LAT INTERFACE DOCUMENT	Document # **LAT-SS-00586-D2**	Date Effective DRAFT 7/1/02
	Author(s) Richard Claus	Supersedes LAT-SS-00586-D1
	Subsystem/Office Integration & Test/Online	
Document Title **I&T Online - Subsystem Interface Control Document**		

This copy dated: 7/11/2002 7:43 PM

I&T Online - Subsystem Interface Control Document

CHANGE HISTORY LOG

Revision	Effective Date	Description of Changes
01		Initial Release
02	June 18, 2002	Updated to reflect the implemented situation, addition of AEM, GLT

TABLE OF CONTENTS

1. Purpose

This document presents a description of the interface between the GLAST LAT I&T Online System and the LAT subsystems.

2. Scope

The scope of this document covers the subsystem interface of the GLAST LAT I&T Online system. It describes how the Calorimeter and Tracker subsystems command the Tower Electronics Module (TEM) and how they receive responses from it. Similarly, the ACD and Global Trigger subsystems correspondingly use the ACD Electronics Module (AEM) and Global trigger Electronics Module (GEM), respectively.

To avoid duplicating the contents of the SCL documentation here, it is expected that readers intending to work with SCL be familiar with the SCL User's Guide, in particular Section 2 - SCL Language Reference and have worked the tutorial given in the SCL Installation Guide.

3. Definitions

3.1. Acronyms

ACD	AntiCoincidence Detector
API	Application Program Interface
CU	Calibration Unit
EGSE	Electronics Ground Support Equipment
EM1	Engineering Model 1 EGSE
EM2	Engineering Model 2 EGSE
FITS	Flexible Image Transport System
FSW	Flight SoftWare
FU	Flight Unit
GASU	Global trigger ACD Signal distribution Unit
GEM	Global trigger Electronics Module
GITOT	GLAST I&T Online Teststand (network name of the embedded system processor)
GITOW	GLAST I&T Online Workstation (network name of the host workstation)
GLT	GLobal Trigger
GUI	Graphical User Interface
I&T	Integration and Test
ICS	Interface Control Systems – makers of SCL
IDL	Interactive Data Language
LAT	Large Area Telescope
LSW	Least Significant Word
MSW	Most Significant Word
NFS	Network File System
PC	Personal Computer
QU	Qualification Unit
ROOT	Rene's (?) Object Oriented Tool
RTE	Run Time Engine
SBC	Single Board Computer
SCL	Spacecraft Control Language
TBD	To Be Determined
TBR	To Be Reviewed

TBS	To Be Supplied
TEM	Tower Electronics Module

4. Applicable Documents

LAT-TD-00035-01	LAT Coordinate System
LAT-TD-00606-D1	LAT Inter-module Communications – A reference manual
LAT-TD-??????-D1	TEM Communications Driver – Software User's Guide
LAT-TD-00605-D1	The Tower Electronics Module (TEM) – A Primer
LAT-TD-00639-D1	The ACD Electronics Module (AEM) – A Primer
LAT-SS-00208-D2.V8	Conceptual Design of the GLAST Calorimeter Readout Controller (GCRC) ASIC
LAT-SS-00424-D2	Design Description of the GLAST Calorimeter Readout Front-End Electronics (GCFE) ASIC
LAT-SS-00170-D2.D9	Conceptual Design of the GLAST Tracker Readout Controller (GTRC) ASIC
LAT-SS-00169-D4.D4	Conceptual Design and Specification of the GLAST Tracker Readout Front-End Electronics (GTFE) ASIC
LAT-TD-00719	FITS format for the Engineering Model tests
ICS-???	SCL Installation Guide
ICS-???	SCL User's Guide

5. Introduction

The Spacecraft Control Language (SCL) test executive from Interface Control Systems, Inc. (ICS) has been chosen to provide the tools for creating the commanding and telemetry databases for the GLAST I&T test stands. Its architecture leads to the general component diagram of a test stand shown in Figure 1. The functional blocks shown in the diagram don't necessarily all reside on one processor. Standard network protocols can be used between the blocks for communication.

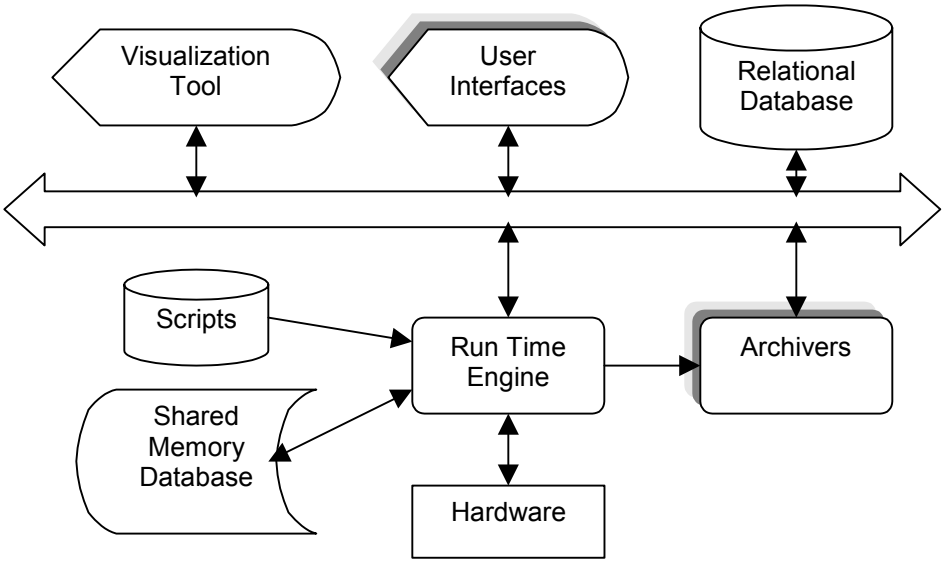

Figure 1 – Test stand software component diagram

The SCL Run Time Engine (RTE) executes application scripts using data stored in the Shared Memory Database. To balance processing loads, more than one RTE may be configured to run on different computers. Each RTE has its own database and scripts.

6. Architecture

The architecture that has been settled on has SCL running on both a PowerPC embedded processor running the VxWorks real time operating system (RTOS) and a PC running the Windows 2000 operating system. ICS does not support all platforms to the same level, which means that the Online system developed on the Windows and VxWorks platforms won't necessarily work on another combination of platforms. The I&T Online group is not able to support multiple platforms.

Figure 2 shows the EM-1 test stand architecture. The switch connecting the SCL compiler to one of the RTEs is activated with the `connect` command, e.g. `connect gitot_cmdin`.

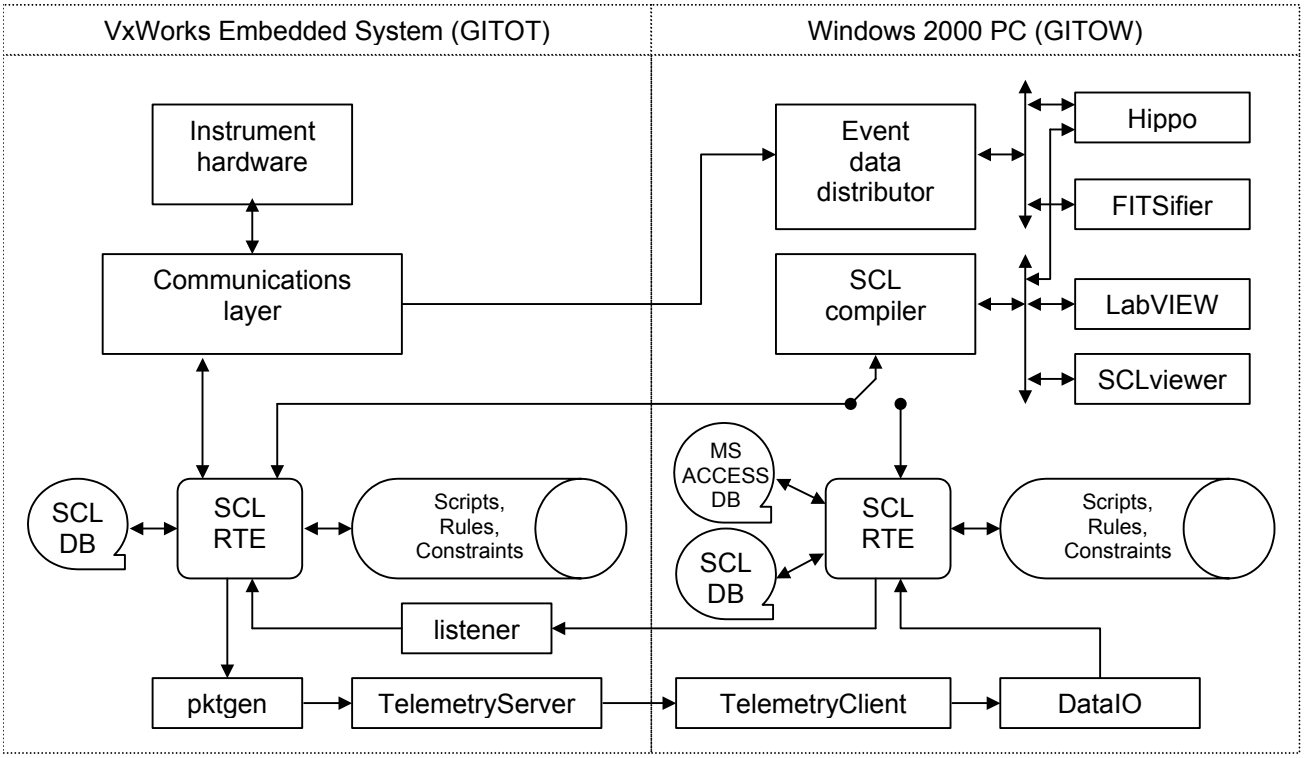

Figure 2 - EM-1 Architecture

7. Naming

The SCL User's Guide, Section 2 SCL Language Reference - *Identifiers* says:

> In SCL, identifiers identify the *names* of scripts, rules, constraints, functions, global variables, local variables, or fields. User-defined identifiers can't be keywords.
>
> An identifier can be a maximum length of 31 characters. Identifiers must begin with an alphabetic character, and can include any combination of the following characters.
>
> ```
> A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
> a b c d e f g h i j k l m n o p q r s t u v w x y z
> 0 1 2 3 4 5 6 7 8 9
> $
> ```
> _ (underscore)

Note that SCL is not case sensitive. In the interest of minimizing confusion, it is recommended that when a particular casing is chosen for an identifier, that casing be used throughout.

SCL keywords are those identifiers that are supplied with the SCL product and are controlled by ICS. Keywords are *reserved words*, meaning that creating identifiers identical to SCL keywords will cause unpredictable behavior. There is only one namespace in the SCL scripting environment and unfortunately ICS does not follow any particular rule in choosing keyword names. To minimize creating namespace conflicts, identifiers shall be prefixed with a subsystem code according to Table 1. This prevents conflicts from arising when unrelated identifiers with otherwise identical names are brought together in a composite system. For example, Calorimeter and Tracker `Init` functions, are made differentiable when given the names `C$Init` and `T$Init`. The prefixes were chosen to minimize typing and under the assumption that the list of subsystems won't grow significantly.

Prefix	Meaning
A	ACD
C	Calorimeter
T	Tracker
O	Online System
I	I&T
F	Flight SoftWare

Table 1 – Subsystem prefix codes

To avoid the converse problem, wherein a reader incorrectly assumes an identifier belongs to a particular subsystem because of its initial letter (e.g., `CEIL` being mistaken for the Calorimeter's `EIL` function), a dollar sign shall be used to separate the prefix from the root. Any identifier starting with, for example, `C$`, can immediately be recognized to belong to the Calorimeter subsystem.

The subsystem prefix is an example of a *field*. Multiple fields may be used to make up an identifier. The dollar sign shall be the delimiter used for field separation purposes, leaving the underscore character for word separation purposes. Subsystems are free to further partition up their namespace by creating multi-word fields within identifiers separated by the dollar sign, e.g., `C$Trg$Clear_Counters` versus `C$Env$Clear_Counters`.

8. Coding Rules

We have avoided providing a set of coding rules, as the Online is not interested in getting into the business of policing coding practices. In addition, it is difficult, if not impossible to arrive at a set of coding rules that all parties will agree to. Therefore, use of good taste and judgment and common sense is required and expected when writing code that is considered a deliverable to the Online. Implementers should set up a set of rules they're comfortable with, or use the Online's style as an example, and apply them consistently in their coding practices. The sections *SCL Programming Techniques*, *List Features* and *Code Format* in the SCL User's Guide are an appropriate starting point for implementers to establish a set of coding rules.

8.1. Guidelines

Each file should contain a header that identifies aspects of the file like an abstract, contact information and the creation date. An example template is shown in Figure 3. Examples and commentary shown in the figure are between angle brackets (<>). Ideally, key information would be easily extractable from the file into an easily web browsable format like html using a package like doxygen. Currently, such an option is not available (doxygen doesn't recognize the SCL scripting language).

```
--
--  Facility:
--      <Facility name, e.g. ACD, CAL, TKR, Online, etc.>
--
--  Abstract:
--      <Brief introduction to what the file contains>
--
--  Author:
--      <Name, Institution, project; e.g., R. Claus, SLAC - GLAST I&T/Online>
--
--  History:
--      <Date – Name - Action; e.g. July 2, 2002 – R. Claus – Created>
--
--  Copyright:  <If required or desired>
--      <Copyright notice>
--
```

Figure 3 – Example script header template

A comment block similar to the file header should precede each function definition. An example template is shown in Figure 4. This block should be tailored according to whether it describes a "subroutine" ("script" in SCL lingo), a function, a rule or a constraint.

```
--
--  Description:
--  <Description of the function>
--
--  Arguments:
--        <arg1 – Description of first argument>
--        <arg2 – Description of second argument>
--
--  Return value:
--  <Meaning of the function return value, or "none">
--
```

Figure 4 - Example function header template

9. The SCL Project Hierarchy

As shown in Figure 5, the SCL project hierarchy is composed of several levels:

- project
- database
- record
- field

Only one project can be loaded at a time in an operational entity (an NT or Solaris "process", or VxWorks boot cycle). A *project* consists of one or more databases, scripts, rules and constraints. It can be arbitrarily large with one or more of tasks. For example, a subsystem may develop a test as a single project and later integrate that test into a project that contains a set of tests. Still later, this set of tests might be submitted for integration into a LAT project, together with other subsystem tests. This process of condensing multiple projects to a single project highlights the need for ensuring that namespace and similar conflicts don't occur.

Records are instantiated from a list of templates to form a *database*. Multiple databases may be created and loaded in a project. As ICS envisioned its product, a *record* is essentially a single valued entity with a series of attributes stored within the record as *fields*. Fields can optionally be further decomposed into a set of bit-fields or can be one of the standard data types, e.g. `chars`, `shorts`, `longs`, `unsigned chars`, `unsigned shorts`, `unsigned longs`, `floats` or `doubles`. Fields can also contain strings. There is no support for 64 bit integers.

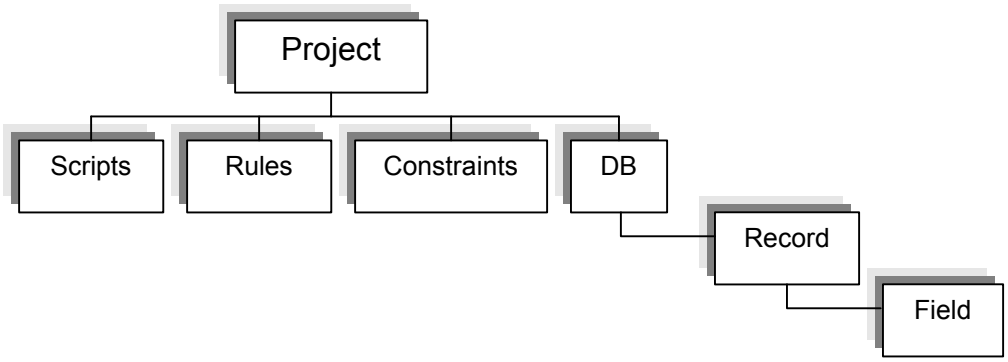

Figure 5 – SCL project hierarchy

To provide a consistent scripting environment over the various types of test stands (EM1, EM2, CU/QU and FU), at least the subset of an SCL project's databases corresponding to some part, or all, of the LAT, will be loaded onto the embedded system. In order to meet functional requirements, Online has augmented the SCL supplied record-type list with so-called "user-defined records". These records have special fields that allow records to be linked together to form a hierarchy. Additionally they contain fields that reflect the individual hardware registers and that can execute the so-called dataless commands (described later).

10. The LAT Electronic Hierarchy and Addressing

At its highest level, the LAT hierarchy (Figure 6) consists of 16 Tower Electronics Modules (TEMs), one ACD Electronics Module (AEM), and one Global trigger Electronics Module (GEM). TEMs and AEMs contain a number of Cable Controllers (CCs). Similarly, each CC contains a number of Read-out Controllers (RCs), and finally, each RC contains a number of Front Ends (FEs). Each of these functional blocks is referred to as a *node*.

For the TEM, the number of instances of functional blocks of each type is listed below in Table 3. Table 3 also shows the mapping between the functional blocks and the available SCL record types. The prefixes GC and GT stand for GLAST Calorimeter and GLAST Tracker, respectively, so GCCC is GLAST Calorimeter Cable Controller, etc.

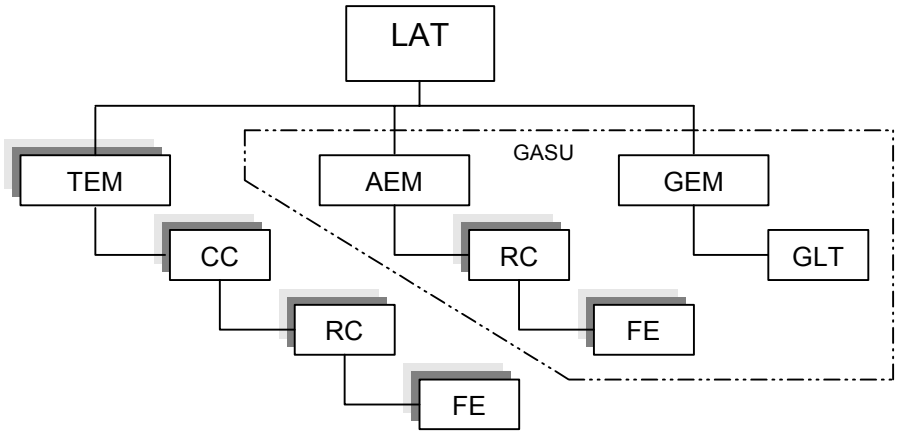

Figure 6 – LAT electronics type hierarchy

The Global Trigger ACD Signal distribution Unit (GASU) contains the AEM and GEM functional blocks, amongst other things. Since the GASU serves only to contain these blocks, its function is of limited interest here and we'll focus only on its contents. The AEM portion is very similar to the TEM hierarchy, although it is lacking the Cable Controller functional block. The GEM contains only the Global Trigger (GLT) functional block.

11. Representing the LAT electronics hierarchy in SCL

There will be a separate SCL database for the:

- LAT
- each TEM
- AEM
- GLT

The principle reason for this is that one database cannot describe all the records needed to describe the hardware. It also leads to the advantage that systems can be constructed in which only part of the entire LAT is described. This eliminates the need for the Calorimeter to be concerned about the ACD's database and vice versa. However, unfortunately the hardware design of the TEM doesn't easily allow the Calorimeter database to be kept separate from the Tracker database.

The LAT database contains a GLAT record (see the Appendix) that can be used to descend into the hierarchy by looking up the database ID for:

- a particular TEM in the LAT coordinate system (LAT-TD-00035-01)
- the AEM
- the GLT

These database IDs are required by SCL in order to refer to the records in the database. Note that no code should be written to depend on any particular value of database ID. There is no guarantee that a given database will have the same database ID for each release of the software, even though it may seem like it in test situations.

A GLAST-specific SCL record type describes each node shown in Figure 6. Each of these record types provides the ability to[1]:

- locate its parent
- query the number of its children
- locate a particular child
- address *all* of its children (broadcast operations)
- access its corresponding registers
- invoke its corresponding *dataless* operations (see the Apendix)

The *dataless* operations associated with the functional block are implemented through additional record attributes (also called *fields*, in SCL parlance).

To find the top of the hierarchy in an SCL script, a few constants have been defined. The defined constants are listed in Table 2. They are made available to SCL scripts, rules and constraints by including onlineConstants.scl in each file in which they are to be used.

Global variable	**Description**
`O_LatDbId`	Database ID of the database containing the GLAT record
`O_LatRecId`	Record ID of the GLAT record in the LAT database
`O_TemRecId`	Record ID of the GTEM record in a TEM database
`O_AemRecId`	Record ID of the GAEM record in the AEM database
`O_GltRecId`	Record ID of the GGLT record in the LAT database

Table 2 – Online system constants

11.1. The TEM Record-Type Hierarchy

Table 3 lists the TEM's functional blocks, their number, the corresponding record type and the naming pattern for instances of these records. The quantities shown in angle brackets, e.g. *<CC>*, are two digit indexes that count from zero. Thus, the first GCCC is called `O$GCCC_00`.

TEM Functional Block	**Total Number per TEM**	**Record Type Mnemonic**	**Record instance name**	**Description**
GCCC	4	`GCCC`	`O$GCCC_<CC>`	Calorimeter Cable Controller
GCRC	16 = 4 *x* 4	`GCRC`	`O$GCRC_<CC>_<RC>`	Calorimeter Readout Controller
GCFE	192 = 16 *x* 12	`GCFE`	`O$GCFE_<CC>_<RC>_<FE>`	Calorimeter Front-End ASIC
GTCC	8	`GTCC`	`O$GTCC_<CC>`	Tracker Cable Controller
GTRC	72 = 8 *x* 9	`GTRC`	`O$GTRC_<CC>_<RC>`	Tracker Readout Controller
GTFE	1728 = 72 *x* 24	`GTFE`	`O$GTFE_<CC>_<RC>_<FE>`	Tracker Front-End ASIC
GTIC	1	`GTIC`	`O$GTIC`	Trigger Interface Controller

[1] Note that some of the record types don't have all the listed capabilities as, for example, some parents don't have parents, some children don't have children, etc.

Table 3 – The TEM's functional blocks

Figure 7 depicts the TEM record type hierarchy. Each of the grey boxes represents one or more instances of the custom record types as detailed in the Appendix.

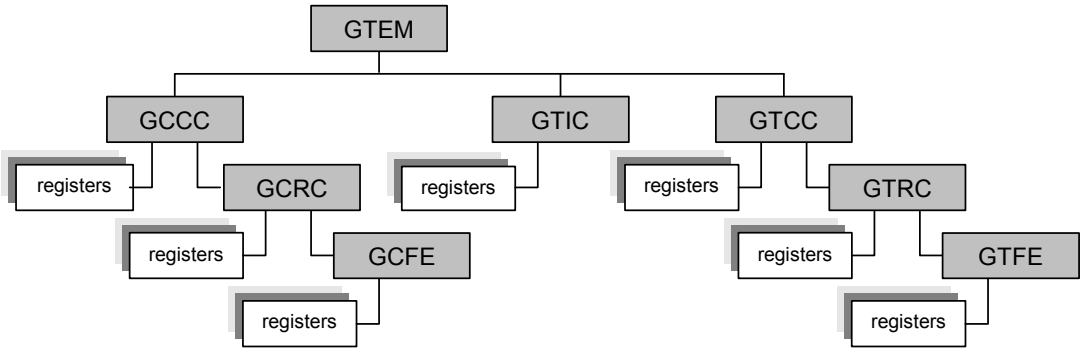

Figure 7 – The TEM record type hierarchy

11.2. The AEM Record-Type Hierarchy

Table 4 lists the AEM's functional blocks, their number, the corresponding record type and the naming pattern for instances of these records. The quantities shown in angle brackets, e.g. *<RC>*, are two digit indexes that count from zero. Thus, the first GARC is called `O$GARC_00`.

AEM Functional Block	Total Number per AEM	Record Type Mnemonic	Record instance name	Description
GARC	12	GARC	O$GARC_*<RC>*	ACD Readout Controller
GAFE	216 = 18 *x* 12	GAFE	O$GAFE_*<RC>*_*<FE>*	ACD Front-End ASIC

Table 4 - The AEM's functional blocks

Figure 8 depicts the AEM record type hierarchy. Each of the grey boxes represents one or more instances of the custom record types as detailed in the Appendix.

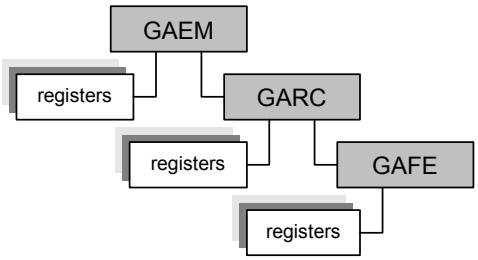

Figure 8 – The AEM record type hierarchy

11.3. The GEM Record-Type Hierarchy

The GEM contains only one functional block, called the GGLT (GLAST Global Trigger). The record type mnemonic for the GGLT is also called GGLT. Normally, only one GGLT record is instantiated in the system and it is called O$GGLT.

Figure 9 shows the GEM record type hierarchy. Each of the grey boxes represents an instance of one of the custom record types. The GGLT is detailed in the Appendix, whereas the GGEM is not yet (July 5, 2002) defined.

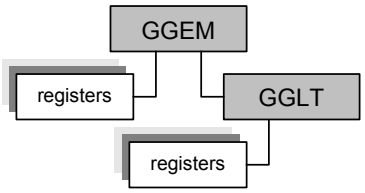

Figure 9 – The GEM record type hierarchy

11.4. Navigating the Record-Type Hierarchies

A useful model for thinking about the SCL representation of the TEM and AEM is a UNIX or Windows file system hierarchy. Records correspond to directories; register and data fields and dataless operation fields correspond to files. When writing an SCL script, one must navigate through the hierarchy to a node of interest in order to operate on one of the features of the destination record. To provide this "cd"-like function, the GLAST-specific records can be linked together via special database fields to allow navigation through a set of functions. These functions, listed with the records in the Appendix, are generally of the form `O$Up<XXX>()`, `O$Down<XXX>()` and `O$All<XXX>()`, where `<XXX>` specifies the record type returned. These functions take at least one argument that is the source record ID, and return the record ID of the destination record. (Earlier drafts of this document discussed passing references to records to and from these functions. This could not be made to work reliably, thus causing the switch to using record IDs.) For those familiar with C++, the first argument is the equivalent of the `this` pointer. An example SCL script showing how to navigate down to the 1st Calorimeter Front End of the 2nd Readout Controller of the 3rd Cable Controller of the 4th TEM to set its `CONFIG_0` register to the value `0xCAFE`, is shown in Figure 10.

```
script O$MyScript
  local  db, tem, fe, tmp           -- Local declarations
  O$GCFE fe                         -- Cast local variable to a GCFE record type

  db  = O$SetDB(O_LatDbId)          -- Switch to database containing GLAT record
  tem = O$DownTEM(O_LatRecId, 3)    -- Look up DB ID of TEM3 in LAT record
  tmp = O$SetDB(tem)                -- Set the current database
  tmp = O$DownCCC(O_TemRecId, 2)    -- Get record ID of GCCC2 record
  tmp = O$DownCRC(tmp, 1)           -- Get record ID of GCRC1 record
  tmp = O$DownCFE(tmp, 0)           -- Get record ID of GCFE0 record
  fe  = O$Id2Ref(tmp)               -- Get reference to GCFE0 record

  fe.CONFIG_0 = 0xCAFE              -- Set the CONFIG_0 register

  db = O$SetDB(db)                  -- Set the current database back
end O$MyScript
```

Figure 10 – TEM navigation SCL script

Modifying the example in Figure 10 to incorporate looping to set the `CONFIG_0` register of *all* the Front Ends attached to TEM_3, CCC_2 and CRC_1 to the value `0xCAFE` is shown in Figure 11.

```
.
.
.
  local  n, i

  n = crc.CFEcnt                  -- Look up number of CFEs in GCRC
  repeat with i = 0 to n – 1      -- Loop over CFEs
    cfe = O$DownCFE(crc, i)       -- Get record ID of GCFE record
    fe  = O$Id2Ref(cfe)           -- Get reference to GCFE record

    fe.CONFIG_0 = 0xCAFE          -- Set CONFIG_0 register

  end repeat
.
.
.
```

Figure 11 – Looping example script portion

Note that the SCL syntax doesn't allow for constructs like:

```
fe = O$Id2Ref(O$DownCFE(crc, i))
```

or compound dereferencing like:

```
value = O$LAT.TEM[3].CCC[2].CRC[1].CFE[0].CONFIG_0
```

One has to step through each record in turn.

11.5. Broadcast Addressing

Broadcasting in the LAT electronics hierarchy can be thought of as a wildcard (e.g., `*`) operation in the file system model. This mechanism is used to address a given field on all instances of a given node in one of the TEM or AEM hierarchies. Broadcasting to all TEMs is not supported. The broadcast operation cannot be used for read operations.

Access to the broadcast mechanism is through the `O$All<XXX>()` function. Similar to the functions introduced above, it takes a source record ID as its only argument (which may be a broadcast record ID itself) and returns the record ID of the broadcast child record type specified by the `<XXX>`. Figure 12 results from recoding the example of Figure 11 using a broadcast. Note the lack of a looping construct.

```
.
.
.
  cfe = O$AllCFE(crc)                -- Get broadcast handle to all GCFEs
  fe  = O$Id2Ref(cfe)                -- Get reference to GCFE record

  fe.CONFIG_0 = 0xCAFE               -- Set CONFIG_0 registers
.
.
.
```

Figure 12 – Broadcast version of Figure 11

A slightly different example of the use of the broadcast operation might be to set the field named `CONFIG_0` of all GCFEs on the 2nd GCRC, for all GCCCs on TEM_3. This would be done as shown in Figure 13.

```
.
.
.
  ccc = O$AllCCC(O_TemRecId)          -- Get broadcast handle to all GCCCs
  crc = O$DownCRC(ccc, 1)             -- Get reference        to     GCRC1
  cfe = O$AllCFE(crc)                 -- Get broadcast handle to all GCFEs
  fe  = O$Id2Ref(cfe)                 -- Get reference to GCFE record

  fe.CONFIG_0 = 0xDEAD                -- Set CONFIG_0 registers
.
.
.
```

Figure 13 – Multiple level broadcast example script

12. Command/Monitoring GUIs

LabVIEW. Discussion of color usage. Discussion of standard action buttons (e.g. close, print, help) in standard locations. TBD. Python/Qt.

13. Data Visualization

Hippo. TBD.

14. Local Database

MS Access. TBD.

15. Persistent Data Storage

FITS files. TBD.

16. Archivers

16.1. Frame

TBD.

16.2. Data

TBD.

16.3. Event

TBD.

17. Message logging

TBD.

18. Test Report Generation

TBD.

19. Problem Reporting

Forums, Remedy. TBS.

20. Release distribution

FTP, ITAR. TBS.

21. Web site

http://www-glast.slac.stanford.edu/LAT/INT/ONLINE. TBS.

22. Appendix

In the following record definitions, fields have been given mnemonics and an order of appearance, the latter of which is important to SCL. The list of fields is broken into three sections for each record:

- record links
- data fields or register representations
- dataless commands
- record operators

The field names and order should still be considered somewhat fluid. The actual record definition files (`$SCL/db/generic/template.sml` and `$SCLPROJECT/db/usertemplate.sml`) should be considered the final arbiter. Note that the SCL project's `usertemplate.sml` file overrides the system wide `template.sml` file.

22.1. The GLAST Large Area Telescope (GLAT) record type

Field	Field Type	Description
TEM	Database pointer	Base TEM database ID
AEM	Database pointer	AEM database ID
GEM	Database pointer	GEM database ID
TEMcnt	unsigned short	Number of TEMs
Dataless operation	**Field Type**	**Description**
CMD_RESET	unsigned byte	Field changes causes RESET command to be sent
Function Name	**Arguments**	**Description**
O$DownTEM	LAT record ID, TEM index	Returns a TEM database ID
O$DownAEM	LAT record ID	Returns the AEM database ID
O$DownGEM	LAT record ID	Returns the GEM database ID

Table 5 – The GLAT record attributes

22.2. The GLAST Tower Electronics Module (GTEM) record type

Field	Field Type	Description
ParentLAT	Record pointer	GLAT record ID
CCC	Record pointer	Base GCCC record ID
TCC	Record pointer	Base GTCC record ID
TIC	Record pointer	GTIC record ID
AllCCC	Record pointer	GCCC broadcast record ID
AllTCC	Record pointer	GTCC broadcast record ID
CCCcnt	unsigned short	Number of GCCCs
TCCcnt	unsigned short	Number of TCCCs
TemID	unsigned short	Physical TEM ID
CONFIGURATION	unsigned long	Configuration and setup
DATA_MASKS	unsigned long	Masks for data taking
STATUS	unsigned long	CSR latched values
COMMAND_RESPONSE	unsigned long	Command/response statistics
CAL_TRGSEQ	unsigned long	Tracker trigger sequencing
TRK_TRGSEQ	unsigned long	Calorimeter trigger sequencing
Dataless operation	**Field Type**	**Description**
CMD_RESET	unsigned byte	Field changes causes RESET command to be sent
Function Name	**Arguments**	**Description**
O$DownCCC	TEM record ID, CCC index	Returns the GCCC record ID given by *index*
O$DownTCC	TEM record ID, TCC index	Returns the GTCC record ID given by *index*
O$DownTIC	TEM record ID	Returns the GTIC record ID
O$AllCCC	TEM record ID	Returns the broadcast GCCC record ID
O$AllTCC	TEM record ID	Returns the broadcast GTCC record ID

Table 6 – The GTEM record attributes

22.3. The TEM Records

22.3.1. Calorimeter Subsystem

22.3.1.1. The GLAST Calorimeter Cable Controller (GCCC) record type

Field	Field Type	Description
ParentTEM	Record pointer	TEM record ID
ChildCRC	Record pointer	Base GCRC record ID
AllCRC	Record pointer	Broadcast GCRC record ID
CRCcnt	unsigned short	Number of GCRCs
TemID	unsigned short	Physical TEM ID
GcccID	unsigned short	Physical GCCC ID
CONFIGURATION	unsigned long	Configuration and setup
LAYER_MASK_0	unsigned long	*Even* layer masks (first pair of GCRCs)
LAYER_MASK_1	unsigned long	*Odd* layer masks (second pair of GCRCs)
FIFO_STATUS	unsigned long	Latched FIFO signals
LATCHED_STATUS	unsigned long	CSR latched values
EVENT_TIMEOUTS	unsigned long	Event timeouts
TRG_ALIGNMENT	unsigned long	Trigger input alignments and delays
Dataless operation	**Field Type**	**Description**
CMD_NOP	unsigned byte	Field change causes NOP command to be sent
CMD_RESET	unsigned byte	Field changes causes RESET command to be sent
Function Name	**Arguments**	**Description**
O$UpTEM	GCCC record ID	Returns the parent TEM record ID
O$DownCRC	GCCC record ID, CRC index	Returns the GCRC record ID given by *index*
O$AllCRC	GCCC record ID	Returns the broadcast GCRC record ID

Table 7 – The GCCC record attributes

22.3.1.2. The GLAST Calorimeter Readout Controller (GCRC) record type

Field	Field Type	Description
ParentCCC	Record pointer	Points to GCCC record
ChildCFE	Record pointer	Points to GCFE records
AllCFE	Record pointer	Points to broadcast GCFE record
CFEcnt	unsigned short	Number of GCFEs
TemID	unsigned short	Physical TEM ID
GcccID	unsigned short	Physical GCCC ID
GcrcID	unsigned short	Physical GCRC ID
CSR	unsigned short	CSR register (status)
LAST_CMD	unsigned short	Last 16 bits of last erring command
DELAY_1	unsigned short	Time Delay (1), Peak hold to ADC acquisition
DELAY_2	unsigned short	Time Delay (2), ADC acquisition time
DELAY_3	unsigned short	Time Delay (3), ADC conversion time
DAC	unsigned short	DAC register
CONFIG	unsigned short	Configuration information (GCRC specific)
Dataless operation	**Field Type**	**Description**
CMD_NOP	unsigned byte	Field change causes NOP command to be sent
CMD_RESET	unsigned byte	Field changes causes RESET command to be sent
CMD_CALSTROBE	unsigned byte	Generate calibration strobe to GCFE
Function Name	**Arguments**	**Description**
O$UpCCC	GCRC record ID	Returns the parent GCCC record ID
O$DownCFE	GCRC record ID, CFE index	Returns the GCFE record ID given by *index*
O$AllCFE	GCRC record ID	Returns the broadcast GCFE record ID

Table 8 – The GCRC record attributes

22.3.1.3. The GLAST Calorimeter Front-End ASIC (GCFE) record type

Field	Field Type	Description
ParentCRC	Record pointer	Points to TEM record
TemID	unsigned short	Physical TEM ID
GcccID	unsigned short	Physical GCCC ID
GcrcID	unsigned short	Physical GCRC ID
GcfeID	unsigned short	Physical GCFE ID
CONFIG_0	unsigned short	Configuration register 0
CONFIG_1	unsigned short	Configuration register 1
FLE_DAC	unsigned short	Low energy trigger discriminator
FHE_DAC	unsigned short	High energy trigger discriminator
LOG_ACPT	unsigned short	Log accept discriminator
RNG_ULD_DAC	unsigned short	Range select discriminator
REF_DAC	unsigned short	DAC for DC reference
Dataless operation	**Field Type**	**Description**
CMD_NOP	unsigned byte	Field change causes NOP command to be sent
Function Name	**Arguments**	**Description**
O$UpCRC	GCFE record ID	Returns the parent GCRC record ID

Table 9 – The GCFE record attributes

22.3.2. Tracker Subsystem

22.3.2.1. The GLAST Tracker Cable Controller (GTCC) record type

Field	Field Type	Description
ParentTEM	Record pointer	Points to TEM record
ChildTRC	Record pointer	Points to GTRC records
AllTRC	Record pointer	Points to broadcast GTRC record
TRCcnt	unsigned short	Number of GTRCs
TemID	unsigned short	Physical TEM ID
GtccID	unsigned short	Physical GTCC ID
CONFIGURATION	unsigned long	Configuration and setup
INPUT_MASK	unsigned long	Input masking
FIFO_STATUS	unsigned long	Latched FIFO signals
LATCHED_STATUS	unsigned long	CSR latched values
EVENT_TIMEOUTS	unsigned long	Event timeouts
TRG_ALIGNMENT	unsigned long	Trigger input alignments and delays
Dataless operation	**Field Type**	**Description**
CMD_NOP	unsigned byte	Field change causes NOP command to be sent
CMD_RESET	unsigned byte	Field changes causes RESET command to be sent
Function Name	**Arguments**	**Description**
O$UpTEM	GTCC record ID	Returns the parent TEM record ID
O$DownTRC	GTCC record ID, TRC index	Returns the GTRC record ID given by *index*
O$AllTRC	GTCC record ID	Returns the broadcast GTRC record ID

Table 10 – The GTCC record attributes

22.3.2.2. The GLAST Tracker Readout Controller (GTRC) record type

Table 11 shows the record definition for the GTRC record type. It should be noted that the CSR_MSW, CSR_LSW pair refer to the Most Significant, and Least Significant 32 bit Words of a physical 64 bit register, respectively. Because SCL doesn't support a 64 bit primitive, we're forced to break the register into two pieces.

Read operations read the entire 64 bit register but return only the longword for which the read was requested (nb, this wouldn't work correctly if the register read had side effects). Write operations are implemented with an *interlocked* read-modify-write operation; the 64 bit register is read, the longword that triggered the operation is replaced in the result and then the combined result is written. "Interlocked" means with respect to other SCL operations that access the register via the SCL database methods (e.g., scripts and pktgen/TelemetryServer), but not with respect to VxWorks processes that use low level routines to access registers.

This mechanism clearly doesn't work for broadcast write operations. The solution to this problem requires that the broadcast write be done in two steps: the first writes the MSW value to an interim location (no operation is carried out on the hardware), the second combines this interim MSW value with the supplied LSW value and broadcast writes the hardware.

Field	Field Type	Description
ParentTCC	Record pointer	Points to TEM record
ChildTFE	Record pointer	Points to GCRC records
AllTFE	Record pointer	Points to broadcast GTFE record
TFEcnt	unsigned short	Number of GTFEs
TemID	unsigned short	Physical TEM ID
GtccID	unsigned short	Physical GTCC ID
GtrcID	unsigned short	Physical GTRC ID
CSR_MSW	unsigned long	CSR register (status), MSW of pair
CSR_LSW	unsigned long	CSR register (status), LSW of pair
SYNCH	unsigned long	Synch register
Dataless operation	**Field Type**	**Description**
CMD_NOP	unsigned byte	Field change causes NOP command to be sent
CMD_RESET	unsigned byte	Field changes causes RESET command to be sent
Function Name	**Arguments**	**Description**
O$UpTCC	GTRC record ID	Returns the parent GTCC record ID
O$DownTFE	GTRC record ID, TFE index	Returns the GTFE record ID given by *index*
O$AllTFE	GTRC record ID	Returns the broadcast GTFE record ID

Table 11 – The GTRC record attributes

22.3.2.3. The GLAST Tracker Front-End (GTFE) record type

See comments for the GTRC record type when dealing with the DATA_MASK, CALIB_MASK and TRIG_MASK 64 bit registers. These registers are handled similarly to its CSR register.

Field	Field Type	Description
ParentTRC	Record pointer	Points to GTRC record
TemID	unsigned short	Physical TEM ID
GtccID	unsigned short	Physical GTCC ID
GtrcID	unsigned short	Physical GTRC ID
GtfeID	unsigned short	Physical GTFE ID
DATA_MASK_MSW	unsigned long	Channel mask (data), MSW of pair
DATA_MASK_LSW	unsigned long	Channel mask (data), LSW of pair
CALIB_MASK_MSW	unsigned long	Channel mask (calibration), MSW of pair
CALIB_MASK_LSW	unsigned long	Channel mask (calibration), LSW of pair
TRIG_MASK_MSW	unsigned long	Channel mask (trigger), MSW of pair
TRIG_MASK_LSW	unsigned long	Channel mask (trigger), LSW of pair
DAC	unsigned short	Threshold/Calibration DACs
MODE	unsigned byte	Mode register
Dataless operation	**Field Type**	**Description**
CMD_NOP	unsigned byte	Field change causes NOP command to be sent
CMD_RESET	unsigned byte	Field changes causes hard reset command to be sent
CMD_CALSTROBE	unsigned byte	Generate calibration strobe
CMD_READEVENT	unsigned byte	Begin event readout sequence
Function Name	**Arguments**	**Description**
O$UpTRC	GTFE record ID	Returns the parent GTRC record ID

Table 12 – The GTFE record attributes

22.3.3. The GLAST Trigger Interface Controller (GTIC) record type

Field	Field Type	Description
ParentTEM	Record pointer	Points to GTEM record
TemID	unsigned short	Physical TEM ID
POWER_SUPPLY	unsigned long	Tracker & Calorimeter Power Supply control
STATUS	unsigned long	TBD
CAL_INPUT_MASK	unsigned long	Masking for *Calorimeter* based inputs
CAL_LRS_MASK	unsigned long	LRS masking for *Calorimeter* based inputs
CAL_LRS_COUNTERS	unsigned long	LRS counters for *Calorimeter* based inputs
TRK_INPUT_MASK_0	unsigned long	Masking for *Tracker* based inputs
TRK_INPUT_MASK_1	unsigned long	Masking for *Tracker* based inputs
TRK_INPUT_MASK-2	unsigned long	Masking for *Tracker* based inputs
TRK_LRS_MASK_0	unsigned long	LRS masking for *Tracker* based inputs
TRK_LRS_MASK-1	unsigned long	LRS masking for *Tracker* based inputs
TRK_LRS_MASK_2	unsigned long	LRS masking for *Tracker* based inputs
TRK_LRS_COUNTERS	unsigned long	LRS counters for *Tracker* based inputs
BUSY_LRS_MASK	unsigned long	LRS masking for *Deadtime* based inputs
BUSY_LRS_COUNTER	unsigned long	LRS counters for *Deadtime* based inputs
ADC_CONFIG	unsigned long	ADC configuration and setup
ADC_SELECTION	unsigned long	ADC multiplexer selection and value
HVDAC_TKR	unsigned long	High Voltage DAC for Tracker
HVDAC_CAL	unsigned long	High Voltage DAC for Calorimeter
Dataless operation	**Field Type**	**Description**
CMD_NOP	unsigned byte	Field change causes NOP command to be sent
CMD_RESET	unsigned byte	Field changes causes hard reset command to be sent
Function Name	**Arguments**	**Description**
-	-	-

Table 13 – The GTIC record attributes

22.4. The GLAST ACD Electronics Module (GAEM) record type

Field	Field Type	Description
ParentLAT	Record pointer	GLAT record ID
ARC	Record pointer	Base GARC record ID
AllARC	Record pointer	GARC broadcast record ID
ARCcnt	unsigned short	Number of GARCs
CONFIGURATION	unsigned long	Configuration and setup
COMMON_STATUS	unsigned long	CSR latched values for (AEM)
CABLE_STATUS	unsigned long	CSR latched cable values
COMMAND_RESPONSE	unsigned long	Command/response statistics
TRGSEQ	unsigned long	Trigger sequencing
Dataless operation	**Field Type**	**Description**
CMD_RESET	unsigned byte	Field changes cause a hard RESET of the AEM
Function Name	**Arguments**	**Description**
O$DownARC	ARC index	Returns the GARC record ID given by *index*
O$AllARC	-	Returns the broadcast GARC record ID

Table 14 – The GAEM record attributes

22.5. The AEM records

22.5.1. The GLAST ACD Readout Controller (GARC) record type

The register fields shown between the triple line boundaries in Table 15 are *functional blocks* in the GARC. These functional blocks exist primarily for the electronics design of the GARC and are irrelevant as far as the Online software is concerned.

Field	Field Type	Description
ParentAEM	Record pointer	GAEM record ID
ChildAFE	Record pointer	Base GAFE record ID
AllAFE	Record pointer	GAFE broadcast record ID
AFEcnt	unsigned short	Number of GAFEs
GarcID	unsigned short	Physical GARC ID
VETO_DELAY	unsigned short	Delay from DISC in to NVETO out
RQST_HVBS	unsigned short	Requested High Voltage for normal operation
RQST_HVSAA	unsigned short	Requested High Voltage for SSA operation
HVBS	unsigned short	Current High Voltage for normal operation (readonly)
HVSAA	unsigned short	Current High Voltage for SSA operation (readonly)
HOLD_DELAY	unsigned short	Delay from trigger to hold
VETO_WIDTH	unsigned short	Pulse width of NVETO
HITMAP_WIDTH	unsigned short	Minimum pulse width of hitmap signals
HITMAP_DEADTIME	unsigned short	Time added to hitmap signals
LOOK_AT_ME	unsigned short	A or B only
HITMAP_DELAY	unsigned short	Delay from DISC to HITMAP signals
PHA_0	unsigned short	PHA readout enables for channels 0-15
VETO_0	unsigned short	Veto enables for channels 0-15
HLD_0	unsigned short	HLD enables for channels 0-15
PHA_1	unsigned short	PHA readout enables for channels 16-17 in LSB
VETO_1	unsigned short	Veto enables for channels 16-17 in LSB
HLD_1	unsigned short	HLD enables for channels 16-17 in LSB
MAX_PHA	unsigned short	Maximum allowable number of PHA values
MODE	unsigned short	Various bit fields for mode settings
STATUS	unsigned short	Status (readonly)
LAST_CMND	unsigned short	Command or data from last command error (readonly)
DIAGNOSTIC	unsigned short	? (readonly)
CMD_REJECT	unsigned short	Number of commands rejected (readonly)

FREE_ID	unsigned short	FREE board ID (readonly)
GARC_VERSION	unsigned short	GARC version number (readonly)
PHA_THRESHOLD_0	unsigned short	PHA threshold for channel 0
PHA_THRESHOLD_1	unsigned short	PHA threshold for channel 1
PHA_THRESHOLD_2	unsigned short	PHA threshold for channel 2
PHA_THRESHOLD_3	unsigned short	PHA threshold for channel 3
PHA_THRESHOLD_4	unsigned short	PHA threshold for channel 4
PHA_THRESHOLD_5	unsigned short	PHA threshold for channel 5
PHA_THRESHOLD_6	unsigned short	PHA threshold for channel 6
PHA_THRESHOLD_7	unsigned short	PHA threshold for channel 7
PHA_THRESHOLD_8	unsigned short	PHA threshold for channel 8
PHA_THRESHOLD_9	unsigned short	PHA threshold for channel 9
PHA_THRESHOLD_10	unsigned short	PHA threshold for channel 10
PHA_THRESHOLD_11	unsigned short	PHA threshold for channel 11
PHA_THRESHOLD_12	unsigned short	PHA threshold for channel 12
PHA_THRESHOLD_13	unsigned short	PHA threshold for channel 13
PHA_THRESHOLD_14	unsigned short	PHA threshold for channel 14
PHA_THRESHOLD_15	unsigned short	PHA threshold for channel 15
PHA_THRESHOLD_16	unsigned short	PHA threshold for channel 16
PHA_THRESHOLD_17	unsigned short	PHA threshold for channel 17
ADC_TACQ	unsigned short	ACD acquisition time
Dataless operation	**Field Type**	**Description**
CMD_RESET	unsigned byte	Field changes cause a hard RESET of the GARC
CMD_CALSTROBE	unsigned byte	Generate calibration strobe
CMD_SET_HVBS	unsigned byte	Set HV to normal vale
CMD_SET_HVSSA	unsigned byte	Set HV to SSA value
Function Name	**Arguments**	**Description**
O$UpAEM	ARC index	Returns the GAEM record ID
O$DownAFE	ARC index	Returns the GAFE record ID given by *index*
O$AllAFE	-	Returns the broadcast GAFE record ID

Table 15 – The GARC record attributes

22.5.2. The GLAST ACD Front End (GAFE) record type

Field	Field Type	Description
ParentARC	Record pointer	Points to GARC record
GarcID	unsigned short	Physical GARC ID
GafeID	unsigned short	Physical GAFE ID
CONFIGURATION	unsigned short	Configuration and setup
VETO_DAC	unsigned short	Veto DAC value
HLD_DAC	unsigned short	?
LLD_DAC	unsigned short	?
BIAS_DAC	unsigned short	Bias value
TCI_DAC	unsigned short	Test charge inject value
VERS_ADDR	unsigned short	Version number (read-only)
WRITE_CTR	unsigned short	Number of load commands since reset (read-only)
REJECT_CTR	unsigned short	Number of commands rejected since reset (read-only)
LOOP_CTR	unsigned short	Number of commands since reset (read-only)
CHIP_ADDR	unsigned short	GAFE chip address (read-only)
Dataless operation	**Field Type**	**Description**
-	-	-
Function Name	**Arguments**	**Description**
O$UpARC	GAFE record ID	Returns the parent GARC record ID

Table 16 – The GAFE record attributes

22.6. The GEM Records

22.6.1. The GLAST GLobal Trigger (GGLT) record type

Field	Field Type	Description
CONFIGURATION	unsigned long bitfield	Configuration and setup
PARITY	offset 0, length 1	1 = even parity, 0 = odd parity
TAG	offset 1, length 2	Initial event tag
EVENT_NUMBER	offset 3, length 15	Initial event number
MASK	Unsigned long bitfield	Trigger enable/disable mask (0 = all enabled)
INTERNAL_TRG	offset 0, length 1	Internal trigger enable/disable (0 = enabled)
EXTERNAL_TRG	offset 1, length 1	External trigger enable/disable (0 = enabled)
CAL_LOW	offset 2, length 1	CAL low trigger enable (0 = enabled)
CAL_HIGH	offset 3, length 1	CAL high trigger enable (0 = enabled)
THREE_IN_A_ROW	offset 4, length 1	Three-in-a-row trigger enable (0 = enabled)
THROTTLE_DISABLE	offset 5, length 1	Throttle enable/disable (0 = enabled)
OPTIONS	Unsigned long bitfield	Dynamic options
DESTINATION	offset 0, length 6	Destination field
MARKER	offset 6, length 3	Marker field
ZERO_SUPPRESS	offset 9, length 1	Zero suppression enable/disable
FOUR_RANGE_READOUT	offset 10, length 1	Four-range-readout enable/disable
TACK	offset 11, length 1	Trigger acknowledge enable/disable
CAL_STROBE	offset 12, length 1	CAL strobe
Dataless operation	**Field Type**	**Description**
CMD_RESET	unsigned byte	Field changes cause a RESET of the GGLT
CMD_SELF_TRIGGER	unsigned byte	

Table 17 – The GGLT record attributes