



Global Level-1 Operations Manual

Rev 0.1
July 13, 1999
J. Lajoie
lajoie@iastate.edu

Introduction

The Global Level-1 trigger (GL1) is the part of the PHENIX online system that is responsible for generating triggers from the Local Level-1 (LL1) reduced bit data , coordinating busies, and managing partitioned running of the PHENIX detector. The GL1 system is comprised of three types of 9U VME-P boards with very specific responsibilities:

- GL1 Board 1 (GL1-1) generates triggers from reduced bit input
- GL1 Board 2 (GL1-2) manages the busies and Dead-4-4 counters
- GL1 Board (GL1-3) generates the granule accept vector and manages the accepted event readout.

This document is intended to describe the typical operation of both the GL1 hardware and software. By referring to this guide you should be able to set up and program GL1. However, it is not intended to make you a GL1 expert or facilitate anything more than rudimentary debugging of the system.

What To Do If You Find an Error in This Manual

As long as PHENIX is in operation, this manual will be a "work in progress." If you find portions of this manual that are misleading or inaccurate, or you have suggestions for making the manual more useful please contact the author (lajoie@iastate.edu).

The current revision (Rev. 0) contains a number of items that are specific to the RHIC Engineering Run (ER).

Definitions

As in all manuals, a few definitions are in order to get things off on the right foot. All computer input, output, typed commands and filenames will appear in **Times Roman Bold** type. All program messages and references to buttons, etc., will appear in **bold** type. File listings are shown in Times Roman type. Computer prompts are shown as ">"; this may be different depending in the system you are logged in to.

The most critical definition, however, is reserved for the phrase "consult an expert". Simply put, an expert is someone who knows more than you. In some cases where you are asked to consult an expert you may know what to do to fix a problem, but it may be more important to find out why things went wrong in the first place.

The PHENIX Level-1 contact is:

John Lajoie
BNL extension 1266 (ER only)
1-515-294-6952 (ISU office)
1-515-963-9818 (home - use this wisely!)
1-515-480-8312 (cell phone - use this very wisely!)

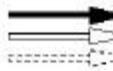
Overview of the Level-1 Trigger

A schematic of the PHENIX Trigger and Timing systems is shown in Figure 1. Note in particular the connections between GL1 and the Granule Timing Modules (GTMs), where each granule controls four signals that can be sent to GL1. Two of these are busies (**FEM Unreliable** and **DCM Busy/Full**) and are sent to GL1-2 to hold off triggers for that granule (and any partition that owns that granule). The other two (**Granule Disable** and **Forced Accept**) are available for use as reduced bit inputs on GL1-1. Note that these inputs are available by programming the LL1 mode bits in the GTM scheduler for any granule, not just LL1 participant systems.

The box labeled LL1 in Figure 1 represents any system that places reduced bit information on the GL1 backplane where it is available for trigger purposes. While in the future this will be dedicated LL1 electronics for systems such as BBC, EMCAL, etc., during the ER this functionality is provided by a transition card in the back of the GL1 crate called the Reduced Bit Input Board (RBIB). This board accepts TTL signals on Lemo cables and converts them to the LVDS standard used by the input transition cards (called the 6Rx due to its six input connectors). The configuration of the GL1 for the ER is described fully in a later section.

In order to better understand how GL1 generates a trigger, it is instructive to look at the flow of a single trigger through the GL1 system, as shown in Figure 2. The available information for making a trigger decision is placed by the 6Rx transitions cards on a backplane bus in the GL1 crate, called the *reduced bit input bus*. The user programs a crossbar to select which bits will be used as the input address to a SRAM lookup table (LUT). These bits can be selected as four groups of 20 (out of the 130 bits available) for each GL1-1 board in the system.

The SRAM LUT is then programmed based on the input address to generate output for selected conditions. For a given trigger, a nonzero output bit from the LUT at this point is called a *raw trigger*. Raw triggers are counted in VME addressable counters on GL1-1. At this point the trigger can be masked off with a user-programmable mask.

LEGEND:  FILLED ARROWS FOR DATA TRANSFER AT BEAM CROSSING RATE
 EMPTY ARROWS FOR DATA TRANSFER AT ACCEPTED EVENT RATE
 DASHED ARROWS FOR DATA TRANSFER AT ASYNCHRONOUS RATE

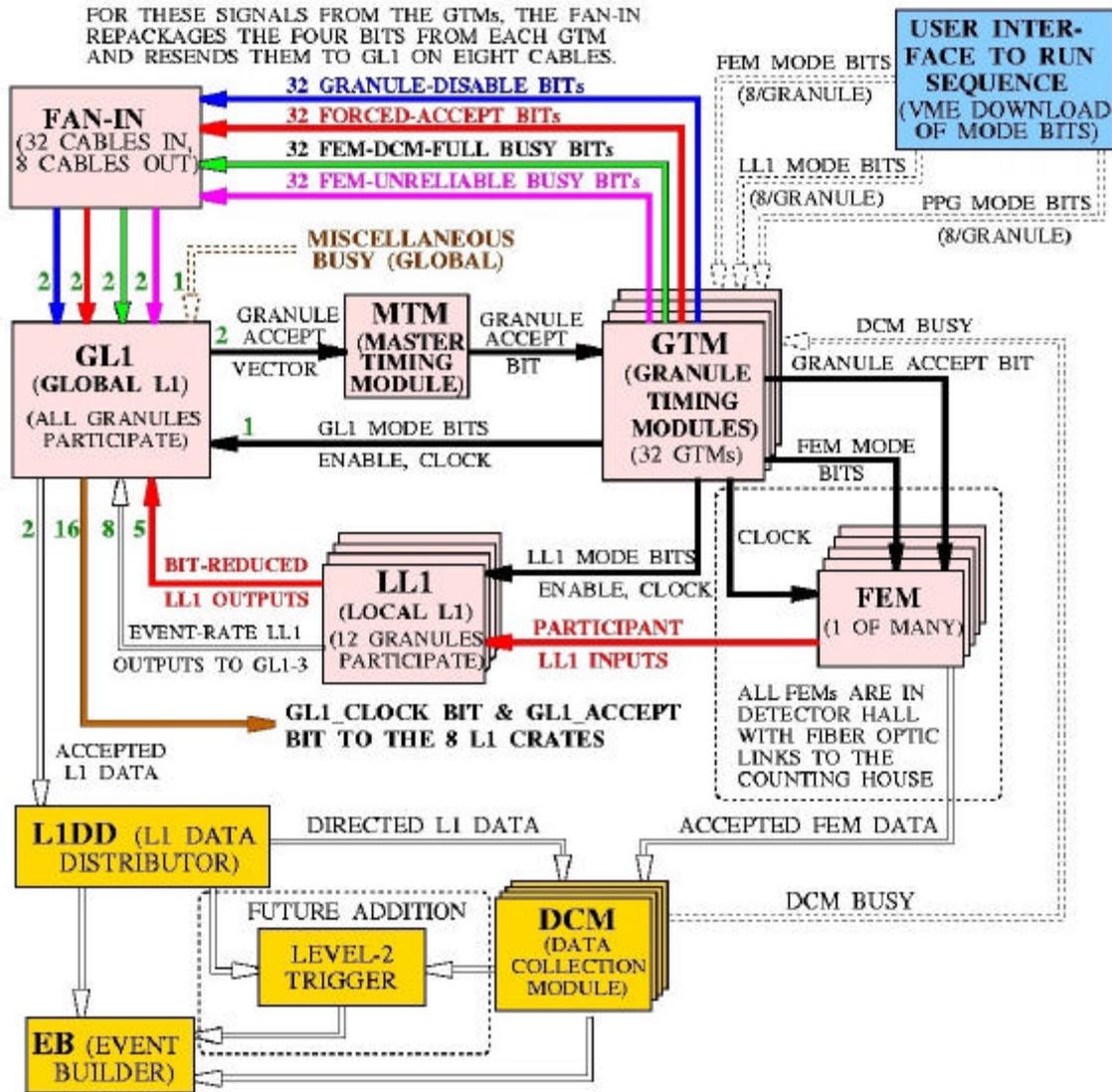


Figure 1: The PHENIX Trigger and timing system. Note in particular the four signals (Granule Disable, Forced Accept, FEM Unreliable, and DCM Busy/Full) that are available for each granule to send as inputs to GL1. The Level 1 Data Distributor does not exist for the Engineering Run; instead the GL1 accepted event data is read out directly via a DCM partitioner board.

The next step is to apply the busy for the partition that owns this trigger. The busies are contained on a second bus on the backplane, called the partition busy bus, that is managed by GL1-2 and includes the Dead-4-4 counters. A programmable crossbar maps the partition busy for the partition that owns the trigger in question and the busy is applied to the raw trigger. A trigger that passes the busy test is called a *live trigger* and is again kept track of by counters in GL1-1.

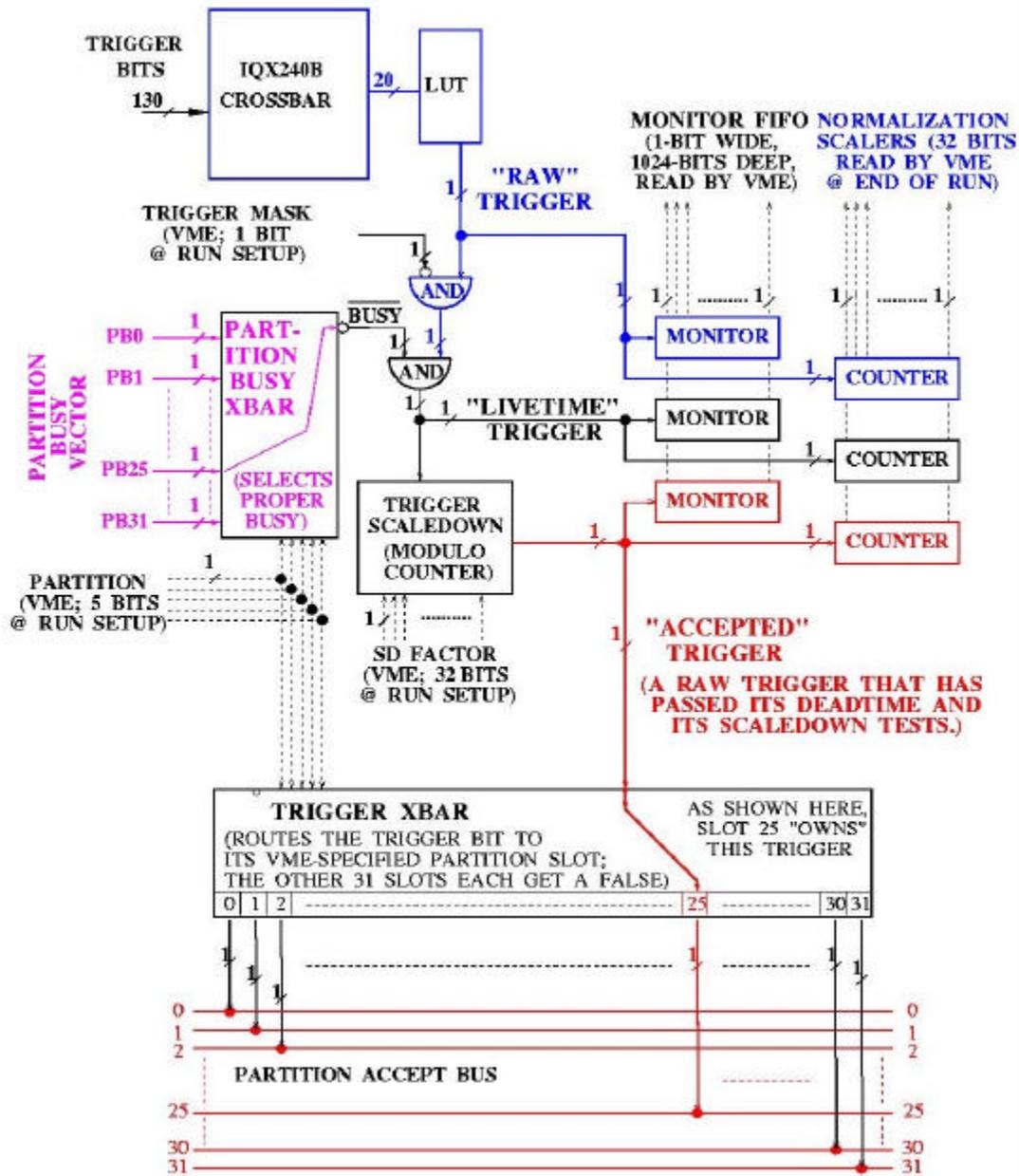


Figure 2: A closeup of a single GL1 trigger, showing the flow and programmable elements in the system.

Finally, a programmable scaledown counter is applied to prescale active triggers, and again the output is counted. A trigger passing the scaledown test is called an *accepted* or *scaled* trigger and is mapped by a third crossbar to a line on the partition accept bus. This bus is monitored by GL1-3 and used to generate the granule accept vector (via another crossbar, not shown in Figure 2) that is sent to the Master Timing Module (MTM).

Like all VME systems in PHENIX, the GL1 crate is controlled by a PPC VME processor (currently ioconddev14). This processor is located in the leftmost slot (slot 0) of the GL1 crate and can be reset by pressing the **Reset** button on the front panel. Note that all the GL1 boards monitor the VME SYSRESET signal and will automatically reset themselves when the VME processor is reset. Thus, resetting the processor will necessitate reloading the GL1 configuration to ensure proper trigger operation. While each of the GL1 boards have independent reset buttons, use of these buttons is strongly discouraged. Should it be necessary, the proper way to reset the GL1 crate is reset the PPC VME processor.

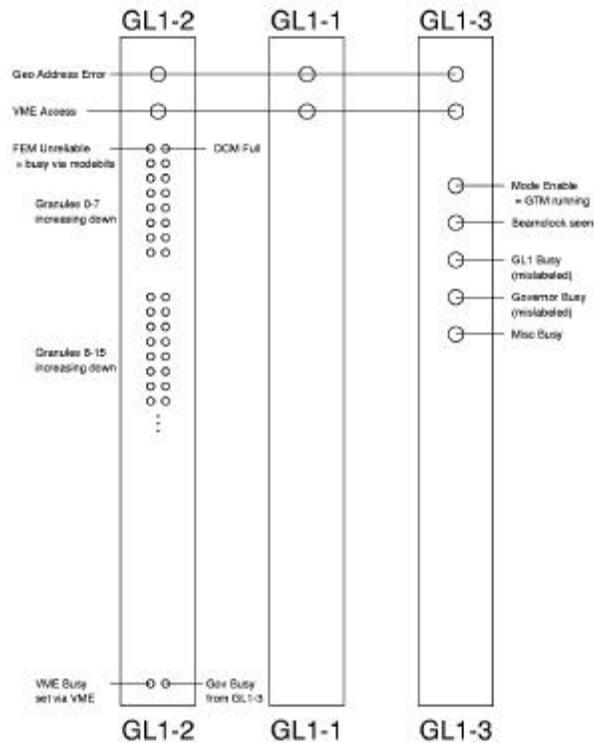


Figure 4 : The GL1 board front panels and a description of their LEDs. The busies indicated by GL1-2 and GL1-2 are likely to be of particular importance to the user.

The GL1 trigger boards contain a set of lights intended to indicate to the user quickly the status of the various GL1 signals, as shown in Figure 4. Of particular importance are the Granule Busy and FEM Unreliable lights on GL1-2. If either of these lights are illuminated for a given granule, triggers for that granule (and any partition that contains it) will be busied out. This is often the first place to look when the complaint "Hey, I'm not getting any triggers!" arises. In addition, the global VME busy on GL1-2 is indicated by a small LED in the lower left of the board.

The GL1-3 board contains a set of "global" busies that will hold off all triggers in all partitions, as indicated by red lights on the front panel. The

Misc/Global busy is (at present) connected to the DCM that is receiving the GL1 accepted event data. The Governor Busy is a busy generated by a programmable "throttle" on the GL1 accepted event rate; currently this throttle is set to limit the accepted event rate to less than 25kHz.

In addition to the busy lights on GL1-3, there is also a green LED to indicate that the GL1 system is receiving a proper beam clock signal from the GTM in the GL1 crate. This light should always be lit during normal operation of the trigger - without a beam clock, most elements of the trigger will not function. When the GTM controlling GL1 is properly cycling (the Mode Enable signal is high) an additional green LED will be lit on the GL1-3 front panel. While this LED should be on for normal operation of the trigger, it should be off before or during configuration of the GL1 system. Certain items, such as the GL1-1 LUTs, will not permit load operations while Mode Enable is high.

Finally, all three boards have a green LED to indicate VME activity. This light can be used as a visual indication to verify that software is actually addressing the correct boards during program operations such as configuration, etc. A red LED at the top of each board indicates failure of the Geographical Address Parity circuit - this LED should never be lit during normal operation. If this LED is lit, it likely indicates a catastrophic failure in the GL1 board or the VME crate - call an expert.

Please note that the GTM controlling the GL1 is not located in the Timing System rack, but in the crate with the GL1 itself. This simplifies the control of GL1 and facilitates software control of the GL1 GTM for downloading and configuration. While the normal GTM tools can be used to configure this GTM and load mode bit files, the GL1 configuration utility takes care of this automatically for ordinary running. In addition, starting and stopping the GL1 GTM is done automatically when using the GL1 configuration utilities.

Much of the cabling that is critical to the proper operation of GL1 is located in the back of the Timing and Control System rack (PRR.1.5). In the bottom of this rack there is an aluminum box labeled GTM->GL1 Transition box which receives 32 RJ45 cables (one from each potential GTM) and maps them to the eight LVDS cables that are sent to GL1. Each RJ45 connector on the transition box corresponds to a very specific granule and labels on the box will show you how to identify the connectors. It is where the RJ45 cable is plugged into the transition box that defines what granule number a given GTM is mapped to. It is important to keep this hardware mapping synchronized with the software described in later sections or mass confusion will result!

In addition, two cables carry the granule accept vector from the Standard Transition card in the GL1 crate (see Figure 3) to the MTM fanout in the back of the Timing and Control rack. This MTM fanout card receives the two LVDS cables from GL1 and splits them out into 32 RJ45 connectors to the GTMs. The

MTM card has two RJ45 blocks; the upper block contains granules 0-15, while the lower contains granules 16-31. Each block's connectors are numbered "like a chip" starting in the upper left hand corner, down the lefthand side, and up the righthand side of the connector.

Once again, the RJ45 cabling is crucial - a given GTM must be connected to the same granule number in both the GTM-> GL1 fanout box and the MTM fanout box. If this is not followed, a given GTM will either not receive accepts (at best) or receive accepts for a different granule (at worst).

The above descriptions of the GL1 and Timing and Control cabling lead to one simple rule - please leave it alone! If you need to swap out a GTM (for whatever reason) leave the cabling and transition card in the back of the crate alone and just swap GTMs in the front of the crate. If you must remove and replace a GTM transition card, double (and triple) check that the GL1 and MTM cabling is correct when you are finished.

Programming the Global Level-1 Trigger

Programming the Global Level-1 trigger consists of five separate steps:

1. Defining the granule-to-partition mapping.
2. Defining the trigger to partition mapping.
3. Defining the reduced bit input to each GL1-1 lookup table.
4. Defining triggers based on the LUT input.
5. Writing a gui config file to download the new configuration.

(Note that steps 2,3, and 4 must be repeated for each GL1-1 board in the system)

This section will take you through each of the five steps in detail. After reading this section you should be able to completely reconfigure the GL1 trigger to suit a wide variety of conditions.

Before we start, let's set some ground rules. The instructions that follow assume that you have logged into phoncs0 as user phoncs, and executed the setup command files for both the production release of the online distribution and the GL1 package. An example session (after you have logged in) would go something like:

> setuponcs R-pro
selecting release R-pro

setting up the ONCS environment R-pro

Current directory is /export/software/oncs/R2.2/online_distribution
> cd GL1/config

The above example will leave you in the directory /export/software/oncs/R-pro/online_distribution/GL1/config. This directory contains the GL1 configuration programs and scripts.

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30	P31
Gran 0																																
Gran 1																																
Gran 2																																
Gran 3																																
Gran 4																																
Gran 5																																
Gran 6																																
Gran 7																																
EMCAL.W																																
TEC.E																																
Gran 10																																
DC.E																																
PC.W																																
BBC																																
Gran 14																																
ZDC																																
Gran 16																																
Gran 17																																
Gran 18																																
Gran 19																																
Gran 20																																
Gran 21																																
Gran 22																																
Gran 23																																
Gran 24																																
Gran 25																																
Gran 26																																
Gran 27																																
Gran 28																																
Gran 29																																
Gran 30																																
Gran 31																																

Figure 5: The granule to partition mapping configuration program. Note that some granules (in the lefthand column) have been given names, as described in the text.

We will start by associating granules with partitions. Execute the script `set_gl2_128b` script by typing `./set_gl2_128b` (While this name may seem rather non-intuitive, the name of the program is associated with a specific type of crossbar in GL1-2). You should be rewarded with a screen like that in Figure 5.

This is a simple mapping program, with partition numbers listed in columns and granule names (or numbers) listed in rows. You assign a granule to a partition by clicking the mouse on the box corresponding to the granule and partition number intersection. If you select a granule incorrectly, you may deselect it by clicking again on the highlighted box. A message will appear under the command buttons confirming the mapping and unmapping actions.

At this point a few important comments are in order. First of all, the granule names and their assignment to granule numbers is contained in the file `.gl1_granule_names` in the config directory (note that this may be a symbolic link to a file in the online configuration directory). The format is very simple - granule number is listed followed by a name, with a granule number -1 terminating the sequence. An example is listed below:

```
8    EMCAL.W
9    TEC.E
12   BBC
15   ZDC
-1
```

You should be aware that only administrative controls keep this file synchronized with the actual hardware configuration (see The GL1 Hardware section). It is critically important that the information in the file be kept synchronized with the GL1 cabling configuration in the Timing and Control rack. This information is used by the `set_gl2_128b` program for display purposes, but it is also used by the run control partition server to list what granules are available in a partition. A great deal of confusion will ensue if the `.gl1_granule_names` file does not match the hardware configuration, so if you must edit it please do so carefully!

OK, so you are happily clicking away and you have a granule to partition map that you like. (Note that the program will prevent you from mapping a granule to more than one partition - a definite PHENIX no-no!) Selecting the **Save File** button at the top of the grid will allow you to save your configuration - choose a filename like `set_gl2_128b_xxxxx.dat`, where `xxxxx` is an identifier of your choice. Something like `set_gl2_128b_modeQ.dat` is fine, but try to be as descriptive as possible. Once you have saved you mapping, you will be able to reload and edit it again later by using the **Load** button.

The next step is to compile the mapping into a set of crossbar files that can be downloaded into the GL1 boards. Click on the **Compile** button; if you receive any message other than **Crossbar files compiled successfully** across

the top of the screen, consult an expert. If all is well, you can exit the program with the **Quit** button.

The compilation procedure for `set_gl2_128b` will have generated two new files - `gl2_128b.actel` and `gl3_128b.actel`, corresponding to the granule to partition mapping required by crossbars on GL1-2 and GL1-3. You should move these files to the online configuration area:

```
> mv gl2_128b.actel $ONLINE_CONFIGURATION/GL1/gl2_128b_XXXXX.actel
> mv gl3_128b.actel $ONLINE_CONFIGURATION/GL1/gl3_128b_XXXXX.actel
```

where `XXXXX` is the configuration name of your choice.

The next step is to assign triggers to partitions. This is done with the `set_gl1_128b` program, which can be executed by typing `./set_gl1_128b` which will produce the configuration screen shown in Figure 6.

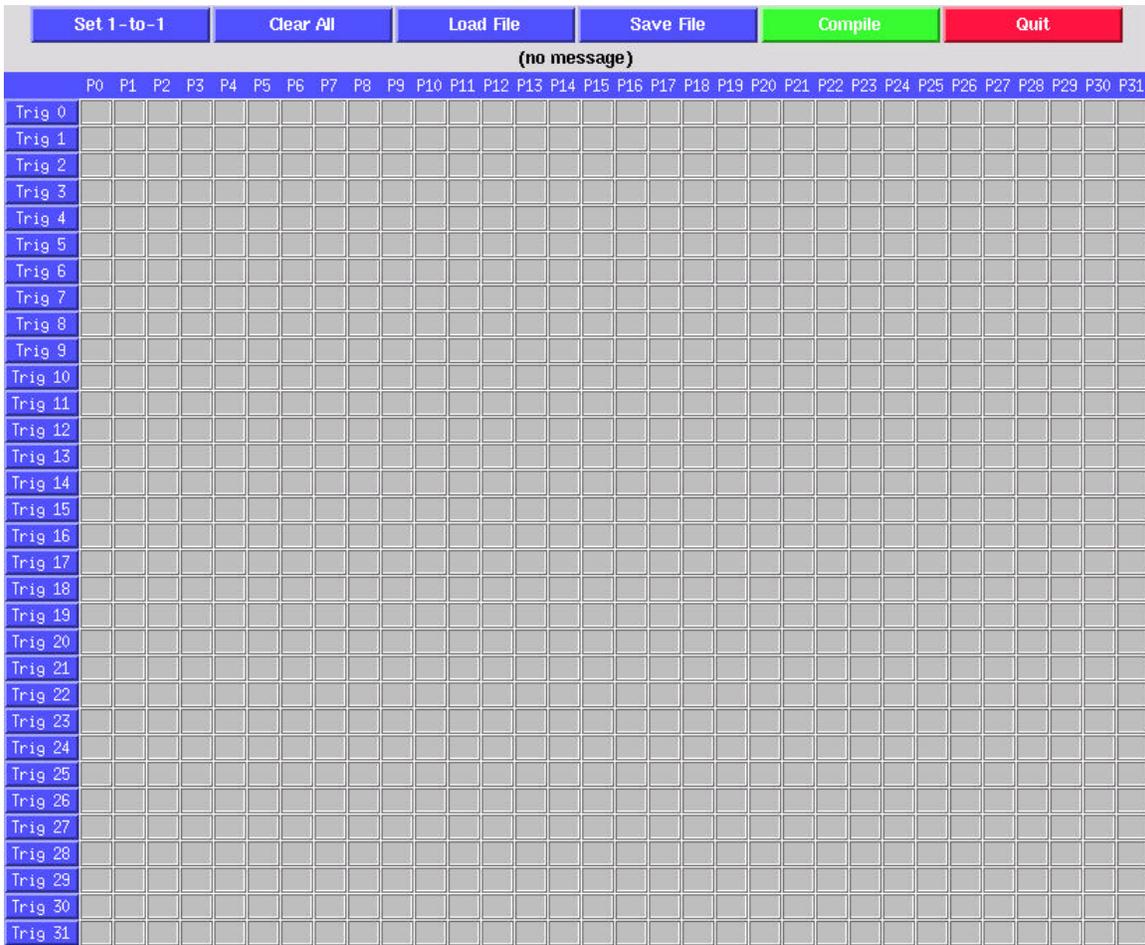


Figure 6: The trigger to partition mapping program. The interface is similar to that used for the granule to partition mapping program.

Since the interface for this program is almost identical to the `set_gl2_128b` program we have previously discussed, we won't go over it in great detail here. Some of the same caveats for mapping granules to partitions apply for mapping triggers to partitions, however. Note that a given trigger can belong to one and only one partition, and the program will prevent mapping a trigger to more than one partition.

When you are happy with the trigger to partition mapping you have generated, save the file under the name `gl1_128b_XXXXX.dat` and compile the crossbar files. You will be rewarded with two new files, `gl1_128b.actel` and `gl1_128b.bin`. These two different types of files are used to provide mappings for two different types of crossbars on GL1-1. As before, copy these files to the online configuration area, renaming them to reflect the name of your configuration:

```
> mv gl1_128b.actel $ONLINE_CONFIGURATION/GL1/gl1_128b_XXXXX.actel
> mv gl1_128b.bin $ONLINE_CONFIGURATION/GL1/gl1_128b_XXXXX.bin
```

The next step is to define the address to the LUTs that will be used to define each trigger by mapping lines from the reduced bit input bus; you will have to do this for each trigger you mapped to a partition in the previous step. Start the configuration program by typing `./set_gl1_240b`. You should see a configuration screen like that shown in Figure 7.

It is useful to remember the architecture of GL1-1 at this point. Each set of four triggers is generated as the output of a LUT that takes 20 bits of address from the GL1 reduced bit input bus. Your job at this point is to define which 20 bits will be mapped to which lookup table. Triggers 1-4 are controlled by LUT 1, 5-8 by LUT 2, etc. Select the LUT you want to work with by the radio buttons at the left of the configuration screen, then select the GL1 reduced bit input bus elements from the menu at the right. You can clear a selection with the right mouse button. Start from the lowest bits in the address and work your way up. Don't worry about uncommitted address bits; the program will automatically force those bits low when the crossbar files are compiled.

As an example, Figure 7 shows a configuration where LUT 1 is using **Forced Accept 9** (the Forced Accept bit for granule 9) as the input to bit 0 of the LUT address.

You may want to record your configuration for the 4 LUTs separately if your configuration becomes complicated, as they will be used in the next step to generate the LUT data files themselves.

As before, save your configuration under the name `gl1_240b_XXXXX.dat` and then compile the crossbar files. If all goes well, you will have generated two

new files: **gl1_240b.U11.bin** and **gl1_240b.U12.bin**. As before, copy them to the online configuration area and rename them to match your configuration name:

```
> mv gl1_240b.U11.bin $ONLINE_CONFIGURATION/GL1/gl1_240b_XXXXX.U11.bin
> mv gl1_240b.U12.bin $ONLINE_CONFIGURATION/GL1/gl1_240b_XXXXX.U12.bin
```

Up to this point, configuring the GL1 has been a purely graphical exercise. However, the program to generate the GL1 LUT files is a command line utility that takes a configuration file as input. What follows next may sound complicated, but keep in mind the basic job of this utility is to loop over all combinations on the 20 bit LUT address and generate raw triggers for only those combinations that you select.

To start with, copy over the example file to a file name representing your current configuration and edit the new file:

```
> cp gl1_lut.input.example gl1_lut_XXXXX.input
> emacs gl1_lut_XXXXX.input
```

where **XXXXX** is the name of your configuration. The above commands should open the file in an emacs window, and the file it contains should look something like this:

```
NAMES
FORCED_ACCEPT9 0x1
UNUSED 0xFFFFFE
END

TRIG 0
END

TRIG 1
END

TRIG 2
END

TRIG 3
END

TRIG 4
FORCED_ACCEPT9.GT.0x0
UNUSED.EQ.0x0
END
```

(The listing should include all 32 triggers as separate sections, but I have truncated the list here to save space.) As you can see, the LUT input file

contains sections which start with a section name (NAMES, TRIG 1, etc.) and are terminated with the END keyword. The NAMES section defines named masks on the 20 bit LUT address, while the TRIG sections generate LUT entries based on those named masks. Note that numbers listed in these section are listed in hexadecimal format (starting with a "0x").

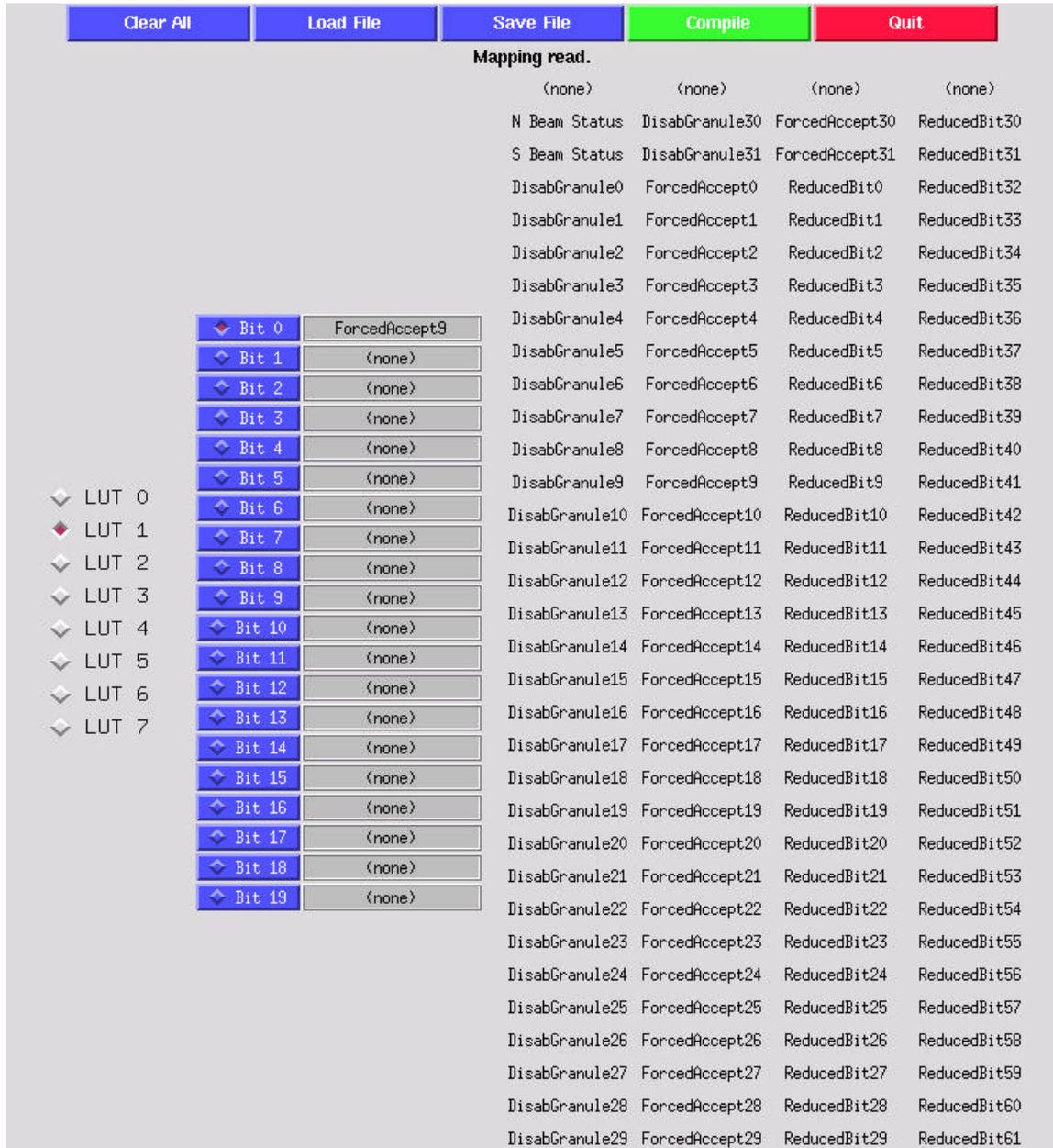


Figure 7: The GL1 lookup table address mapping program. The LUT to be configured is selected by the radio buttons at the left, while the address bit to be mapped is selected by the radio buttons for each bit. Clicking on a named signal will map that signal to the input bit.

The example file contains two named masks - FORCED_ACCEPT9 and UNUSED. FORCED_ACCEPT9 is a mask that selects the first bit (bit 0) in the

LUT address, while UNUSED is a named mask that selects all other bits. These masks are used to generate triggers in the TRIG sections that come later in the file. In the example, trigger four (in the TRIG 4 section) is defined as the address bits specified by FORCED_ACCEPT9 being set while the unused bits are low (recall that in the `set_gl1_240b` program unused LUT address bits are grounded). This is a complicated way of coding a trigger that will fire when the forced accept bit for granule nine is set.

When writing a LUT configuration input file, keep in mind that you should use named masks that represent the data bits they are defining. This will make it much easier later on when you try to recall how a trigger was defined from the input file. Also, note that close coordination is required between the mapping you define for address bits using the `set_gl1_240b` program and the triggers you define in the LUT input file. Getting confused here is a great way to screw up a trigger, and at present there is no software enforcement that the address bit mapping and the LUT definitions coincide (One way to help this would be to have the `set_gl1_240b` program generate a skeleton trigger input file - JGL).

Fill in the NAMES section with masks that define the trigger elements you want to use, then fill in the TRIG section with tests based on these masks to define trigger conditions. The comparisons in the TRIG section recognize the keywords ".GT." (greater than), ".LT." (less than), ".EQ." (equals), ".GE." (greater than or equal to) and ".LE." (less than or equal to).

When you are ready, save the file, exit the emacs editor, and generate the LUT data file from the input file:

```
> ./gl1_lut gl1_lut_XXXXX.input gl1_lut_XXXXX
Collecting input names...done - found 16 named inputs.
Collecting trigger definitions...done
Generating LUT entries according to rules...done
```

This will generate a file `gl1_lut_XXXXX.lut`, assuming there were no errors. If there were any errors in format, etc., the program will try to indicate to you the nature of the error and the section it was found in. Move the LUT file to the online configuration area:

```
> mv gl1_lut_XXXXX.lut $ONLINE_CONFIGURATION/GL1/gl1_lut_XXXXX.lut
```

It is important to note that for the ER there is only one GL1-1 board. When there are multiple GL1-1 boards in the near future, you will have to repeat the previous three steps (`set_gl1_128b`, `set_gl1_240b` and `gl1_lut`) for each GL1-1 board in the system. (NOTE: the naming convention for the files will also have to change to indicate which GL1-1 board the files are for. JGL)

At this point you have generated all of the necessary configuration files for GL1. You must now generate a file that defines this configuration for the GL1 download program. Start by changing directory to the online configuration area for GL1, copying the example file over to a new file tagged with your configuration name, and opening the file in an editor window:

```
> cd $ONLINE_CONFIGURATION/GL1
> cp config_startup_example.config config_startup_XXXXX.dat
> emacs config_startup_XXXXX.dat
```

The example config file should look like:

```
# Crate startup config file

BOARD GL1-1 10 {
  actelxbar gl1_128b_XXXXX.actel
  xbar2    gl1_128b_XXXXX.bin
  xbar1    gl1_240b_XXXXX.U12.bin
  xbar0    gl1_240b_XXXXX.U11.bin
  trmask   0x0
  lut      gl1_lut_XXXXX.lut
# pipectl 0x39
}

BOARD GL1-2 8 {
  actelxbar gl2_128b_XXXXX.actel
  tstreg    0x0 0x0 0x00040002
  pipectl   0x39
}

BOARD GL1-3 12 {
  actelxbar gl3_128b_XXXXX.actel
  orxbar    0xFFFFFFFF
  govdelay  377
  pipectl   0x08
}

BOARD GTM 3 {
  modebits  GTM.GL1.gtm
}
```

This file consists of a number of named sections for the GL1 boards. Comments lines are preceded by the # character. To update this file for your configuration, replace the **XXXXX** entries with the chosen configuration name. Note that this format allows you to create a mixed configuration from a variety of configuration names, but that will be very difficult to maintain in the long run.

Other entries, such as pipectl, orxbar, govdelay, the section headers, and the GTM section should be left alone and are for "experts only".

For the ER, only one GL1-1 board is in place. When there are additional GL1-1 boards available, each board will have an entry in the configuration file.

If you have reached this point after working through all of the above configuration steps, congratulations! You now have a complete GL1 configuration and you are ready to try it out by downloading it into the GL1 system.

The GL1 Management GUI

The GL1 system is most easily controlled by the average user though a Tcl/Tk gui. This gui encapsulates the most common GL1 functions into a set of easy to follow menus and buttons.

To start the gui, change directory to the GL1 area and start the gl1 script:

```
> cd $ONLINE_DISTRIBUTION/GL1/gui
> gl1
```

This will start up the gui, and you will be presented with a window with buttons across the top for the various GL1 elements. You should see a window that looks like that in Figure 8. Each page of the gui is organized the same: the top row of buttons accesses the configuration pages for the different boards, the **Options** section contains buttons for the user commands, while the **Status** section displays status information. From this window you can control the configuration of the gui itself (with such command as **Set rsh command** and **Set timeout**) as well as the GL1 crate itself. At the bottom of the window is a listing of the number of boards in the GL1 crate, the rsh command used, the timeout value, and a checkbutton that indicates the current state of the GL1 global VME busy.

The configuration of the GL1 crate (in terms of what board is in what slot) is loaded by the GL1 gui from the file **gl1_crate.config** in the GL1 online configuration directory. At the present time this file looks like:

```
#Wed Jun 9 14:13:32 CDT 1999
#VME  TYPE  SERIAL  TRANS CARD
10   GL1-1  0        0
8    GL1-2  0        0
12   GL1-3  0        0
```

indicating that there is a GL1-1 board in slot 10, a GL1-2 board in slot 8, and a GL1-3 board in slot 12. Editing this file should be left for experts only, and is included here only for reference.

If you are having a problem with GL1 and you suspect that a board is dead or not answering, you can scan the current crate configuration with the **Scan crate** button. This will pop up a window showing you the configuration of GL1 boards found in the crate, along with any transition cards.

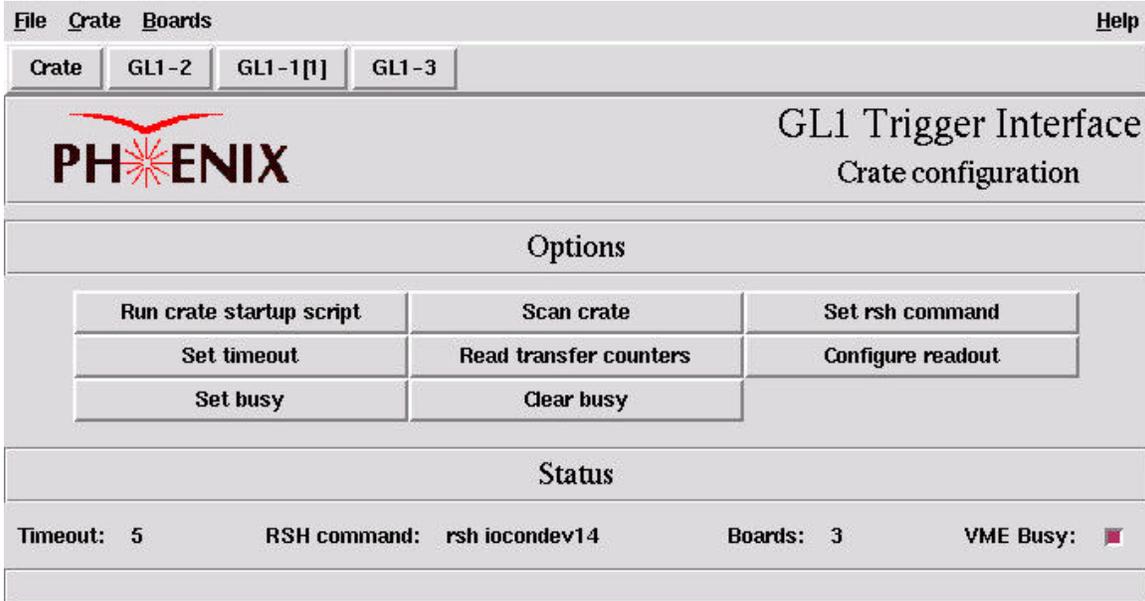


Figure 8: The GL1 gui crate configuration window. Commands are accessed using the buttons under the Options heading or by the menu headings in the upper left. Configuration pages for different elements can be accessed using the buttons directly above the PHENIX logo.

In order to download the current configuration that you defined in the previous section, select the **Run crate startup script** button. You will be presented with a file box that you can use to select the `crate_startup_xxxxx.config` file for your configuration. At this point the gui will parse the configuration file and download the GL1 components - if you watch the VME access lights on the boards, you will see multiple accesses for each board. When the download is complete, you will be presented with a window asking you to **Configure readout**. This command sets the number of word to be read out from each board for an accepted event - just click the **Set** button and don't edit any of the values. Finally, you will be presented with a text window that will display a log file of the download process. If you suspect that something has gone wrong, you should see an error message in this file from one of the configuration programs on the VxWorks processor. If all is well, click the **OK** button.

The entire process should take less than one minute from the time you click the Run crate startup script button to the point where you are presented with

the log file window. If you suspect that the program is hanging, the problem is likely with the rsh daemon on the VxWorks processor. See the instructions in the previous section on resetting the VME processor in the GL1 crate.

Note that running a crate configuration script automatically sets the global VME busy, and the **VME Busy** checkbox should be red after the script is complete. In order to clear the busy, click the **Clear busy** button on the GL1 gui crate page. The trigger will now be fully enabled, and should now provide triggers and accepts (for those partitions that are not themselves busy, of course).

The GL1-1, GL1-2 and GL1-3 pages in the configuration gui contain additional commands that allow you to read out and reset the GL1 counters. Note that counter readout windows are updated automatically by a background task - if you want to monitor the GL1 counters you can set the counter window in a corner of our screen and it will update automatically, approximately every second.

A consequence of this automatic update procedure via *rsh* (remote shell) is that every instance of the GL1 configuration gui will try to read from the GL1 boards every second. If too many people start the GL1 gui and leave the program running the rsh daemon running in the PPC controller will eventually die and all communication with the GL1 crate will be lost. If this happens, reset the PPC controller as described in a previous section.

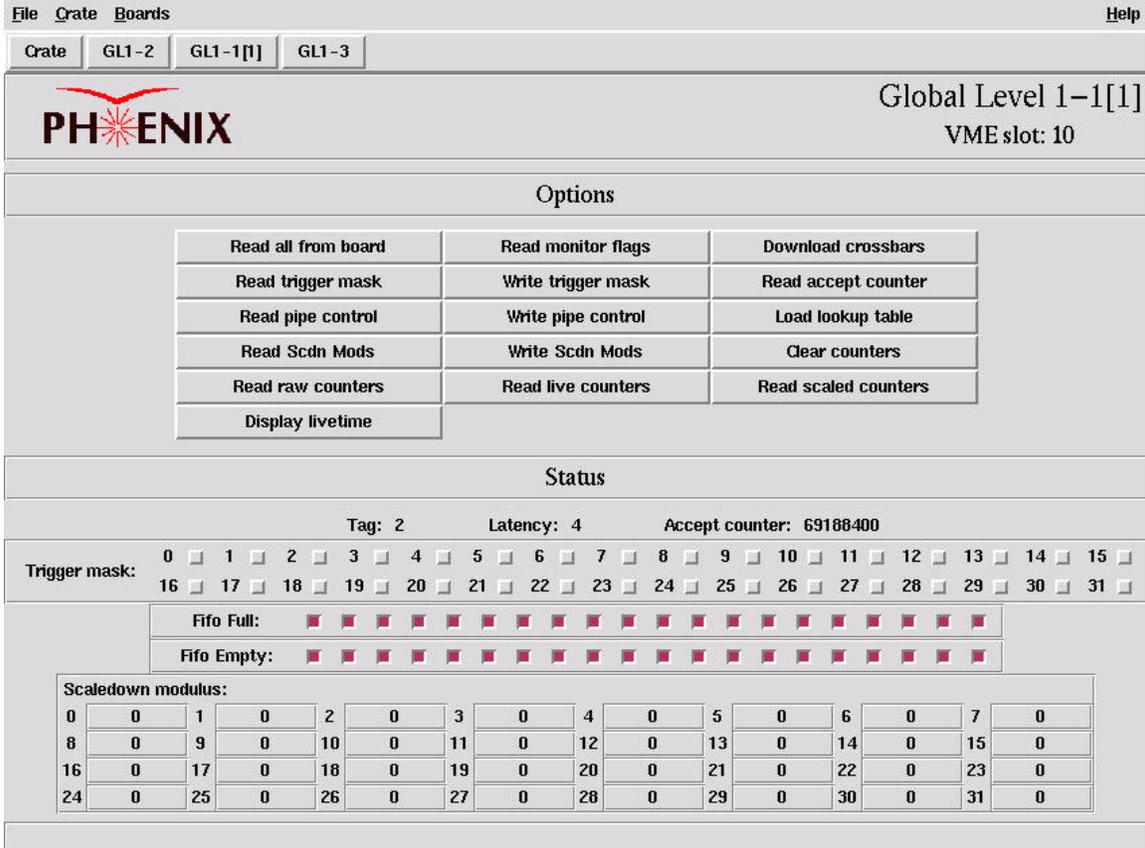


Figure 9: The GL1-1 page of the GL1 configuration gui. Access to various GL1-1 functions is provided through the buttons in the top part of the screen, while status information regarding the trigger mask, monitor fifos and scaledown settings is shown in panels at the bottom of the window.

An example of the GL1-1 window is shown in Figure 9. The GL1-1, GL1-2 and GL1-3 functions will be more fully described as this document is expanded. For now, feel free to explore. Of particular use are the **Clear counters** function on the GL1-1 page (which clears the raw, live and scaled trigger counters for the board) and the **Clear granule counters** function (which allows you to clear selected granule accept counters in GL1-3).

The GL1 ER Configuration (June-July '99)

For the Engineering Run (ER) there are no LL1 systems in place. The LL1 functionality is provided by a set of reduced bit inputs from scintillation counters, the BBC and ZDC via the RBIB board.

These reduced bit definitions are defined and indicated in the listing below. If you notice an error in this listing or it has become out of date, please notify the author (lajoie@iastate.edu). You can use this listing below as a reference when defining triggers based on reduced bits for the ER. The ZDC and BBC signals are self-explanatory, signal names starting with "T" refer to the

scintillation counters, and the BFEBbunch signal is a signal from the ATR indicating injection into the RHIC blue ring.

Table 1: The ER run reduced bit assignments for the BBC, ZDC, and scintillation counters.

Cable No.	Panel No.	RBIB Input No.	Reduced Bit No.	Signal Name
10	1	1	0	BBC(N)
11	2	2	1	BBC(S)
12	3	3	2	BBC(N)*BBC(S)
13	4	4	3	ZDC(N)
14	5	5	4	ZDC(S)
15	6	6	5	ZDC(N)*ZDC(S)
16	7	7	6	(BBC(N)*BBC(S)) AND (ZDC(N)*ZDC(S))
17	8	8	7	TN
18	9	9	8	TS
19	10	10	9	T1E
20	11	11	10	T2E
21	12	12	11	T1W
22	13	13	12	T2W
23	14	14	13	TEM1
24	15	15	14	TEM2
-	-	16	15	BFEBBunch