Global Level-1 Operations Manual

Rev 1.0

March 29, 2000

J. Lajoie & S. Belikov

lajoie@iastate.edu

belikov@bnl.gov

Iowa State University, Ames, Iowa

Index

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 authors (lajoie@iastate.edu and belikov@bnl.gov).

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, and 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 (Year-1 run 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!)

Sergei Belikov, PHENIX Counting House (7821,7815)

924-0458 (home)

Back to Index

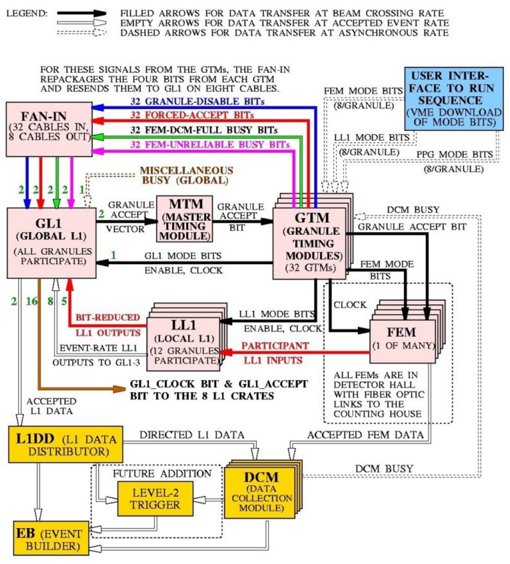
Overview of the Level-1 Trigger

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
FOR THESE SIGNALS FROM THE GTMs, THE FAN-IN REPACKAGES THE FOUR BITS FROM EACH GTM AND RESENDS THEM TO GL1 ON EIGHT CABLES.
USER INTER-FACE TO RUN SEQUENCE (VME DOWNLOAD OF MODE BITS)
FEM MODE BITS (8/GRANULE)
LL1 MODE BITS (8/GRANULE)
PPG MODE BITS (8/GRANULE)
FAN-IN (32 CABLES IN, 8 CABLES OUT)
32 GRANULE-DISABLE BITs
32 FORCED-ACCEPT BITs
32 FEM-DCM-FULL BUSY BITs
32 FEM-UNRELIABLE BUSY BITs
MISCELLANEOUS BUSY (GLOBAL)
2 2 2 2 1
GL1 (GLOBAL L1) (ALL GRANULES PARTICIPATE)
2 GRANULE ACCEPT VECTOR
MTM (MASTER TIMING MODULE)
GRANULE ACCEPT BIT
GTM (GRANULE TIMING MODULES) (32 GTMs)
DCM BUSY
GRANULE ACCEPT BIT
1 GL1 MODE BITS ENABLE, CLOCK
FEM MODE BITS
CLOCK
2 16 8 5
BIT-REDUCED LL1 OUTPUTS
EVENT-RATE LL1 OUTPUTS TO GL1-3
LL1 (LOCAL L1) (12 GRANULES PARTICIPATE)
LL1 MODE BITS ENABLE, CLOCK
PARTICIPANT LL1 INPUTS
FEM (1 OF MANY)
ALL FEMs ARE IN DETECTOR HALL WITH FIBER OPTIC LINKS TO THE COUNTING HOUSE
GL1_CLOCK BIT & GL1_ACCEPT BIT TO THE 8 L1 CRATES
ACCEPTED L1 DATA
L1DD (L1 DATA DISTRIBUTOR)
DIRECTED L1 DATA
ACCEPTED FEM DATA
DCM BUSY
FUTURE ADDITION
LEVEL-2 TRIGGER
DCM (DATA COLLECTION MODULE)
EB (EVENT BUILDER)

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.

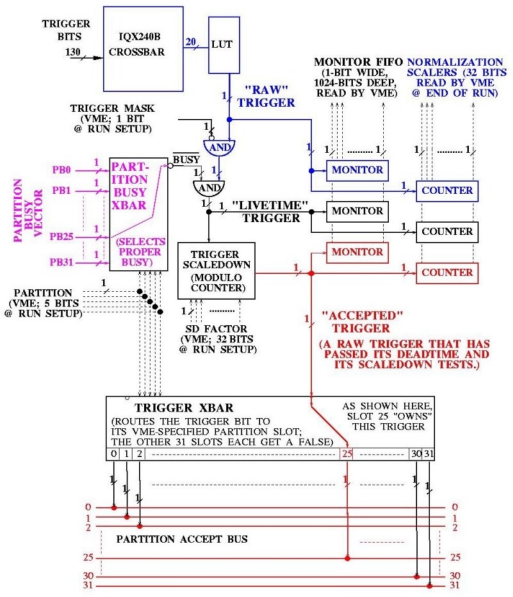

TRIGGER BITS
130
IQX240B CROSSBAR
20
LUT
"RAW" TRIGGER
MONITOR FIFO (1-BIT WIDE, 1024-BITS DEEP, READ BY VME)
NORMALIZATION SCALERS (32 BITS READ BY VME @ END OF RUN)
TRIGGER MASK (VME; 1 BIT @ RUN SETUP)
AND
BUSY
AND
PARTITION BUSY VECTOR
PB0
PB1
PB25
PB31
PART-ITION BUSY XBAR
(SELECTS PROPER BUSY)
"LIVETIME" TRIGGER
MONITOR
COUNTER
MONITOR
COUNTER
MONITOR
COUNTER
TRIGGER SCALEDOWN (MODULO COUNTER)
PARTITION (VME; 5 BITS @ RUN SETUP)
SD FACTOR (VME; 32 BITS @ RUN SETUP)
"ACCEPTED" TRIGGER
(A RAW TRIGGER THAT HAS PASSED ITS DEADTIME AND ITS SCALEDOWN TESTS.)
TRIGGER XBAR
(ROUTES THE TRIGGER BIT TO ITS VME-SPECIFIED PARTITION SLOT; THE OTHER 31 SLOTS EACH GET A FALSE)
AS SHOWN HERE, SLOT 25 "OWNS" THIS TRIGGER
0 1 2 25 30 31
PARTITION ACCEPT BUS

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).

From the point at which reduced bit data is presented on the reduced bit input bus to when the granule accept vector is sent to the MTM requires three beam clocks in GL1.

Back to Index

The GL1 Hardware

The GL1 trigger crate is located in PRR.1.1 in the electronics area of the PHENIX counting house. The Timing System is located in rack PRR.1.5. The Level-1 trigger rack is typically left closed to prevent dust and contamination from building up on the GL1 boards. If you need to open the rack you will need a "crate key".

For reference, Figure 3 shows a schematic diagram of the GL1 crate and associated 9U VME cards, transition cards and beam rate inputs. This diagram can be used to identify the GL1 transition cards and GL1 internal cabling. It should be noted that each end of the GL1 LVDS cabling is labeled with the signal names on each connector end.

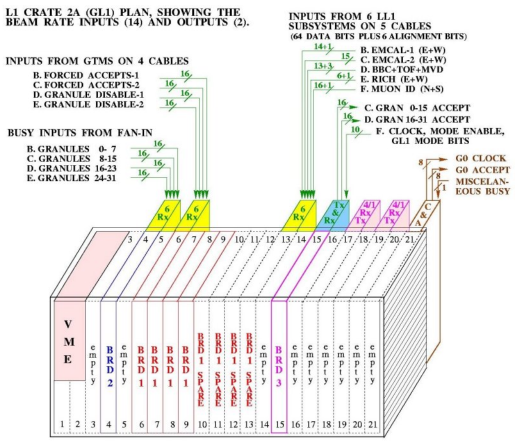


Figure 3: A schematic view of the GL1 crate, showing the associated transition cards and beam rate inputs. Each line in the above corresponds to an LVDS cable. Note that for the ER GL1 uses a shorter backplane and the 9U and transition cards may not be located exactly as shown.

Like all VME systems in PHENIX, the GL1 crate is controlled by a PPC VME processor (currently iocondev14). 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 <u>strongly discouraged</u>. Should it be necessary, the proper way to reset the GL1 crate is reset the PPC VME processor by pressing "Reset" button on the PPC front panel.

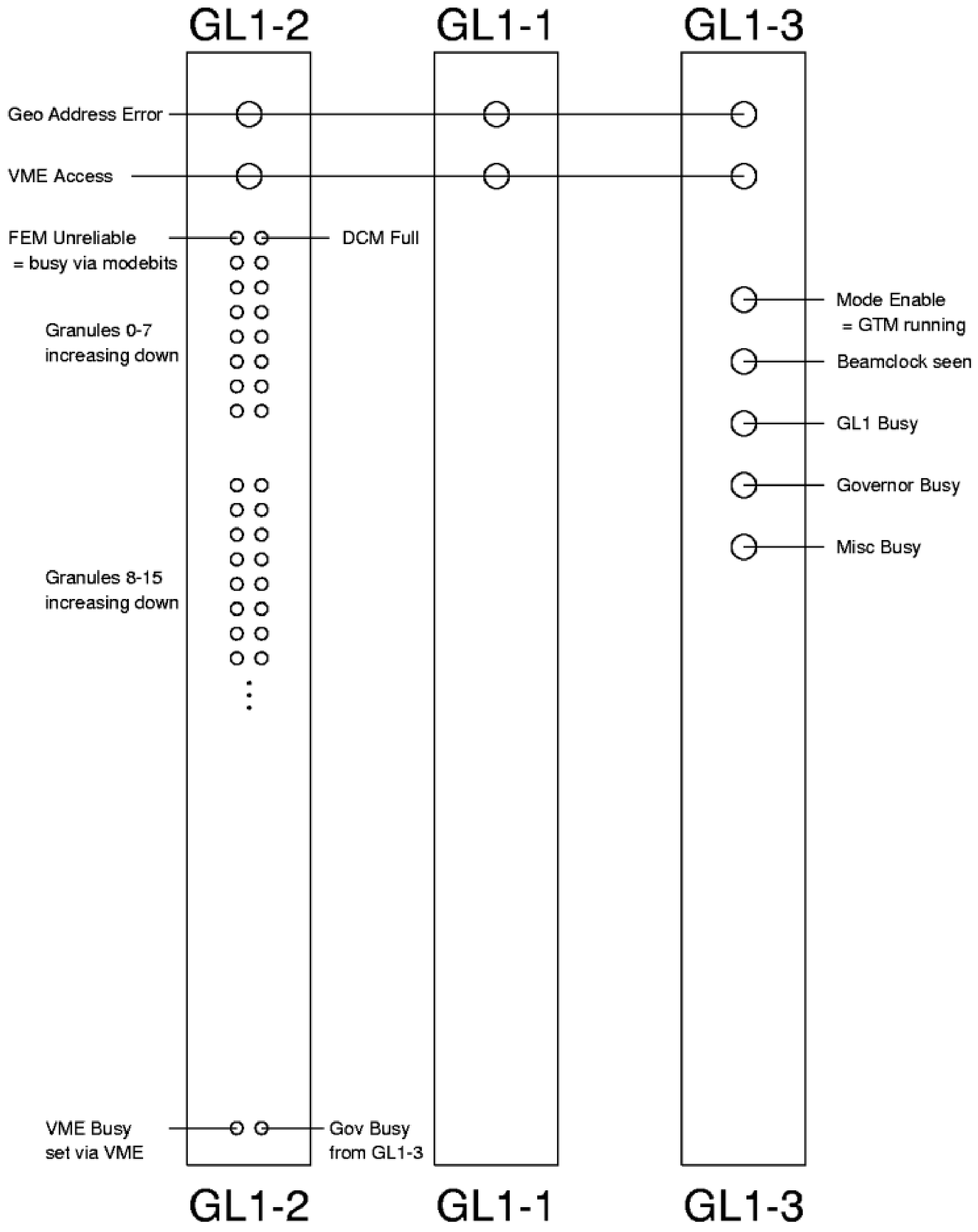


Figure 4. The GL1 board front panels and a description of their LEDs. The busies indicated by GL1-2 and GL1-2 are likely to be of particular importance to the user.

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 corner 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.

Back to Index

Programming the Global Level-1 Trigger

Programming the Global Level-1 trigger consists of five separate steps, that are combined in one GUI:

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. Creating Look Up Table.
6. Setting scale down counters.

(Note that steps 2-6 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. There are two reasons for this:

1. GL1 requires the commercial program iqxcomp that can run only on Sun computer.
2. The software creates subdirectories and files inside the GL1 directory that belongs to PHONCS.

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

----

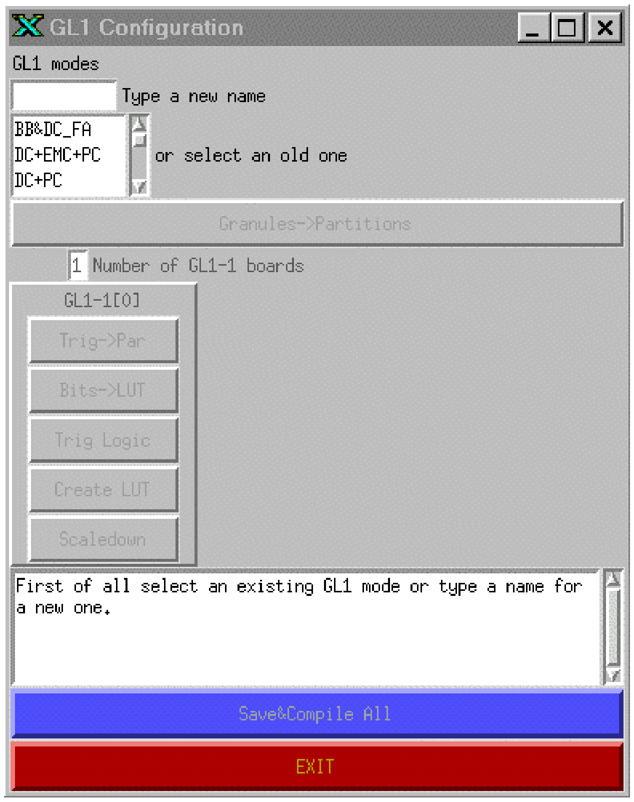
cd $GL1_MAIN/config

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.

Back to Index

How to create a new GL1 configuration mode

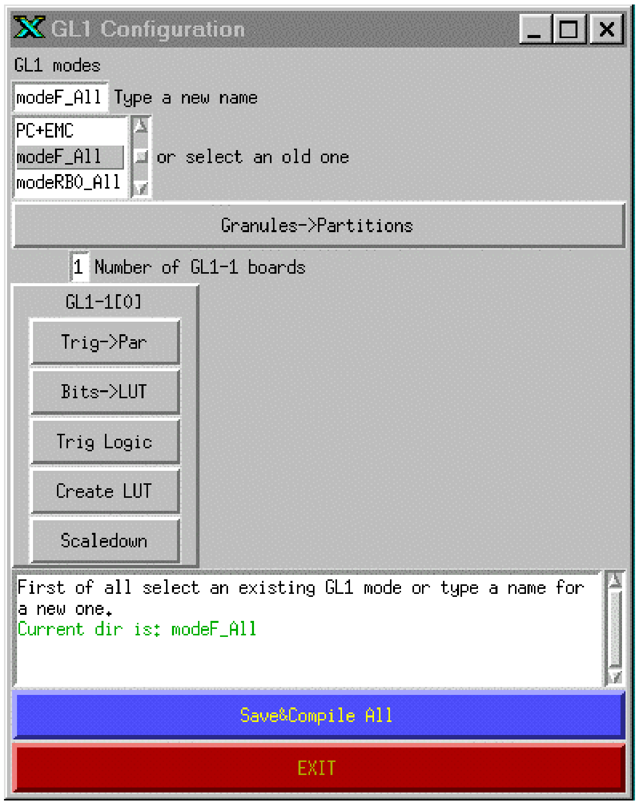
To start GL1 configuration GUI type: **GL1cfgC** - stands for GL1 configuration for CORBA version. You will see the next window:



First of all select an existing GL1 mode or type a name for a new one.

Figure 5. GL1cfgC GUI window as it appears at the very beginning.

As you can see all buttons (except EXIT) are disabled. To activate them choose a name for a GL1 mode. You can type a new name in the upper small self-expanded window or choose a name of already existed mode from the list box below. Figure 6 shows how GUI will look like after you have chosen the mode.



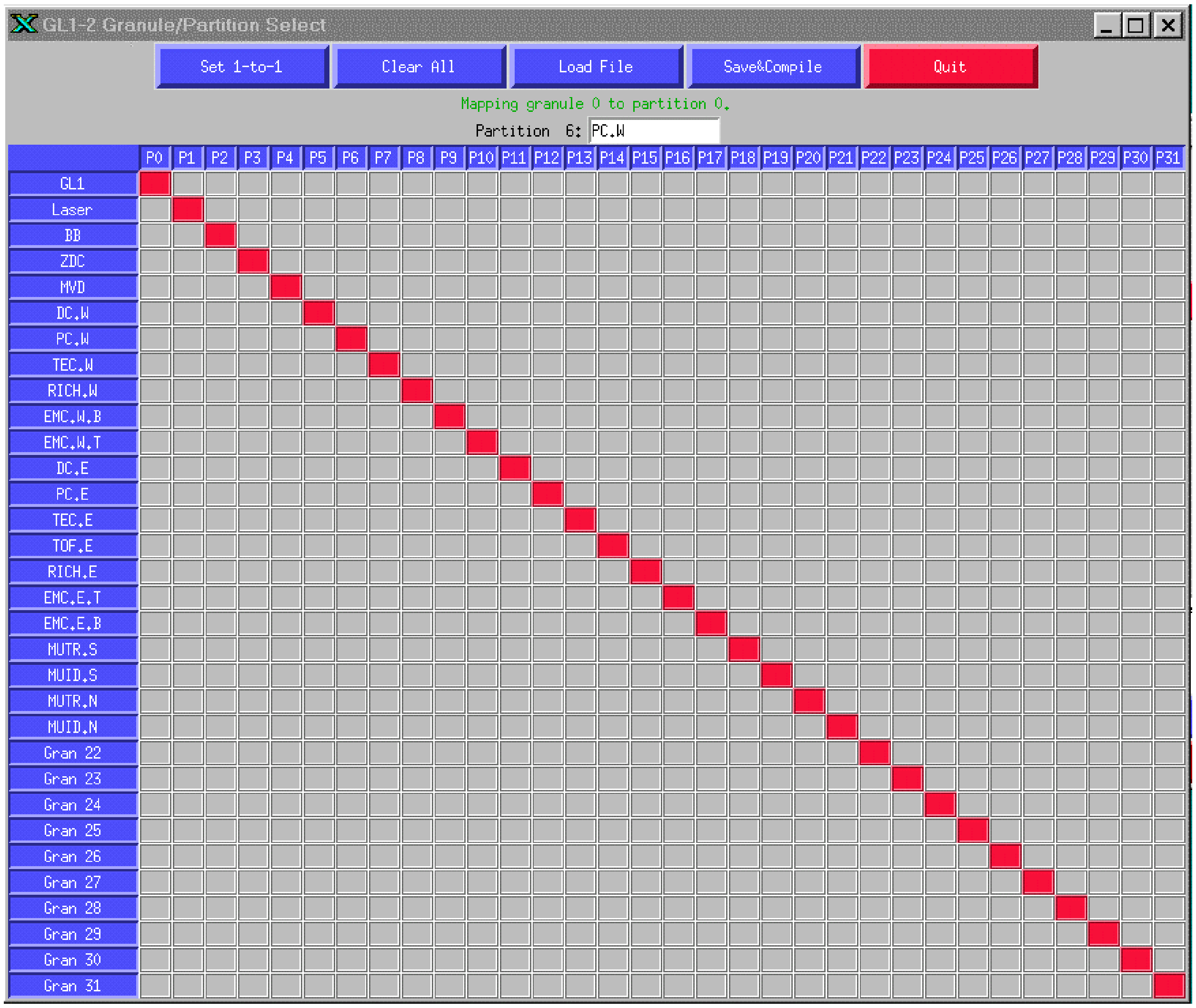
```
First of all select an existing GL1 mode or type a name for
a new one.
Current dir is: modeF_All
```

Figure 6. GL1 GUI window after mode modeF_All was chosen by user.

The name of GL1 mode is actually the name of the subdirectory located at $GL1_DATA (the same as $ONLINE_CONFIGURATION/GL1/) area. Each such subdirectory keeps 16 configuration files but you need know nothing about them – program will create and modify them itself.

In this document we use **modeF_All** configuration as an example. This mode has 32 partitions, each partition has one granule and one trigger. For simplicity partition number is the same as its granule and trigger number (1-to-1). Only Forced Accepts are used to configure triggers: forced accept of a granule N fires trigger number N that belongs to the partition N.

We will start by associating granules with partitions. Press the button "Granules-Partitions". You should be rewarded with a screen like that in Figure 7. This is a simple mapping program, with partition names (or 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 or by clicking on correct position (since granule can belong to only one partition the previous selection will be deselected). A message will appear under the command buttons confirming the mapping and unmapping actions.



Mapping granule 0 to partition 0.

Partition 6: PC.W

	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
GL1	🟥																															
Laser		🟥																														
BB			🟥																													
ZDC				🟥																												
MVD					🟥																											
DC.W						🟥																										
PC.W							🟥																									
TEC.W								🟥																								
RICH.W									🟥																							
EMC.W.B										🟥																						
EMC.W.T											🟥																					
DC.E												🟥																				
PC.E													🟥																			
TEC.E														🟥																		
TOF.E															🟥																	
RICH.E																🟥																
EMC.E.T																	🟥															
EMC.E.B																		🟥														
MUTR.S																			🟥													
MUID.S																				🟥												
MUTR.N																					🟥											
MUID.N																						🟥										
Gran 22																							🟥									
Gran 23																								🟥								
Gran 24																									🟥							
Gran 25																										🟥						
Gran 26																											🟥					
Gran 27																												🟥				
Gran 28																													🟥			
Gran 29																														🟥		
Gran 30																															🟥	
Gran 31																																🟥

Figure 7: The granule to partition mapping configuration window. Mode modeF_All mapping is shown.

At this point a few important comments are in order. First of all, the granule names and their assignment to granule numbers are contained in the file **.gl1_granule_names** in the $GL1_MAIN/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. An example is listed below:

0. GL1
1. Laser
2. BB
3. ZDC
4. MVD
5. DC.W
6. PC.W
7. TEC.W

8. RICH.W

…

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 configuration program for display purposes only, 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!

You can change a partition name: click the corresponding name in the Partition Names row and just edit its name in the entry above. If you move mouse over partition box you will see its name in the entry. On fig. 7 the name of partition 6 is displayed.

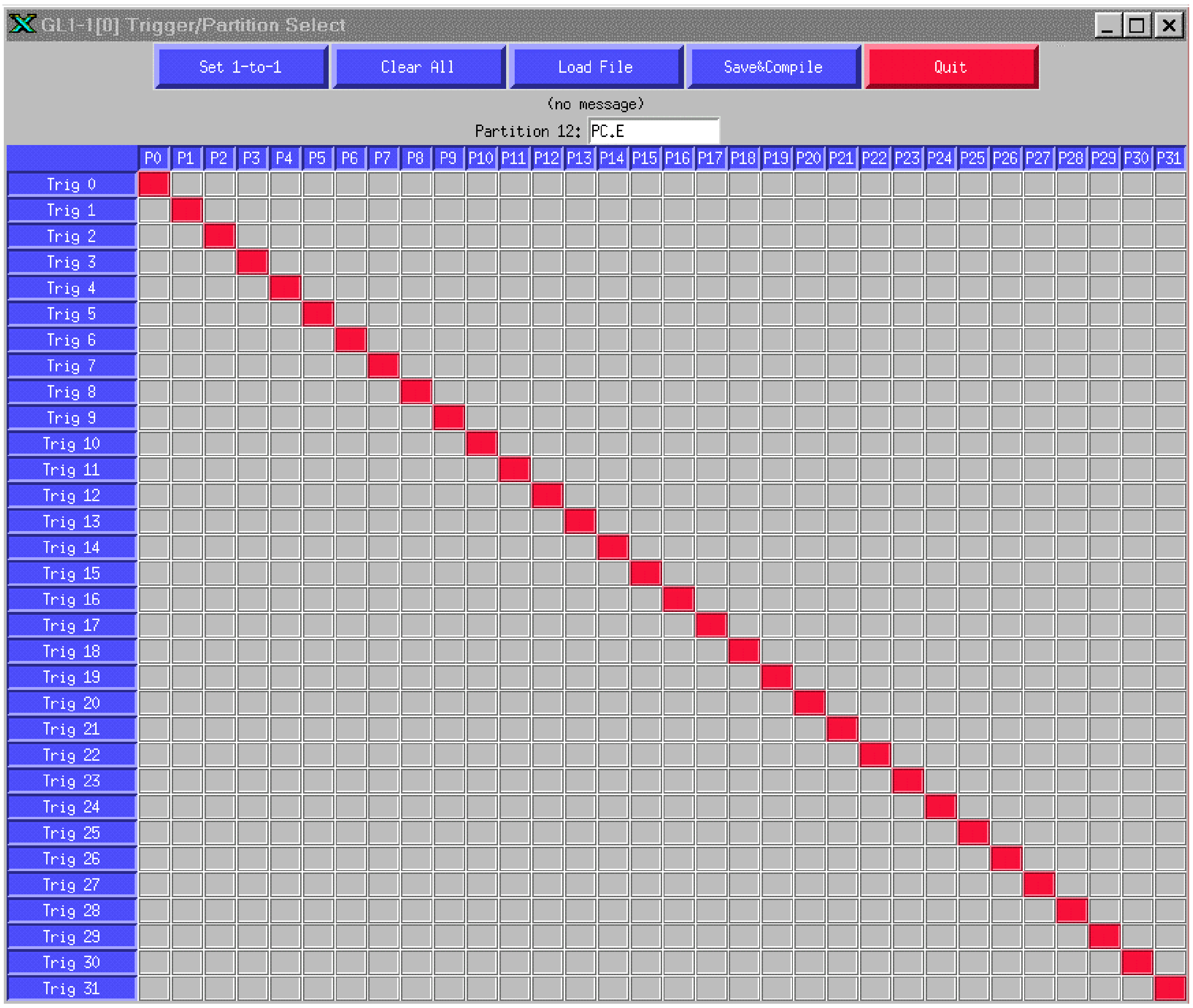
When you hit **Save&Compile** button all 32 partition names will be saved into the file named **partition.names** located in current mode subdirectory. These names will be used later by Partition Server and by Run Control to describe partitions of current GL1 configuration.

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&Compile** button at the top of the grid will allow you to save your configuration file and to create crossbar files that can be downloaded into the GL1 boards. 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.

You can load Granules-Partitions mapping from other GL1 mode by pressing **Load** button. You must click **Save&Compile** button to create corresponding configuration files in your mode configuration!

Pay attention to "**GL1 Configuration**" window: before you started granules mapping the "**Granules-Partitions**"button was gray; when you made any changes to the mapping the button becomes red; when you saved & compiled your mapping the button becomes green. This is the policy for all buttons of "**Granules-Partitions**" window, it indicates the status of each operation.

The next step is to assign triggers to partitions. Clicking button "**Trig-Par**" of any GL1-1 button group will produce the configuration screen shown in Figure 8.



(no message)

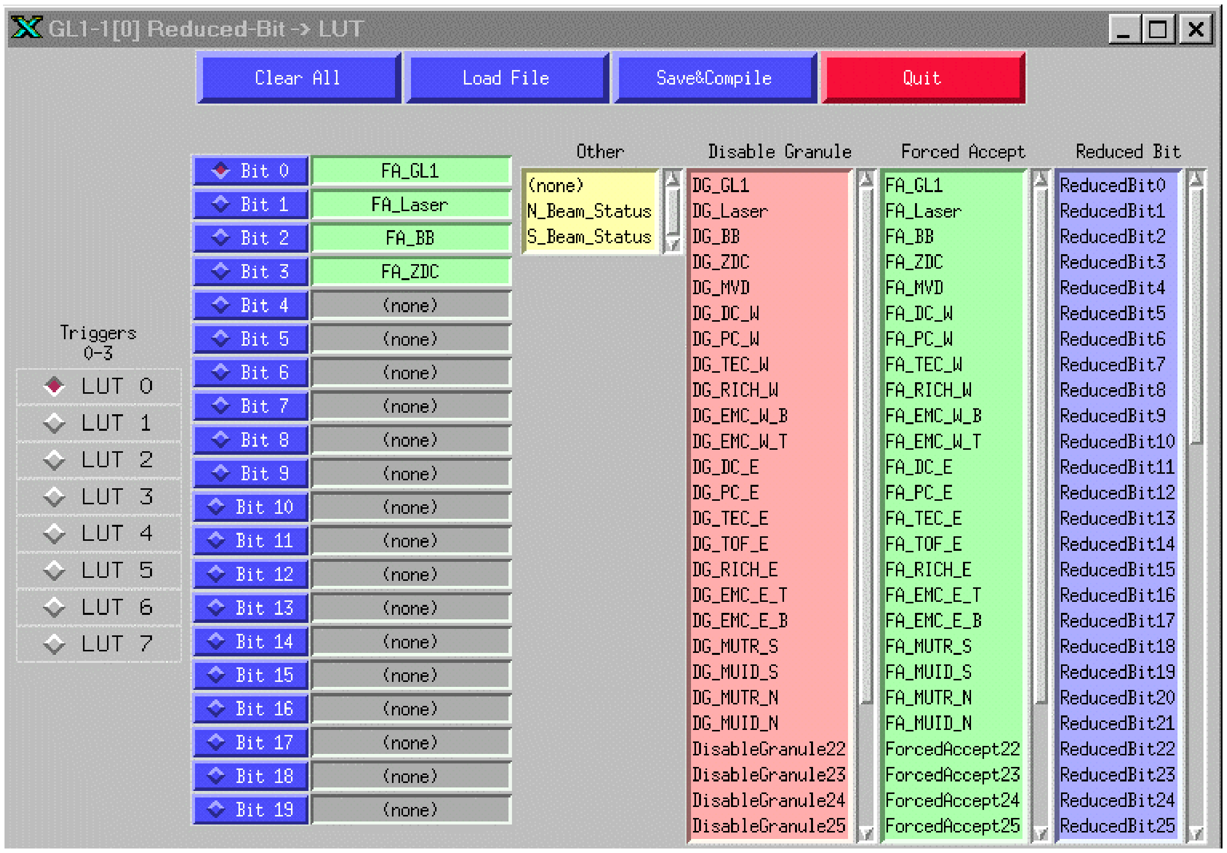
Partition 12: PC.E

	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
Trig 0	🟥																															
Trig 1		🟥																														
Trig 2			🟥																													
Trig 3				🟥																												
Trig 4					🟥																											
Trig 5						🟥																										
Trig 6							🟥																									
Trig 7								🟥																								
Trig 8									🟥																							
Trig 9										🟥																						
Trig 10											🟥																					
Trig 11												🟥																				
Trig 12													🟥																			
Trig 13														🟥																		
Trig 14															🟥																	
Trig 15																🟥																
Trig 16																	🟥															
Trig 17																		🟥														
Trig 18																			🟥													
Trig 19																				🟥												
Trig 20																					🟥											
Trig 21																						🟥										
Trig 22																							🟥									
Trig 23																								🟥								
Trig 24																									🟥							
Trig 25																										🟥						
Trig 26																											🟥					
Trig 27																												🟥				
Trig 28																													🟥			
Trig 29																														🟥		
Trig 30																															🟥	
Trig 31																																🟥

Figure 8: The trigger to partition mapping program. The interface is similar to that used for the granule to partition mapping program. Configuration for modeF_All is shown.

Since the interface for this program is almost identical to the "**Granules-Partitions**" 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 <u>one and only one</u> partition, and the program will prevent mapping a trigger to more than one partition.

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 pressing "**Bits-LUT**" button. You should see a configuration screen like that shown in Figure 9.



Bit 0	FA_GL1
Bit 1	FA_Laser
Bit 2	FA_BB
Bit 3	FA_ZDC
Bit 4	(none)
Bit 5	(none)
Bit 6	(none)
Bit 7	(none)
Bit 8	(none)
Bit 9	(none)
Bit 10	(none)
Bit 11	(none)
Bit 12	(none)
Bit 13	(none)
Bit 14	(none)
Bit 15	(none)
Bit 16	(none)
Bit 17	(none)
Bit 18	(none)
Bit 19	(none)

Figure 9: 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. ModeF_All configuration for LUT 0 is shown as an example.

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 any of 4 list boxes at the right. As you can see some signals of "Forced Accept" and "Disable Granule" have names that correspond to granule name. If you want to clear some bit assignment first click this bit then click <none string in "Other" list box. Start from the lowest bits in the address. When you "connect" the selected bit the next bit will be automatically selected, so just click bit names one by one. Don't worry about uncommitted address bits; the program will automatically force those bits low when the crossbar files will be compiled.

Note: try always to start from Bit 0 and do not skip bits – it will significantly reduce LUT creation time.

When you have fulfilled all necessary connections press button "**Save&Compile**". Program will save your Bits-LUT configuration, create and compile some crossbar files, and write for you a skeleton for raw triggers firing logic.

Up to this point, configuring the GL1 has been a purely graphical exercise. 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, click "**Trig Logic**" button – it should open the skeleton of logic file in an **emacs** window, and the file it contains should look something like this:

```
LUT 0 {#define ForcedAccept0_bit0 0x1#define ForcedAccept1_bit1 0x2#define
FA_BBC_bit2 0x4

#define FA_ZDC_bit3 0x8

#define Others 0xffff0

TRIG 0 {

Others.eq.0

}

TRIG 1 {

Others.eq.0

}

TRIG 2 {

Others.eq.0

}

TRIG 3 {

Others.eq.0

}

}

LUT 1 {

#define FA_MVD_bit0 0x1

#define FA_DC_W_bit1 0x2

#define FA_PC_W_bit2 0x4

#define FA_TEC_W_bit3 0x8

#define Others 0xffff0

TRIG 4 {

Others.eq.0

}

…
```

(The listing should include all 8 LUTs and 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 that start with a LUT name (LUT 0) and are surrounded with braces {}. Inside LUT section Bit-LUT program defines named masks on the bits connected to this LUT address and one mask "Others" for each LUT that defines all disconnected bits. Names of the masks are the same as in Bit-LUT list boxes except additional suffix _bitX is added where X indicates to what bit is this signal connected. This is done just to remind you Bit-LUT configuration and to avoid ambiguity in case when you connect the same signal to more then one input bit.

You may be already have noticed that in addition to mask definitions inside each LUT section there are four TRIG sections also surrounded with braces {}. This is up to you to determine logic that will fire each trigger.

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 ".LT." (less than or equal to). Program is case insensitive to comparison keywords, so you can write .gt. as well as .GT. But program **is** case sensitive to mask names, so be careful while typing them (better just use Copy and Paste).

All lines in TRIG section are combined by logical AND, so for example

`TRIG 2 { FA_BBC_bit2.gt.0 Others.EQ.0 }`means: fire trigger 2 when Forced Accept from granule BBC is high AND disconnected entries (mask Others) are low.

The following is a fragment of logic file for modeF_All configuration.

```
LUT 0 {

#define ForcedAccept0_bit0 0x1

#define ForcedAccept1_bit1 0x2

#define FA_BBC_bit2 0x4

#define FA_ZDC_bit3 0x8

#define Others 0xffff0

TRIG 0 {

ForcedAccept0_bit0.gt.0

Others.EQ.0

}

TRIG 1 {

ForcedAccept1_bit1.gt.0

Others.EQ.0

}

TRIG 2 {

FA_BBC_bit2.gt.0

Others.EQ.0

}

TRIG 3 {

FA_ZDC_bit3.gt.0

Others.EQ.0
}

}

LUT 1 {

#define FA_MVD_bit0 0x1

#define FA_DC_W_bit1 0x2

#define FA_PC_W_bit2 0x4

#define FA_TEC_W_bit3 0x8

#define Others 0xffff0

TRIG 4 {

FA_MVD_bit0.gt.0

Others.EQ.0

}

…
```

You can define your own mask if you wish. For example, suppose you want to fire trigger 0 if any of 4 LUT input bits has signal. You can add to mask definitions:

```
LUT 0 {#define ForcedAccept0_bit0 0x1#define ForcedAccept1_bit1 0x2

#define FA_BBC_bit2 0x4

#define FA_ZDC_bit3 0x8

#define Others 0xffff0

#define Any_of_4 0xf
```

and define trigger 0 logic as:

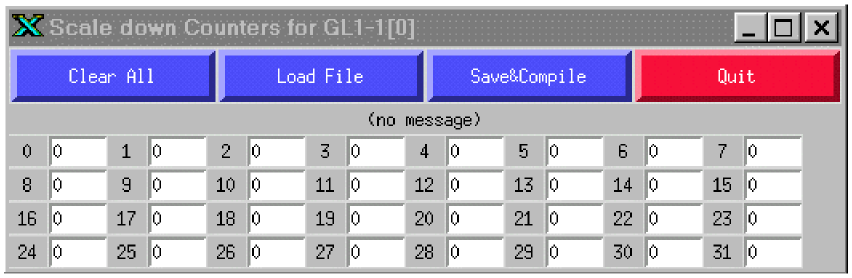
```
TRIG 0 { Any_of_4.gt.0 Others.EQ.0 }
```

Thus by combining mask bits and by adding new lines in trigger section you can create sophisticated logic for trigger firing.

You will definitely say: "Hey, guys! Your code is half C-like half Fortran-like! Can it be only in one language?" Calm down, it will be completely C-like later.

OK, you modified the trigger logic skeleton file as you desired. What's next? First, save file just by choosing "**Save Buffer**" from **File** menu of emacs. Then exit emacs. ***Note: you can't continue to use GL1cfgC program until you close this emacs session***. You will see that "**Trig Logic**" button becomes green while "**Create LUT**" is now red. Push this red button and wait while program will create a Look Up Table. If program finds any error in your logic file it will stop compilation and will write error message in GL1cfgC message window. The text color will be red. If no errors are found the program will create and write down the LUT. The size of LUT file is 1M strings, so it will take some time to do it. If everything is OK program will type "Done!" in message window and will convert the color of "**Create LUT**" button to green.

Now the last button of GL1 configuration GUI: "**Scaledown**". It permits you to define scale down counters for each trigger. We definitely will not use it in Year-1 run, so just keep all values equal to 0.



(no message)

0	0	1	0	2	0	3	0	4	0	5	0	6	0	7	0
8	0	9	0	10	0	11	0	12	0	13	0	14	0	15	0
16	0	17	0	18	0	19	0	20	0	21	0	22	0	23	0
24	0	25	0	26	0	27	0	28	0	29	0	30	0	31	0

Fig. 10. This window will pop up when you hit "Scaledown" button. You can set a scale down factor for each trigger of this GL1-1 board.

Congratulations! You configured GL1 for one GL1-1 board.

It is important to note that for the Year-1 run 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 four steps (**Trig-Par, Bit-LUT, Trig Logic, Create LUT,Scaledown**) for each GL1-1 board in the system.

Generally program defines the number of GL1-1 board itself reading file .GL1temlate in $GL1_DATA area. This file is template for GL1.config file that is created by GL1cfgC in each mode subdirectory. Only experts have right to update this file. Nevertheless you can change the number of GL1-1 boards yourself by typing the desired number into the entry labeled "Number of GL1-1 Boards". If you increase the number of boards additional blocks of GL1-1 buttons will appear, if you decrease this number the rightmost blocks will disappear. Program will make itself the corresponding changes to GL1.config file.

Button "**Save&Compile All**" will save and compile **Granules-Partitions** and **Trig-Par, Bit-LUT** configurations for each GL1-1 if any of them was changed, but you must make steps **Trig Logic** and **Create LUT** yourself for each GL1-1. You better use **Save&Compile** button on each panel.

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.

Back to Index

How to download a GL1 configuration mode

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

To start the gui, change directory to the GL1 area and do the following:

- phoncs>**cd $GL1_MAIN/java**
- java>**gl1gui**

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 11. 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.

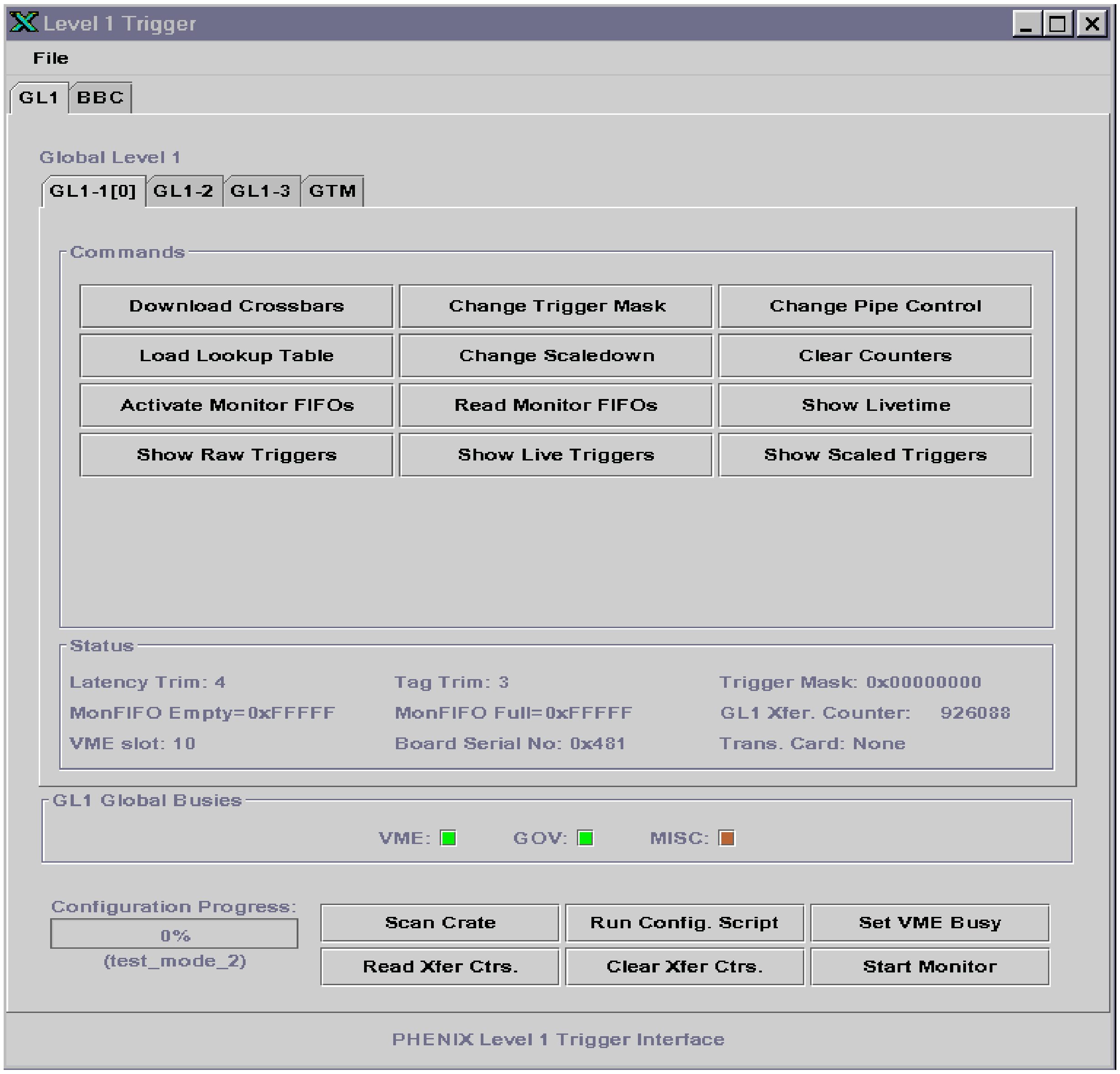
Note: If instead of GUI you see on your terminal the message

"Exception during ResolveName

Object – gl1. – not found."

then just reset the GL1 PPC controller.

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.



Latency Trim: 4
Tag Trim: 3
Trigger Mask: 0x00000000
MonFIFO Empty=0xFFFFF
MonFIFO Full=0xFFFFF
GL1 Xfer. Counter: 926088
VME slot: 10
Board Serial No: 0x481
Trans. Card: None

Figure 11: 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 Config. Script"** button. You will be presented with a file box that you can use to select the **GL1.config** file from the directory of 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. **Configuration Progress** bar will show you the downloading progress. If any error occur during the downloading the message window with problem explanation will appear on the screen.

If "**Configuration Progress"** bar has reached 100% without any error message – congratulations, your GL1 mode has been successfully downloaded!

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 VME Busy"** button on the GL1 gui (right bottom corner). 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.

The GL1-1, GL1-2, GL1-3 and BBC 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 Gran. Counters"** function (which allows you to clear selected granule accept counters in GL1-3).

We will describe later in the chapter for advanced user how to use all these and other buttons. For an ordinary person on shift you already know enough to create and download GL1 configuration mode. Good luck!