of *named triggers*. Each resulting named trigger is subjected to a prescaler and a dead time signal. A named trigger carries more information than a simple GO/NOGO. It must also specify what information is to be collected, how it is to be collected and where (which event processing CPU or CPUs) it is to be dispatched. The GLAST trigger may request only one type of data from the detector (standard physics data). Variations in the readout style may be requested from the CAL and the ACD. For example, on CNO triggers, the CAL would switch from auto-ranging and zero suppressing to only zero suppressing

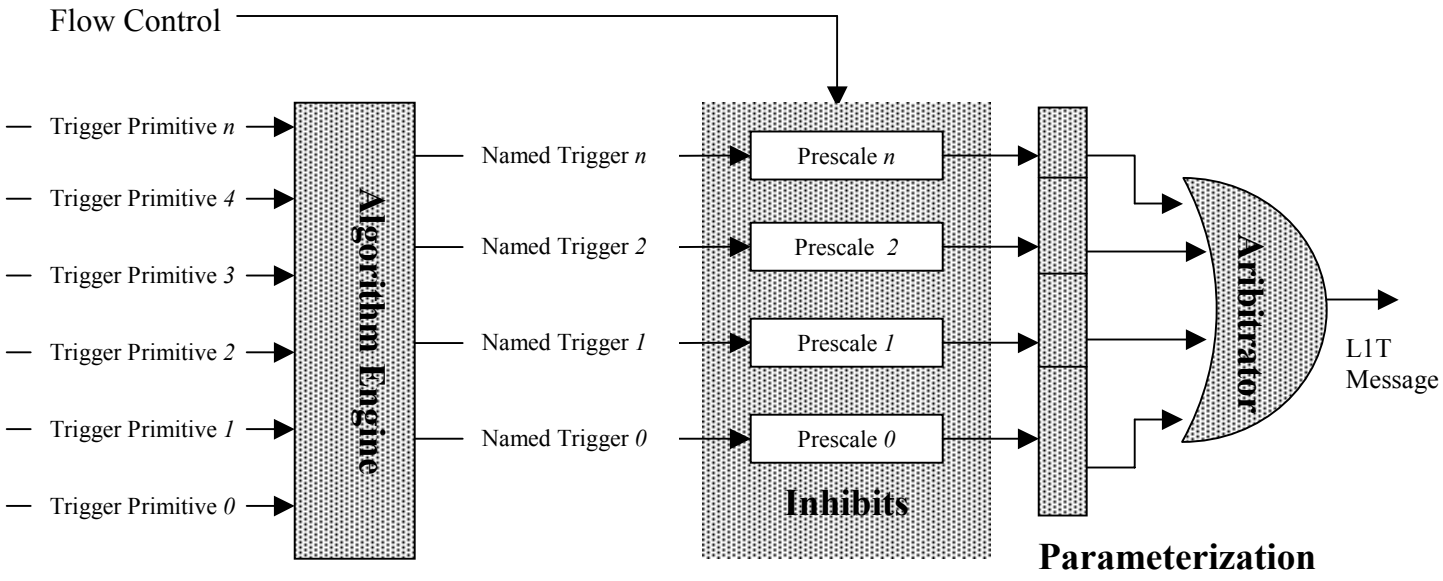

Figure 1 Block Diagram of the GLT

A detailed discussion of each of these concepts and an example implementation is presented.

3 Trigger Primitives

The raw information needed to form a trigger is the trigger primitives. Each of the 16 towers and the ACD contributes a number of trigger primitives from which a trigger decision is made. These trigger primitives then are transmitted to the GLT where they are combined and reduced to a more manageable number. All trigger primitives must be time-aligned to 50nsecs (the system clock rate) and stretched to ~500nsecs to cover jitter. In all cases, masks and/or overrides are provided to handle potentially malfunctioning inputs.

The trigger primitives fall into two classes; those from one of the 16 towers and those from the ACD.

A tower contributes 3 signals or a total of 48 for the 16 towers to the GLT.

- A TKR 3-in-a-row signal
- A CAL LO discriminator signal
- A CAL HI discriminator signal

The ACD contributes the following 33 primitives

- 16 signals indicating a signal from the tiles that shadow a GLAST tower
- 15 signals consisting of 3 signals from each of the 5 planes (the 4 side + the top) of the ACD
    - 1 or more LO discriminator signals from the tiles in a plane
    - 2 or more LO discriminator signals from the tiles in a plane
    - 3 or more LO discriminator signals from the tiles in a plane
- 1 signal giving the OR of the HI discriminator signals from the 25 top tiles.
- 1 signal giving the OR of the HI discriminator signals from the 80 side tiles

The ACD group defined a primitive of the OR of the LO discriminators from one side AND'd with the corresponding signal from the opposite side (two signals). These are not truly primitives, but can be derived from the 1 or more LO discriminators signals of the tiles in a plane.

3.1 Tower Primitives

The TKR contributes one trigger 2-bit primitive and the CAL contributes two trigger primitives. These primitives are combinations of very low-level signals available from the detector front-end electronics.

3.1.1 CAL Trigger Primitives

Each of the 96 log ends of the Calorimeter develop two discriminator signals relevant to the trigger, a so-called LO discriminator value and a HI discriminator value. The HI discriminator value is set at a level meant to ensure reasonable efficiency for energy depositions > 10 – 20 GEV. The trigger primitives are the ORs of all the LO discriminator values and the ORs of all the HI discriminator values. Note that this is a poor man's energy trigger since this is not a sum of the energy in the CAL. Individual noisy discriminator channels can be masked out.

At this time, the CAL group is considering a slightly different, but more useful, definition for the CAL HI primitive. This only changes its meaning, not the way the GLT system handles it.

3.1.2 TRACKER 3-in-a-row Primitive

Each of the 18 X layers and 18 Y layers of the TKR contributes one bit of information representing the OR of all the hit strips in a layer. Again, individual noisy layers may be masked out. Also remember that the TKR subsystem has the ability to mask indiviual noisy strips.

Logic is performed on this information to determine whether any pattern of 6 consecutive layers (3 X and 3 Y) has been hit. The result is a 2-bit quantity that roughly indicates where the 3-in-a-row starts. This allows some degree of selective pairing of a tracker 3-in-a-row with an appropriate ACD tile. One of the four possible values is used to represent a 3-in-a-row starting in the plane adjacent to the top plane of ACD tiles. The other 3 values are used to select which row or rows of the ACD tiles in the side planes will be considered as a possible match.

3.2 ACD Primitives

Each ACD tile produces a LO and HI discriminator signal. The LO discriminator signals the passage of a minimum ionizing particle. The HI discriminator signals the passage of a heavy nucleus, a so-called CNO. It is generally used to generate a trigger accept, as contrasted with the LO discriminator which is typically used as a veto.

3.3 Transmission of Primitives from the Source to the Global Trigger

Once the trigger primitives have been formed, they must be transmitted from the source, either the ACD or the tower, to the GLT. There are 4 * 16 tower trigger primitives + 33 ACD primitives. (This counts the 2-bit tracker 3-in-a-row as 2 signals.) If each requires 2 separate differential pairs (one for redundancy), it will take an imposing 97 x 2 x 2 = 388 wires between the towers and the GLT.

A possible way to reduce the number of wires is to serially encode the information on fewer wires. Optimistically, only 17x2 differential pairs (the x 2 for redundancy) are needed. The savings is substantial, 388 wires down to 72. However, this strategy is not without its costs. Once any trigger primitive on a wire becomes active, a message encoding the state of primitives must transmitted. No other trigger information can be transmitted for at least the length of the trigger message.

For the tower based primitives, 4 bits + a start bit need to be transmitted, costing 250nsecs/message. If the overall rate of the tower primitives is 10KHz, this results in a .25% dead time. This small dead time may be a good system trade, since this technique could reduce the wires needed to transport the tower based primitives from 256 to 64. Monitoring of this dead time is non-trivial and needs to be explored.

The ACD situation is not as encouraging. The nominal counting rate of ach ACD tile is ~1KHz. Many of the ACD trigger primitives OR large numbers of tiles together (up to the full number of ~100). This combined with the number of bits (~35) would lead to a dead time of 100 x 1KHz * 35 * 50 nsecs = 17.5%. A better strategy may be to use serializers to effectively up the bit rate between the ACD and GLT. The current candidate takes 21 inputs and transmits them on 3 outputs plus 1 clock. It would take 2 to handle the35 ACD signals, reducing the 132 wires to 16.

4 Construction of a Named Trigger Request

This section details how the trigger primitives are transformed by the GLT into a named trigger request.

4.1 Reduction of the Trigger Primitives

Once the trigger primitives arrive at the GLT, they are combined to form more useful, refined trigger primitives. If these quantities could have been formed locally they would have, but the low level trigger primitives are geographically distributed making the GLT the logical place to do the combination. The list includes the:

- OR of all tower's CAL LO
- OR of all tower's CAL HI
- OR of all tower's ACD HI
- OR of all tower's 3-in-a-row
- OR of all tower's (3-in-a-row & shadowing ACD tiles), Vetoed 3-in-row
- AND of opposite side ACD vetoes, at least one side with > 1 hit

These quantities will serve to generate the named triggers.

4.2 Named Trigger Requests

The refined trigger primitives defined in the previous section are now used to generate named trigger requests. Examples (not an exhaustive list) of named trigger requests are

- 3-in-a-row
- 3-in-a-row with ACD Veto
- High Energy CAL
- Lo Energy CAL
- CNO
- Periodic

A maximum number of 8-12 named trigger requests seems reasonable. Note that at any given time 0 or **more** of these named trigger requests could be active. Before a named trigger request results in a trigger message it is subjected to a dead time signal, i.e. the detector must be able to absorb a new event and the named trigger request must not be inhibited by its prescaler.

5 Inhibits

A named trigger request is subject to two types of inhibits, flow control and prescalers.

5.1 Flow-Control

The named trigger request is subject to the flow control signal. This signal is part of the dataflow system and indicates when insufficient resources exist to absorb a new trigger. For example, if the TKR on *any* tower has all 4 of its front-end buffers occupied, the system is unable to absorb a new event and the flow-control signal inhibits the named trigger request.

5.2 Prescalers

Prescalers allow the implementation of monitoring triggers and provide a means of limiting a named trigger request's rate without suppressing it altogether. A looser implementation of a frontline physics trigger can be prescaled, making it possible to cross check a sub sample of the frontline physic's triggers against looser triggers.

Only simple countdown prescalers should be used for physics triggers. While other types of prescalers are possible, for example timeouts or rate limiters, calculating the effective live time when using them is a tricky business. The effective live time for simple count down prescalers is 1/*count_down*.

For triggers that do not need live time calculations, such as a trigger that monitors the pedestals of the CAL and ACD by disabling the zero suppression, a simple timeout trigger could be used. The only difference between a countdown and a timeout trigger is the prescaler's counter is decremented by the actual named trigger request in the former case and by the clock in the later case.

6 Trigger Parameterization

Each named trigger request has an associated set of information. This information indicates the information to be gathered, called the readout style, and the destination of the event, called the dispatch class. Readout styles must be transmitted at trigger time, since they modify the information being gathered. Information directing the event to its destination CPU(s) can be, but need not be transmitted with the event. Given the expense of transmitting information in the trigger message, a better choice is to gather this information at event build time.

6.1 Readout Styles

Each named trigger has an associated set of information that defines the components to be readout and how they are to be readout. In order to simplify the DAQ readout path, the trigger may only request one type of data to be readout. At one time the so-called Low Rate Physics (rate counters) was also transported along the data path. This led to unnecessary complications involving the logic to handle the requests for disparate types of data. Physics data consists of reading out data from each of the TKR, CAL, ACD and GLT itself and is the only data transported along the data path. The Low Rate Physics is transported along a different path.

However, the readout style of the event data **is** specified by the GLT. The readout styles are:

- CAL auto-ranged disabled
- CAL, zero-suppress disabled
- ACD pulse height, auto-range disabled
- ACD pulse height, zero-suppress disabled

6.2 Dispatch Classes

A dispatch class is used in deciding which CPU(s) will handle a particular event The likely dispatch classes are:

- Physics, this handles all physics data independent of the trigger type
- Monitor, handles all monitor triggers, such as those from CNO events
- Control, handles control events, i.e. begin and end triggers

For example, physics data will likely be round robined to each of the available event processors using the event sequence number as a dispatch number. Monitor data may be directed to a specific CPU pre-selected to handle those event types. Control events (see below), are directed to all CPUs.

Dispatch classes may or may not have an identifiable piece of hardware in an actual implementation. For instance, given the relatively small number of named triggers, it is probably easier to associate the dispatch information directly with the named trigger.

6.3 Arbitration of Simultaneous Named Trigger Requests

The GLT must handle the case when multiple named trigger requests occur simultaneously.. The scheme chosen allows at very simple arbitration. The sense of the signal allows readout styles to be OR'd. This, in some sense, maximizes the information content. The scheme does not work if mutually exclusive types of readouts are desired. This is a limitation of the implementation, but seems reasonable.

If dispatch classes can also be OR'd. It is the responsibility of the Flight Software to ensure that events sent to multiple CPUs are according to the dispatch class that sent them to that CPU. For example, if a event is a member of both the Physics and CNO Monitor dispatch class, then one CPU may process it as a Physics event and an other may process it as a CNO event.

7 Trigger Request Message

When a trigger is declared, the following information **must** be attached to the trigger message sent to each contributor.

- A start bit indicating a trigger is being requested.
- The readout style.

The trigger message conveys enough information to allow the front-end electronics to initiate the acquisition of the desired data and specifies exactly how the desired data is to be formatted.

It may be **desirable** to also send the following information

- An event sequence number
- The time of the event

These two numbers uniquely name the event. Current designs have this function being handled locally on each source. Having these numbers dispensed by the trigger is may be more robust, however, it may not be practical.

The purpose of the event sequence and time is only to uniquely identify an event while it in the DAQ pipeline. As such, the range of these numbers can be limited, reducing the number bits required to represent these values. These numbers are not intended to be used to generate overall an event sequence number nor to provide an absolute timestamp.

The trigger message must be delivered within approximately 2-3*u*secs of the initiating event to correctly latch the front-end signals. A jitter of 500*n*secs must be maintained. Note that all this requires is that the trigger start bit obey these conditions. The remainder of the trigger message is allowed to arrive somewhat later. For example the style information is not needed for another ~1*u*sec and the event identifiers, in the form of the sequence number and timestamp are not needed until the data is committed to the readout FIFOs, likely not until ~5*u*secs.

7.1.1 Trigger Message Format

The following is a possible trigger message format. The trigger message is a bit serial message. In this example bit 0, is tranmitted first.

Bits	Meaning
0	Start Bit
1	Disable CAL Auto Range
2	Disable CAL Auto Zero Suppression
3	Disable ACD PHA Auto Range
4	Disable ACD PHA Zero Suppression
5-20	Time (low 16 bits)
21-36	Sequence Number (low 16 bits)

Table 1 Trigger Message Format

This format is constructed so that the most time critical information arrives first.

8 Control Functions

The master CPU must be able to control and configure the GLT. Configuration commands setup the various tables and prescalers on the GLT. Commands are issued using the same standard GLAST serial command protocol also used to command and configure the towers and the ACD subsystems.

The CPU must have the ability to request a trigger. These commands cause data to be inserted into the data stream in a synchronized fashion and behave exactly as a trigger.

The only differnce is that the requester is the master CPU, not the front-end electronics.

8.1 Marker Events

Marker events are events requested by the CPU in order to synchronize the DAQ pipeline. These commands should consist of a command field indicating type of marker event plus a small data field (8 bits is overkill) that acts as an instance identifier.

In order to ensure that this command is not discarded by the dead time signal, the GLT must queue this request, or by some other means, ensure that this request is delivered. The nature of this trigger will be noted in the GLT's contribution to the data stream allowing the receiving CPU to identify it as such.

Two important marker events are *begin* and *end* markers, needed to start and stop the flow of data. For example, a stop run is accomplished by issuing the appropriate command to the GLT, disabling the trigger and then waiting until the data generated by the trigger request is delivered to the event processing CPUs. The event processing CPUs, upon receiving the event generated by the stop run command, now knows that the run is over and that all subsequent events should be tossed. This effectively drains the data pipeline without having to resort to timeouts. It is a goal of the GLAST DAQ to use timeouts only in error situations.

The informational content of the such control triggers is minimal, only a timestamp, an L1T count and a reflection of the marker type and marker ID is absolutely necessary. This will allow the upstream logic to assemble these packets just as it would for a normal event. For simplicity, these triggers will also cause the front-end electronics to produce data just as any other trigger.While not strictly necessary, these commands could also enable and disable the towers trigger. This **is** necessary if one needs a way of indivisibly starting/stopping the trigger **and** inserting the data packet.

Other CPU issued triggers act much more as normal triggers. These could be used to issue keepalive triggers, random triggers or special triggers to gather monitor information. This latter style requires that, in addition to just issuing a trigger request, the command carries the readout style.

9 Trigger Live Time Monitoring

The effective live time of the LAT is monitored by the GLT. The GLT maintains two non-resetting counters, one counting the elapsed time and one counting the elapsed time gated by the flow control signal. As previously mentioned, the flow-control signal indicates ability of the electronics to absorb or take another event. The flow-control signal includes the any dead time the trigger itself imposes while it is busy computing trigger decisions. Note that the trigger may produce dead-time independent of whether any L1T is actually issued.

This information is attached to each event. When the sequence of events is time ordered, the effective live time between any two events can be computed by forming the ratio of the differences. For example the live time for a run can be computed by examining these counters in the first and last event in the run.

10 Trigger Data Contribution

The GLT contributes a block of data to each triggered event. This data will contain

- The live time counters
- The state of the trigger primitives.
- The named triggers requested by this trigger.
- Marker type and ID, if any
- Destination Information, i.e. which CPUs to send the event to

The last item, the *Destination Information* presents a design choice. Either the trigger can directly assign the destination(s) or enough information is attached to the event so that each destination makes its own decision when the event arrives. The latter is a somewhat more modular approach. This document favors having the GLT make the decision. In either case, this information is placed in the trigger data contribution. The *Event Builder* uses this information to determine the destination CPUs when it pulls the trigger data contribution. Some implementations of the *Event Builder*, may require the GLT information block to be read first so that the destination of the remaining contributors is known immediately.

The other items on the list aim to satisfy the goal of monitoring the live time of the LAT and the performance and integrity of the GLT itself. For example, the list of named output triggers generated may be checked against the trigger primitives, verifying the operation of the Algorithm Engine.

11 Example

This is a semi-realistic example of how the GLT would be used. Consider the following named triggers defined by Table 2. The numbers listed under the ReadOut Class and Dispatch Class are defined below in Tables 3 & 4 respectively.

Two new primitives have been added; one reflecting a CPU request and one representing a primitive that is always true. The *True* primitive when combined with a timeout prescaler, allows one to implement periodic triggers. In the interest of simplicity the CAL LO and the additional ACD veto primitives have been omitted from this example.

TRIGGER		TRIGGER PRIMITIVE						Pre-Scale	Read-Out Class	Dis-Patch Class
	NAME	ACD HI	CAL HI	TKR vetoed	TKR	CPU	True			
0	TKR, not Vetoed	X	X	0	1	X	X	1	0	0
1	TKR (any)	X	X	X	1	X	X	10	0	0
2	High Energy	X	1	X	X	X	X	1	0	0
3	CNO	1	X	1	1	X	X	1	1	1
4	ACD w Pulse Height	X	X	1	1	X	X	10Hz	2	1
5	ACD & CAL Pedestal	0	0	0	0	X	X	.1Hz.	3	1
6	Periodic/Keep Alive	X	X	X	X	X	1	30Hz	0	0
7	CPU Control	X	X	X	X	1	X	1	0	2

Table 2 Named Trigger Definitions

READOUT TYPES		DEFINITION
#	Name	
0	Physics	ACD discriminators, CAL (zero-suppressed, auto-ranged), TKR
1	CNO	ACD discriminators (pulse height), CAL (zero-suppressed only), TKR
2	ACD PH	ACD discriminators, CAL (zero-suppressed, auto-ranged), TKR, ACD Pulse Height
3	Pedestal monitor	ACD (disable zero suppress), CAL (disable zero suppress), TKR

Table 3 Readout Type Definitions

DISPATCH CLASS		DEFINITION
#	Name	
0	Physics	Handle as normal physics data (round-robin)
1	Monitor	Handle as monitor data (dedicate CPU)
2	CPU Control	Handle as CPU requested control trigger (begin/end run) (all CPUs)

Table 4 Dispatch Class Definitions

The dispatch classes along with the event sequence number serve as input in selecting which CPU(s) a given event is destined..

As an example, suppose the primitives are ACD HI=1,CAL HI=1,TKR (vetoed) = 1 TKR=1. This would activate the named triggers 1,2,3,4. Now further assume that neither the dead time nor the prescalers inhibit these named triggers from producing a trigger request. The resulting trigger message is.

- READOUT LIST = CAL (zero-suppress, no auto-range), ACD discriminators and pulse height.

The dispatch list is PHYSICS and CNO.

- DISPATCH LIST = PHYSICS, CNO

The dispatch list along with the event number will be used to index a lookup table giving the complete list of CPUs this event will be sent to. Once received, each CPU will process the event in accordance to its assignment. That is, the CPU designated to handle CNO events will process it for CNO information, the CPU designated to handle it as a regular event will process as a regular event. It is possible that this event could be sent to that this event will be sent to 1 or 2 CPUs depending on the state of the round robining..