Stanford Linear Accelerator Center
2575 Sandhill Road
Menlo Park California 94025

ACD Test-stand architecture

Document Version: LAT-TD-xxx-02
Document Date: 27 February, 2003
Document Status: First draft to support system distribution
Document Author: Michael Huffer

Abstract

This memo describes both current and proposed test-stand hardware/software designed and fabricated at SLAC, used to assist in debugging, commissioning, and testing ACD Front-End Electronics. This system is commonly referred to as a "ACD test-stand". This memo defines and describes three *generations* of such test-stands:

G1 The architecture currently deployed

G2 Supports the ICD described in [1]

G3 Based on the GASU and LCB

SLAC supports only *one* generation at a time. Note that historically, responsibility for design and fabrication of any test-stand is divided at the "wire" between the embedded and host environments. The Electronics group at SLAC assumes responsibility *upstream* of, and including the embedded system. The I & T group assumes responsibility *downstream* of the embedded system. Coordination and maintenance of the entire system is shared between both Electronics and I & T groups. This particular memo address only those architectural issues upstream of, and including the embedded system. Further, the proposals expressed in this memo supersede those parts of [3] which pertain to both the ACD and its related AEM.

1 References

1 "ACD-LAT Interface Control Document (ICD) - Mechanical, Thermal and Electrical", LAT-SS-00363-04, by *M. Amoto, et al*

2 "LAT Comm I/O Board - Triggering", LAT-TD-00597-03, by *Curt Brune*

3 “Test-stand architecture redux”, Version 1.1 LAT-TD-00861-01, by *Michael Huffer*

4 “The LAT Communications Board design specification”, Version 2.1 LAT-TD-860-01, by *Michael Huffer*

5 “The ACD Electronics Module - A Primer”, Version 2.1, LAT-TD-00639-D1, by *Michael Huffer*

6 “The GLAST Global Trigger design specification”, Version xxx, LAT-TD-xxxx-01[1], by *Michael Huffer*

7 “The GLAST Event Builder design specification”, Version xxx, LAT-TD-xxxx-01[1], by *Michael Huffer*

8 “The GLAST Fan-In/Fan-Out Unit design specification”, Version xxx, LAT-TD-xxxx-01[1], by *Michael Huffer*

9 “LATTest Stand Communications Interface - Software User’s Guide”, LAT-TD-01199-01, by *Curt Brune*

2 G1

G1 is the generation of test-stand currently deployed. It was designed to support a limited emulation of the ACD Front-End Electronics. This limited emulation was defined and provided by the ACD group at Goddard. It consists of the following components:

One “bud box”: The so-called “bud-box” is the initial hardware provided by the ACD group to SLAC. It includes one FPGA containing the VHDL for the GARC, a low fidelity emulation of two GAFEs, an external interface, all contained in a single aluminium box.

One comm board: Working in conjunction with the processor board described below, this board provides an emulation of the AEM[2] (called the “pseudo AEM”) and a rudimentary trigger system (called the “mini-GLT”).

One processor board: Working in conjunction with the comm board described above, this board provides an emulation of the AEM[2] (called the “pseudo AEM”) and a rudimentary trigger system (called the “mini-GLT”). It also provides an interface to import command and export data to the downstream components of the test-stand. Currently, the processor is a *Motorola* MVME 2306.

One VME crate: Used to contain two cards: a *Motorola* MVME 2306 and a Comm board, both as described above.

1. Cyper-docs number yet to be assigned
2. As described in [5].

Finally, of course, all the cabling to interface these components together is necessary. A diagram which shows these components, cables, and connections is illustrated by Figure 1.

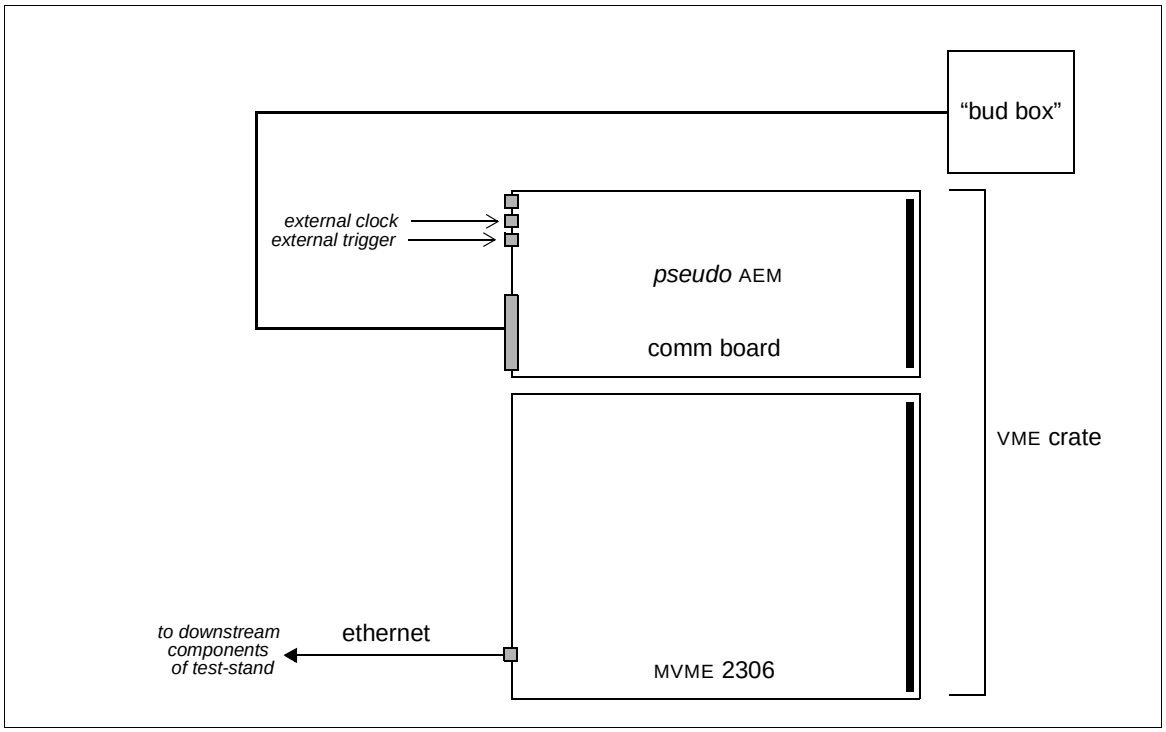

Figure 1 G1 test-stand arraignment

2.1 Pseudo AEM

The "pseudo" AEM is a limited fidelity emulation of the AEM interface described in [5]. It is designed to support at most only *one* FREE board worth of ACD Front-End Electronics. The emulation assumes the FREE board address is fixed as *zero* in the system. It implements the register model described in [5] and under the stimulus of the mini-GLT (described below) produces event data in the format specified in [5]. The fidelity of the simulation is defined in [9].

2.2 Mini-GLT

The mini-GLT is a rudimentary trigger system designed to fill the gap between deployment of the test-stand, where some form of triggering was necessary to read-out event data and the design and fabrication of the "real" trigger system (the GLT, contained within the GASU). Of necessity, it bears little, or no resemblance in either functionality or interface to the GLT. In

addition, as it was designed to present a common interface to I & T, independent of sub-system usage, many of its functions are superfluous in operating the ACD test-stand. It appears to its FSW interface as a set of three VME registers described in [2] and [9].

3 G2

G2 has all the functionality of G1, however, rather then supporting the “bud-box” interface, it supports the ICD described in [1]. The upstream implementation of this interface may be satisfied in a variety of fashions, with the test-stand being impervious to any difference in implementation. In order to illustrate the division of responsibilities, two potential upstream implementations of the ICD are illustrated below; the first, using the GARC / GAFE test-board combination and the second, using a FREE card.

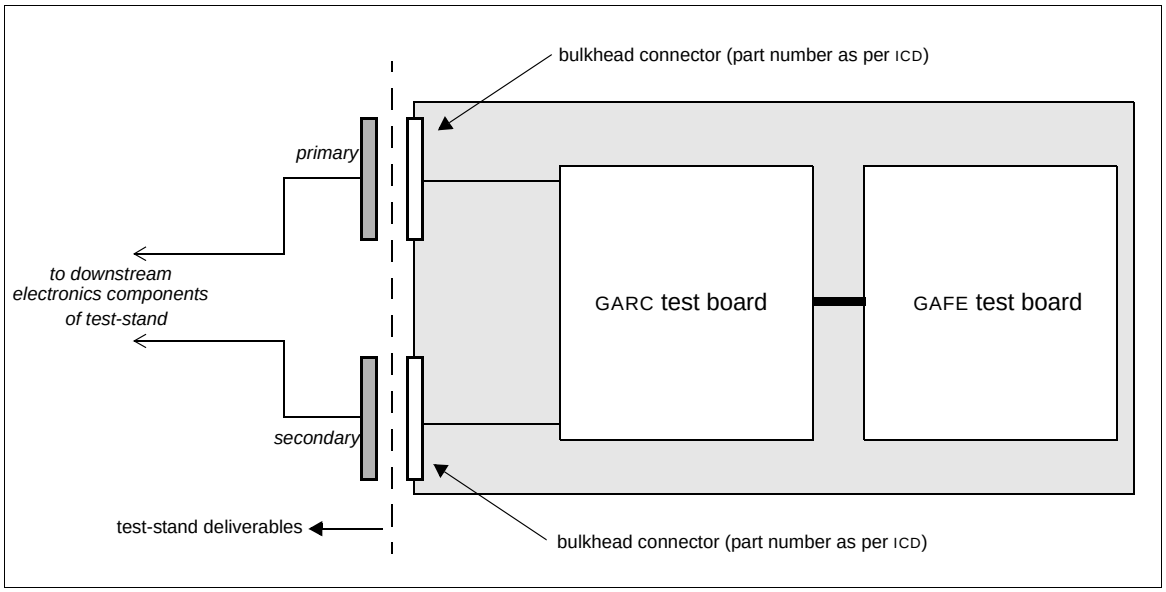

Figure 2 G2 interface using GARC/GAFE test boards

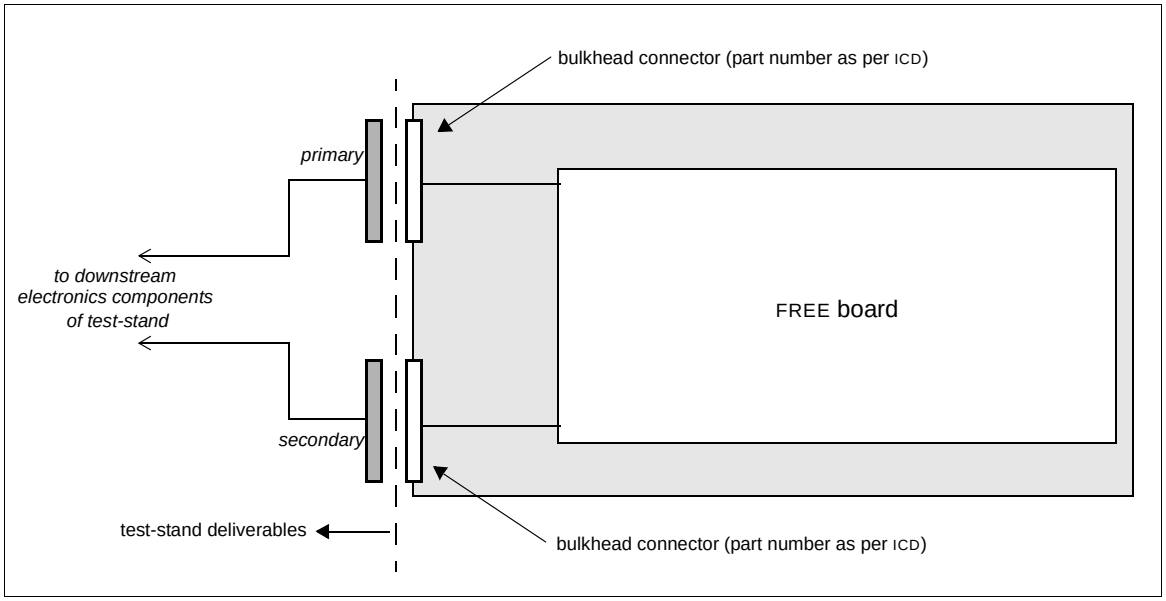

Figure 3 G2 interface using a FREE board

In order to support the functionality required of the ICD Generation *2* consists of the following components:

LAT supply: This component converts wall-power to 28 Volts, which will be then used to satisfy the ICD requirements. The realization of this component is a commercial, off the shelf, bench power supply (the same bench supply used by other sub-systems in their test-stands).

One comm board: The comm board's function is no different then the comm board of Generation *1*, that is, to provide (along with the processor) an emulation of the AEM and a rudimentary trigger system. However, the internal VHDL and external interface will of a necessity be different from G1. The user may test redundancy by connecting the cable between comm board and ACD electronics to either the electronic's *Primary* or *Redundant* connectors. No changes to either the comm board or systems software is necessary.

One processor board: Working in conjunction with the comm board described above this board provides an emulation of up to two AEMs and mini-GLTs. It continues to provide the same command and event interface of G1.

Finally, of course, all the cabling to interface these components is provided. A diagram which shows these components, cables, and connections is illustrated in Figure 4.

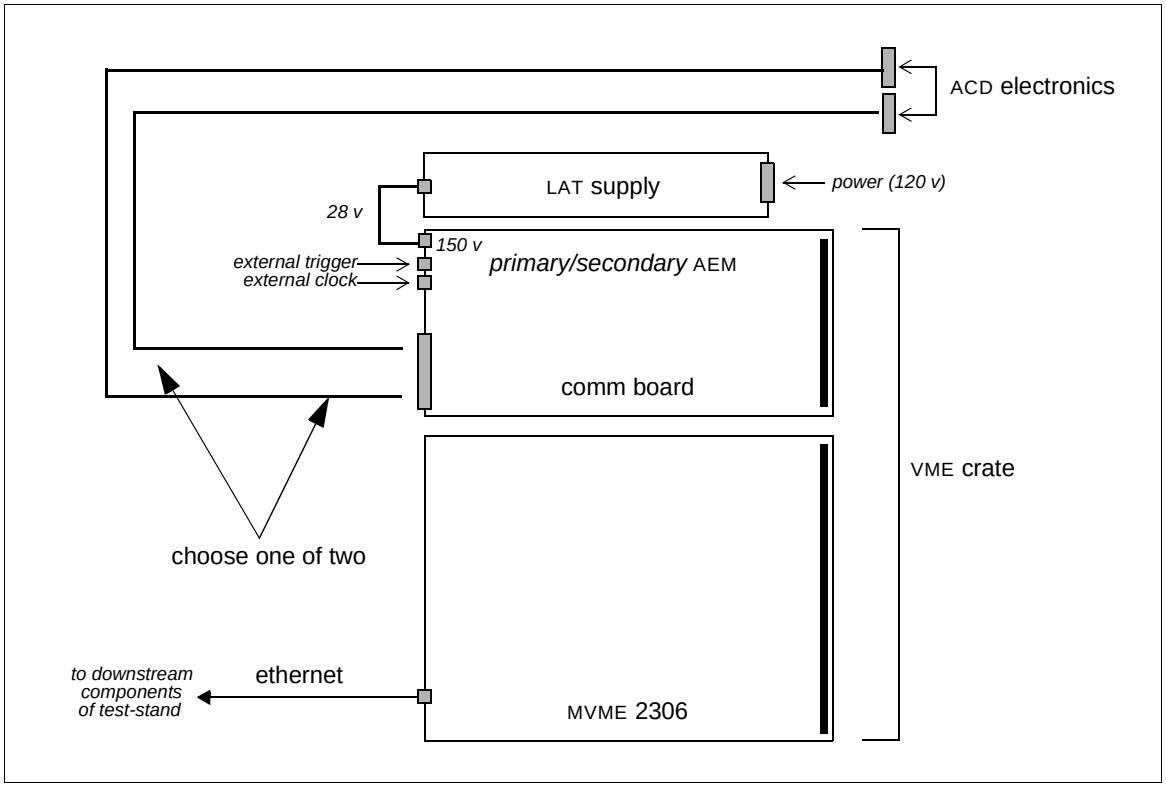

Figure 4 G2 test-stand arraignment

3.1 New Features

Generation *2* contains all the features of Generation *1* plus:

- Interface to agreed on definition
- Support redundancy and fail-over
- Higher fidelity of simulation of on-board registers of AEM
- Supplies all power, including V_{DD} (3.3. V) and bias (28 V)[1]
- Processes both VETO and CNO signals, allowing them to be used to trigger system
- Allow monitoring of VETO or CNO signal rates

1. The quality of this supplies with respect to ACD requirements remains to be tested

3.2 Restrictions

In comparison to the functionality represented by the flight Data Acquisition System, this test-stand has the following (identified) restrictions:

- does not provide either an accurate or complete representation of the trigger system
- AEM emulation supports only one FREE board worth of electronics[1]
- AEM emulation does not allow:
  - Programmatic control over turning on / off power to FREE card
  - Monitoring of FREE card temperature
  - Monitoring of FREE card V_{DD}
  - Monitoring of High Voltage
- maximum event rate is approximately 1 KHZ
- does not allow margin control over V_{DD}

Although I don't believe these affect the interface, for completeness the wires of the ICD which are *not* brought to the comm boards are as follows:

- ACD_VDD_1
- ACD_VDD_2
- ACD_28V_1x
- ACD_HV1_xP, ACD_HVI_xM
- ACD_HV2_xP, ACD_HV2_xM
- ACD_TEMP_xP, ACD_TEMP_xM

3.3 Pseudo AEM

Their are no functional differences between the *off*-board behaviour of the pseudo AEM of Generation *1* and of Generation *2*. The fidelity of the emulation of the *on*-board registers of the AEM is described below. Unless noted, the definition of "*Not implemented*" is as follows:

Register field: The initial value of the field is zero. The field may be either written to, or read from, however, the functionality behind the field is not emulated. Reading the field will simply return either the last written value, or zero.

Register: The initial value of the register is zero. The field may be either written to, or read from, however, the functionality behind the register is not emulated. Reading the register will simply return either the last written value, or zero.

1. Assumed to reside at address *zero* (0)

Function block: All registers of the block have the behaviour of an "*not implemented*" register as described above.

3.3.1 Configuration register

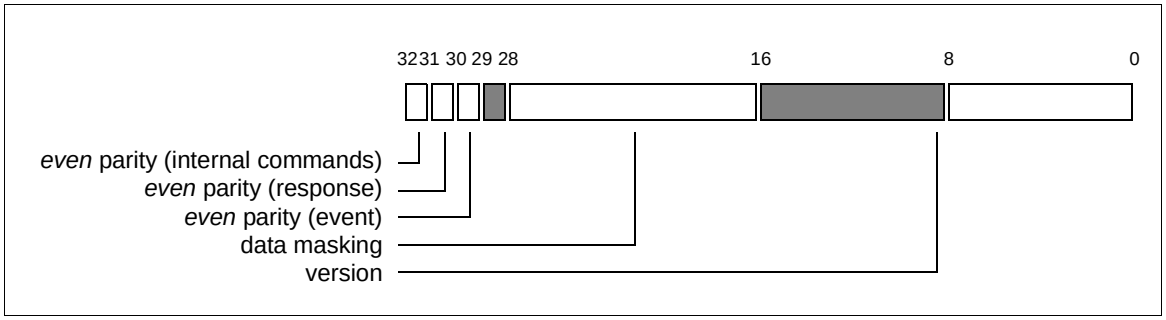

Figure 5 AEM Configuration register

even parity (internal commands): Implemented[1].

even parity (response): Not implemented.

even parity (event): Not implemented.

data masking: Implemented. However, only the LSB (corresponding to $FREE_0$) will have any affect.

version: ***Type*** and ***Board Revision*** fields will have a value of *zero*. The VHDL revision field will be defined by Curt to follow software revisions of the emulation.

3.3.2 Address register

Implemented.

3.3.3 Timeout register

Not implemented.

1. Implementation note: changes in this field drive changes in the "even trigger message parity" field of the register described in Section 3.4.9.

3.3.4 Common status

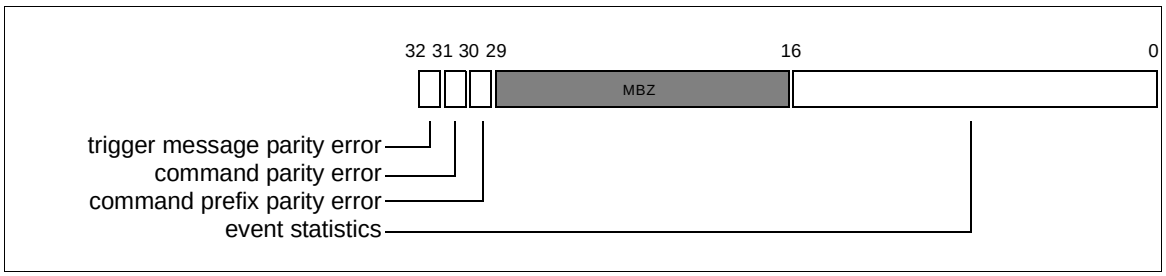

Figure 6 AEM Status register

trigger message parity error: Implemented, however will always have a value of *zero*.

command parity error: Implemented, however will always have a value of *zero*.

command prefix parity error: Implemented, however will always have a value of *zero*.

event statistics: Implemented.

3.3.5 Cable status

Implemented, however the register will always read *zero*.

3.3.6 Command/Response Statistics register

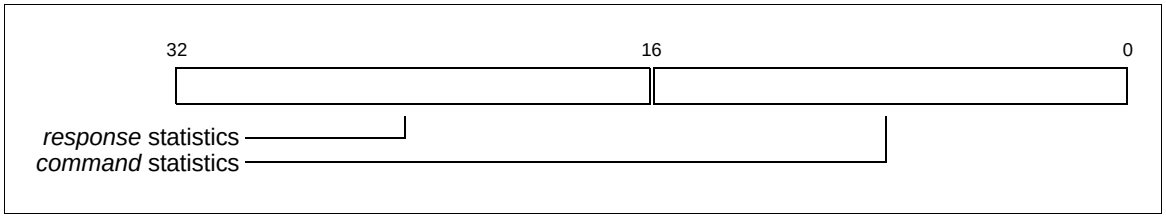

Figure 7 AEM Command/Response statistics register

response statistics: Implemented.

command statistics: Implemented.

3.3.7 Trigger sequencing register

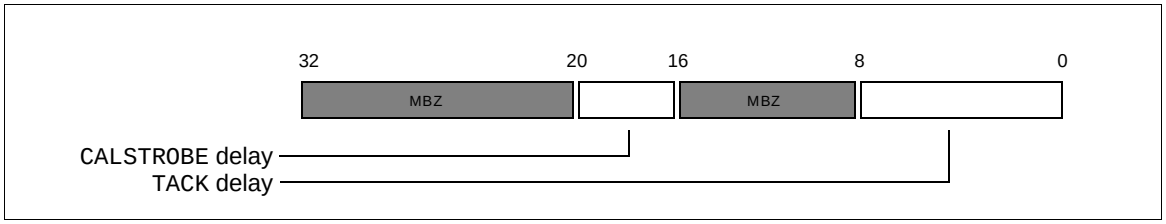

Figure 8 AEM trigger sequencing register

CALSTROBE delay: Not implemented.

TACK delay: Implemented.

3.3.8 Power management

Not implemented.

3.3.9 Environmental monitoring

Not implemented.

3.4 Mini GLT

While not strictly necessary to change the mini-GLT interface to support the ICD, G2 provides a convenient point to implement additional functionality requested by the ACD group. This in turn, unfortunately requires a change in the interface. As these changes are somewhat substantial, I define the entire new interface rather then list the differences between the old and the new[1].

1. Implementation note: This interface is identical to the mini-GLT interface currently defined for the transition board with the following exceptions:

   a. The addition of the two registers described in sections 3.4.7 and 3.4.8

   b. extending the register described in Section 3.4.6 to include the vetoes and CNO

Any other differences are as indicated. Question: Should we extend these changes to the transition board?

3.4.1 External trigger electrical interface

Any external trigger signal is brought into the front panel of the comm board to the 2-pin LIMO connector labelled TST0. This signal must follow TTL level conventions. The interface expects a 1 Volt pulse with a minimum width of two clock cycles (100 nanoseconds with the default 20 MHZ clock). A *positive* transition of this pulse constitutes a single trigger.

3.4.2 The initial sequence register

The message encoded and transmitted by the mini-GLT and received by a source is called a “Trigger Message”. For each message sent, the mini-GLT tags the message with a sequence number. Once set, the mini-GLT will “increment” this sequence number. This register determines the *initial* value of the sequence number and is illustrated in Figure 9: The *current* sequence number is discussed in see Section 3.4.3

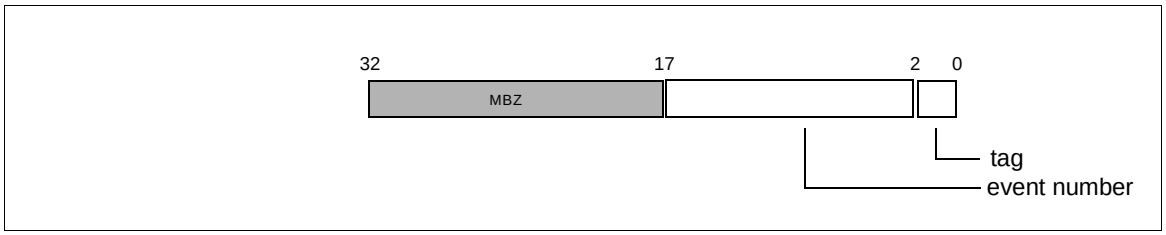

Figure 9 The initial sequence register

tag: The initial two *least* significant bits of the 17-bit event sequence number. The remaining 15-bits of the sequence number are contained in the ***event number*** field.

event number: The initial 15 most significant bits of the 17-bit event sequence number. The low-order two bits of the sequence number are contained in the ***tag*** field.

3.4.3 The current sequence register

The message encoded and transmitted by the mini-GLT and received by a source is called a “Trigger Message”. For each message sent, the mini-GLT tags the message with a sequence number. Once set (see Section 3.4.2), the mini-GLT will “increment” this sequence number. This register is *read-only* and returns the current value of the sequence number as illustrated in Figure 10:

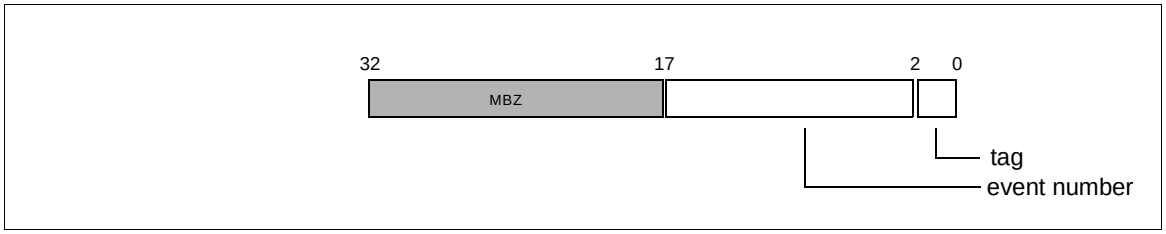

Figure 10 The current sequence register

tag: The current two *least* significant bits of the 17-bit event sequence number. The remaining 15-bits of the sequence number are contained in the ***event number*** field.

event number: The current 15 most significant bits of the 17-bit event sequence number. The low-order two bits of the sequence number are contained in the ***tag*** field.

3.4.4 Trigger message register

The message encoded and transmitted by the mini-GLT and received by a source is called a "Trigger Message". The mini-GLT allows the user program control over the contents of a message when sent. This control is expressed with the register illustrated in Figure 11:

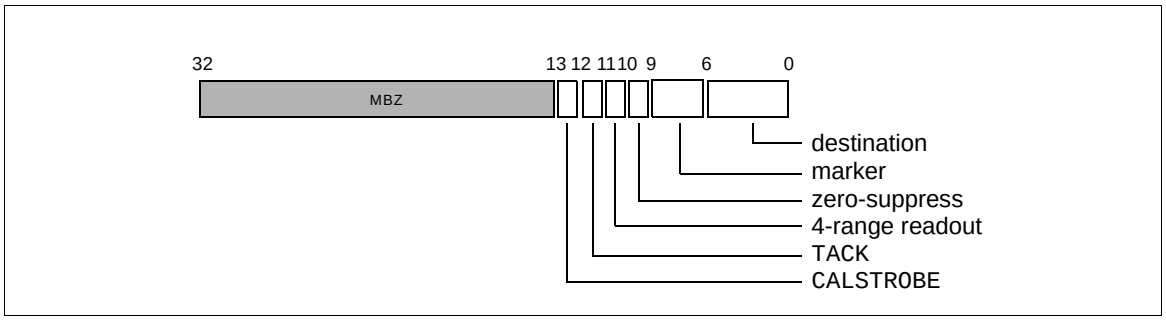

Figure 11 The trigger message register

destination: The GLT in conjunction with its nodes and the Event Builder determines the destination address of event data triggered by the mini-GLT. The destination is one of the possible thirty-two CPUs occupying a slot in either an EPU or SIU crate (see [1]). For any given trigger message the destination of its corresponding event is determined by the value of this field. Table 2 enumerates how different values of this field map to possible event destinations.

marker: User defined. This information is simply transmitted by the mini-GLT and by convention is reflected by each node in their event contribution. For example, flight software will use this field in order to insert "markers" at well-known times into the event stream.

zero suppress: Determines (for those nodes on the trigger fabric which are capable of performing zero suppression[1]) whether or not their event data should be zero suppressed. If this field is *set*, zero suppression is performed. If this field is *clear*, event data is not zero suppressed.

4-range readout: Determines (for those nodes on the trigger fabric which have this capability[2]) whether or not their event data should include all four ranges from its digitizers. If this field is *set*, the node will emit all four ranges of its event data. If *clear*, the node will auto-range and send the one appropriate range for its event data.

TACK: Specifies whether a *second* command should be issued by a node to its Front-End Electronics. If this field is true (*set*), a second command is emitted (following the first command specified in the CALSTROBE field described below). This second command is always a TACK. If the field is false (*clear*), a second command is *not* emitted.

CALSTROBE: Specifies what type of command should *first* be issued by a node to its Front-End Electronics. If this field is true (*set*), the first command emitted is a CALSTROBE. If the field is false (*clear*), the first command emitted is a TACK (see the TACK field described above).

Table 1 Using the destination field of the trigger message register

Field value[a]	Effect on event destination
0	Defers decision to Event Builder (see [7])
1 - 1E	Sent to *one* (the specified) address
1F	Sent to *all* addresses

a. Hexadecimal

3.4.5 Solicit trigger register

Writing to this register will generate a transition on the *solicited* trigger line. If this line is *not* masked (see Figure 12) and the system is *not* busy, a trigger message will be transmitted. The value written to this register is ignored. Reading this register will return unpredictable results.

1. This includes, for example, the calorimeter data from a tower and event data from the ACD.
2. For example, the calorimeter fraction of a tower.

3.4.6 Trigger source mask register

The mini-GLT process transitions on twenty-one different trigger lines. These lines correspond to:

- The solicited trigger
- The external trigger (TST0)
- The 18 Veto signals from the 18 GAFEs on a FREE board
- The CNO signal from a FREE board

Each one of these lines can be masked off from trigger consideration using the register illustrated in Figure 12. The mini-GLT masks the value of this register against transitions on the input lines, and ORs the result. If the result is *true* and the busy line is *not* asserted, a trigger message will be generated. Note: This is a *mask* register. Each of twenty-one different lines is represented by a bit offset, as illustrated in the Figure 12. If the bit at the corresponding offset is *clear,* the corresponding trigger line will be considered in the trigger calculation. If the bit at the corresponding field is *set,* the line will not be used in the trigger calculation.

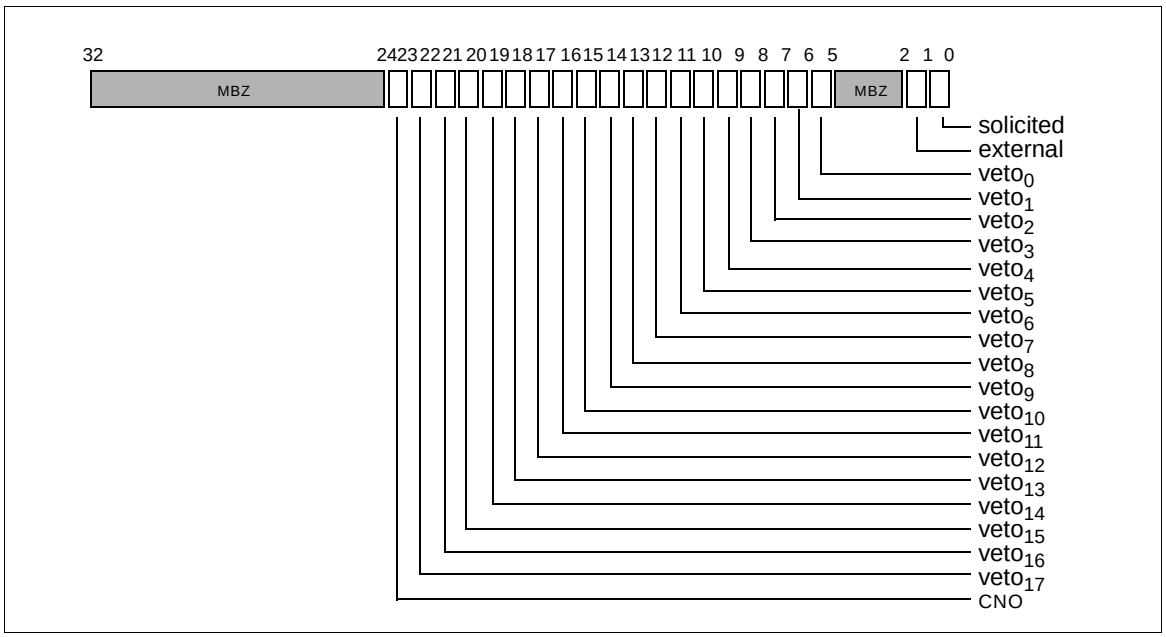

Figure 12 Trigger source mask

3.4.7 Trigger counter mask

The mini-GLT process transitions on twenty-one different trigger lines. The number of transitions on these lines is tallied by the mini-GLT and is available in the trigger counter register described in Section 3.4.8. The register described in this section allows the user to define which transitions, on which lines will be counted by the trigger counter. Each of

twenty-one lines is represented by a bit offset as illustrated in Figure 13. If the bit at the corresponding offset is *clear*, the trigger line will be counted. If the bit at the corresponding field is *set*, the line will not be counted. Note: this counter counts *upstream* of trigger processing; consequently the value of the register described in Section 3.4.6 has *no* effect on which lines are counted.

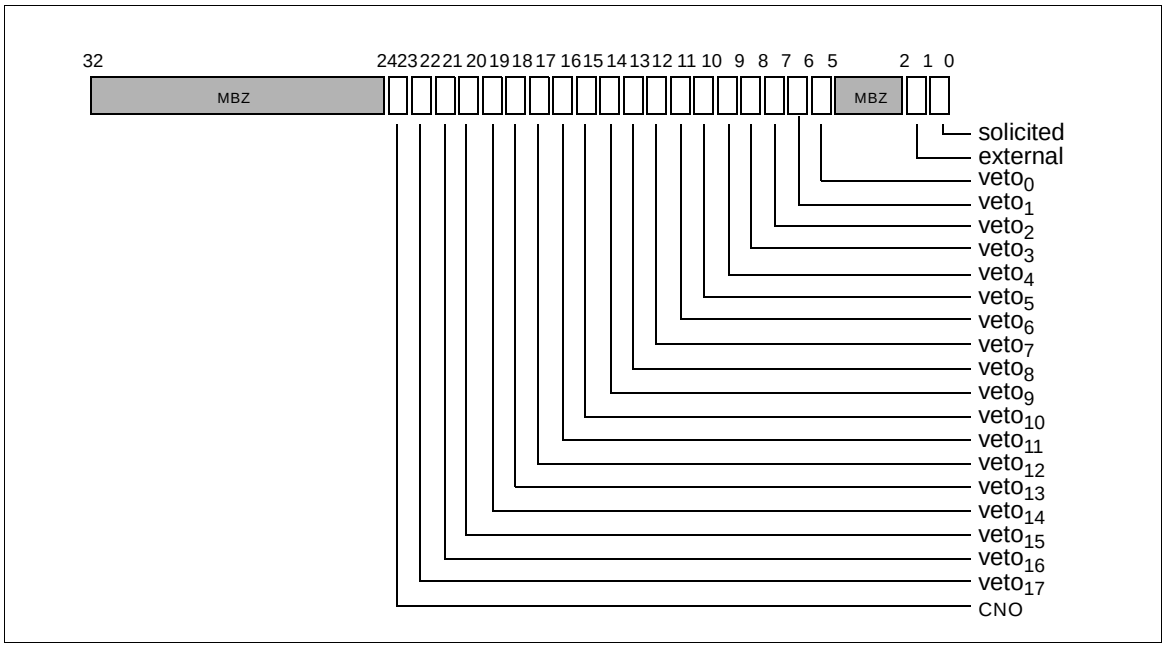

Figure 13 Trigger counter mask register

3.4.8 Trigger counter

The mini-GLT process transitions on twenty-one different trigger lines. The number of transitions on these lines is tallied by the mini-GLT and the counter's current value is available in the register illustrated in Figure 14. The register described in Section 3.4.7 is used to define which of the different lines should be counted. This is a 24 bit counter, which on overflow, simply wraps and continues to count from zero. Either writing to this register[1], or a system reset will initialize this counter to a value of *zero*.

1. The value written is ignored.

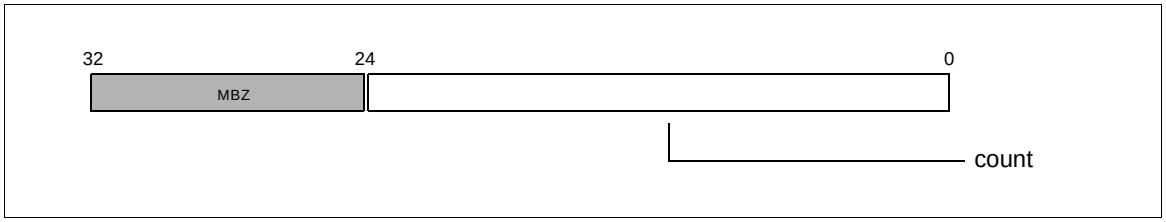

Figure 14 Trigger counter register

3.4.9 Test features register

The mini-GLT has the capability to defeat certain standard features, which in the normal course of operation would always be enabled. This functionality is present only to allow *testing* of those features. One should think long and hard before modifying the default value of this register.

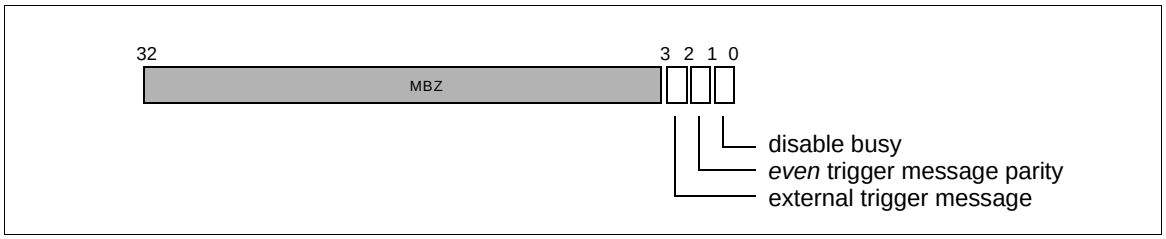

Figure 15 Test features register

disable busy: If *set*, the mini-GLT ignores transitions of the busy line of either the TEM or AEM[1].

***even* trigger message parity:** Determines whether the parity generated by the mini-GLT for a trigger message is *odd* or *even*. If the field is *clear*, *odd* parity is generated. If the field is *set*, *even* parity is generated.[2]

external trigger message: Determines the type of trigger message sent in response to an *external* trigger request. If the field is *clear*, the CALSTROBE and TACK fields of the trigger message will be clear. If this field is *set*, the value of these fields will determined by the values found in the CALSTROBE and TACK fields in the register described above.

1. Implementation note: The *busy* signal for the comm-board implementation is tied to the “almost-full” flag of the record FIFO.

2. Implementation note: The setting of this field determines the parity of the internally generated TACK and CALSTROBE commands. This field is in turn controlled by the emulation of the Configuration Register within the AEM (see Section 3.3.1).

3.4.10 Cables and connectors

Table 2 Signal name assignments of comm board J1 connector to satisfy ICD described in [1]

row[a]	column 0[b]	column 1	column 2	column 3
0	ACD_CLK_xM	ACD_CLK_xP	ACD_NVETO_00xM	ACD_NVETO_00xP
1	ACD_NRST_xM	ACD_NRST_xP	ACD_NVETO_01xM	ACD_NVETO_01xP
2	ACD_NSCMD_xM	ACD_NSCMD_xP	ACD_NVETO_02xM	ACD_NVETO_02xP
3	N/A	N/A	ACD_NVETO_03xM	ACD_NVETO_03xP
4	N/A	N/A	ACD_NVETO_04xM	ACD_NVETO_04xP
5	N/A	N/A	ACD_NVETO_05xM	ACD_NVETO_05xP
6	N/A	N/A	ACD_NVETO_06xM	ACD_NVETO_06xP
7	N/A	N/A	ACD_NVETO_07xM	ACD_NVETO_07xP
8	N/A	N/A	N/A	N/A
9	N/A	N/A	N/A	N/A
10	N/A	N/A	ACD_NVETO_08xM	ACD_NVETO_08xP
11	N/A	N/A	ACD_NVETO_09xM	ACD_NVETO_09xP
12	N/A	N/A	ACD_NVETO_10xM	ACD_NVETO_10xP
13	N/A	N/A	ACD_NVETO_11xM	ACD_NVETO_11xP
14	N/A	N/A	ACD_NVETO_12xM	ACD_NVETO_12xP
15	N/A	N/A	ACD_NVETO_13xM	ACD_NVETO_13xP
16	N/A	N/A	ACD_NVETO_14xM	ACD_NVETO_14xP
17	N/A	N/A	ACD_NVETO_15xM	ACD_NVETO_15xP
18	ACD_GND_0x	ACD_GND_1x	ACD_GND_2x	ACD_28V_RTN_0x
19	ACD_VDD_0x	N/A	N/A	ACD_28V_RTN_1x
20	ACD_28V_0x	N/A	N/A	N/A
21	N/A	N/A	ACD_NVETO_16xM	ACD_NVETO_16xP
22	N/A	N/A	ACD_NVETO_17xM	ACD_NVETO_17xP
23	N/A	N/A	ACD_NCNO_xM	ACD_NCNO_xP
24	N/A	N/A	ACD_NSDATA_xM	ACD_NSDATA_xP

a. Row numbers are assigned starting from the *bottom* of the connector

b. Column numbers are assigned starting from the column *closest* to the PCB (the column labelled out)

4 G3

Generation *3* brings a new component to bear within the test-stand, the GASU box. Although the GASU has a myriad of functionality, from the perspective of the ACD these are its significant features:

- hardware AEM which will now allow the test-stand to support multiple FREE boards
- realistic trigger system (the GLT)

Generation *3* consists of the following components:

LAT supply: Provides 28 volts. Reused from Generation *2*.

One GASU box: Contains a pair of redundant boards. Each board contains an AEM, GLT, Fanin / Fanout Unit (FIFO) and Event Builder (EB).

One transition board: An emulation of the functionality required of the PDU by the GASU.

One LCB: A LAT Communication Board.

Finally, of course, all the cabling to interface these components together is necessary. A diagram which shows these components, cables, and connections is found in Figure 16. Note; that for simplicity the connectivity for *only* one FREE board is illustrated. However, up to twelve boards, each with redundant paths can be accommodated.

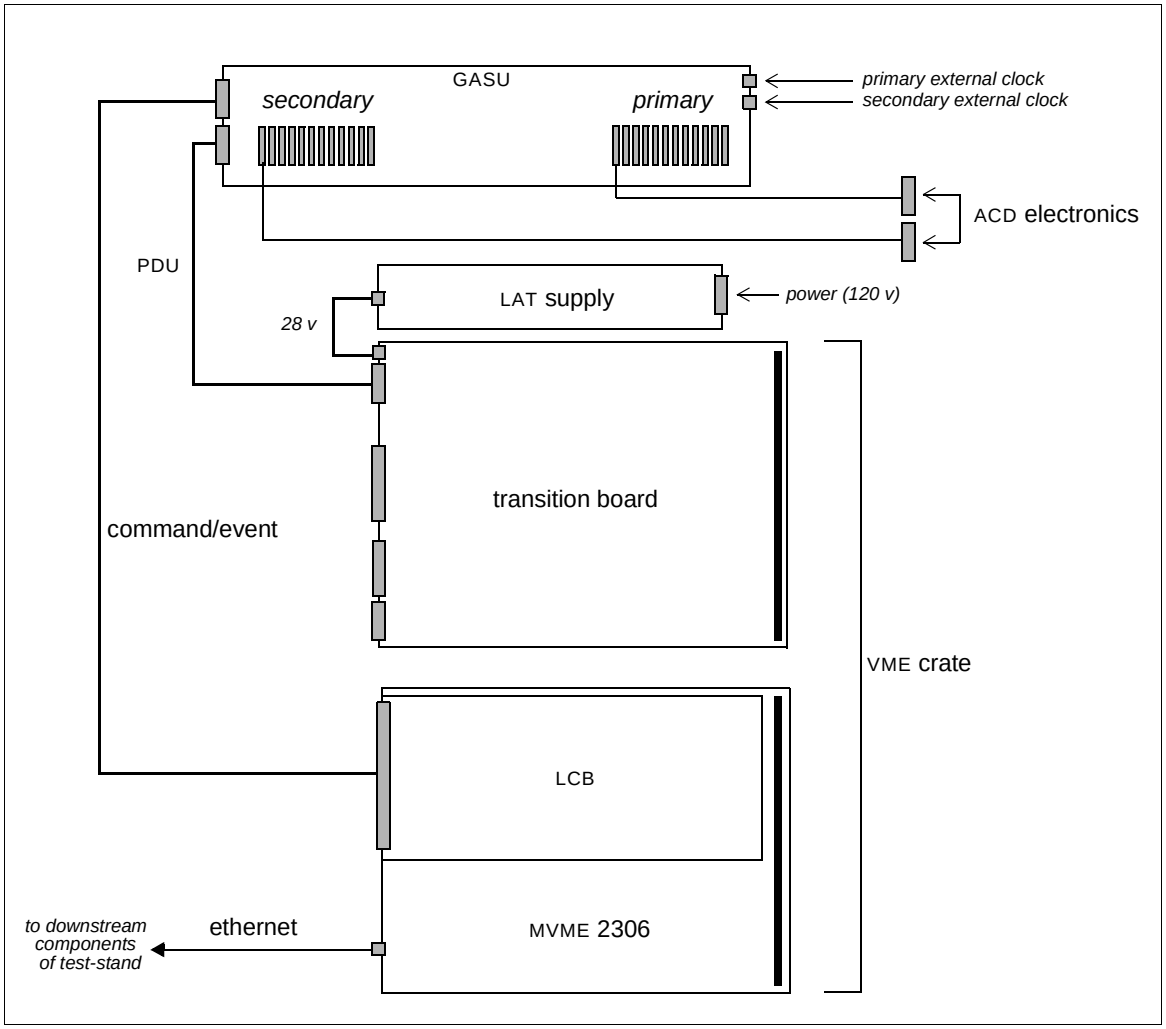

Figure 16 G3 test-stand arraignment

4.1 The GASU box

The GASU *box* contains the following

- Two GASU *boards* (see below), one of which is intended to be redundant.
- Power converters for both the ACD electronics and the GASU itself.
- An enclosure that contains the boards, converters and connectors. This enclosure is designed for thermo-vac use, *including* the engineering version.

4.1.1 The GASU board

The GASU board is composed of four individual, relatively orthogonal modules:

- the ACD Electronics Module (AEM) (see [5]).
- the Global Trigger (GLT) (see [6]).
- the Command / Response FanIn / Fanout Unit (FIFO) (see [8]).
- the Event Builder EB (see [7]).

Each of these modules appear as a separate node on the command / response and event[1] fabrics. In addition, the GASU provides the system clock (as part of the FIFO). This clock may be either internal or external. If internal, the clock runs at the standard LAT rate of 20 MHZ.

In general, to operate the AEM requires programming of the other modules. To study and use the veto and CNO information of the ACD requires using the GLT.

4.2 LAT supply

Same bench supply as used in Generation *2*.

4.3 Transition board

Most of the functionality of the transition board is neutered when used in conjunction with a GASU. What remains is the functionality necessary to support:

- Power margin testing of both the FREE boards and the GASU itself
- Environmental monitoring of temperature and power of the GASU

4.4 LCB

The PMC version of the LCB. This is an engineering board implemented on a PMC form factor rather then its flight form factor, cPCI. It occupies the PMC slots of the *Motorola* MVME 2306 currently used in a test-stand. It provides an exact functional representation of the LCB used in flight.See [4], with particular emphasis on Appendix B. The comm-board emulation of the LCB may be used as risk mitigation against the production of the LCB.

1. Only true for those modules which contribute to an event (the AEM and GLT).

4.5 New Features

Generation *3* contains all the features of Generation *2* plus:

- accurate representation of flight triggering system (the GLT)
- multiple FREE board support.
- removes all restrictions of Generation *2*

4.5.1 Variants

The GASU box, even in its engineering incarnation is a very expensive article to manufacture. The cost is driven principally be its large number of connectors and corresponding cables. In many applications all the GASU's connectors and cables are not required and a partially loaded GASU is attractive from a monetary cost perspective. Therefore, two variants of the GASU are envisioned, one with all its connectors installed and one with only a subset of its connectors installed. The latter variant will be called a ***development*** GASU. It is defined to support the following:

- One FREE board (address fixed, but TBD)
- One TEM (address fixed, but TBD)
- One crate (address fixed, but TBD)

In all other respects a development GASU is no different from a fully loaded GASU. It is likely, but not yet known, given the complexity of fabrication, that once a development GASU has been constructed, it may *not* be upgradable to a fully loaded box. Supporting two variants of the GASU complicates both the fabrication and spares issues. Consequently, more research is necessary to see whether such a two variant scheme is practical.

Test-stands which use a development GASU will be called Generation *3a*. Test-stands which use a fully loaded GASU will be called Generation *3b*.

The GASU box is designed from the beginning to support operation in thermo-vac. However, the work to *test* this assertion has neither been scheduled, or its scope understood. It is also possible then once these tests are accomplished their lessons may be applied by producing yet another variant of the GASU. Note: In any case, an external cooling system must be applied to the GASU for thermo-vac operation.