European Laboratory for Particle Physics
Laboratoire Européen pour la Physique des Particules
CH-1211 Genève 23 - Suisse

FRL

FRL software

Document Version: beta 0.1a
Document Date: 3 September 2003
Document Status: draft
Document Author: Dominique Gigi, Dirk SAMYN

Abstract

This document describes the software aspects for the FRL test setup.

This document is still **preliminary**.

1 Introduction

In the CMS DAQ, the Frontend Readout Links (FRL) collect the data from the FED and send them to the FED Builder through the Myrinet interface cards.

2 Test Setup

2.1 Introduction

To be as complete as possible, four areas in the FRL must be tested:

- the PCI bus (bits and protocol signals),
- the input (one link at the time, FIFO, LVDS, ...),
- the ZBT memory in spy mode (data, address, control, ...),
- JTGA chain (using the reprogramming through PCI with the JBC program).

2.2 Crate Layout

16 cards will be put in a crate and tested at the same time for each input link.

2.3 FRL Layout

Figure 1 Test FRL Layout and **Figure 2 Test FRL Block Diagram** describe the layout of a FRL in the test setup.

The Myrinet emulator is a GIII programmed to perform 2 actions:

- according to the event parameters loaded in the SDRAM memory (2 x 64-bit words per event), the GIII sends events to the LVDS link (using the standard SLINK64 protocol). It will also be able to set the TEST signal.
- the GIII receives events through the PCI bus (PCI or PCI-x) and checks the data and header of the packets. (An event is sent to Myrinet in packets of programmable size with a header in front of it).

Two words are used to describe an event:

- Word 1:
    - bits [23..0] Event length (payload)
    - bits [43..32] Source (FED #)
    - bits [55..48] SEED (for data pseudo random generator)
- Word 2:
    - bits [11..0] Bunch crossing number
    - bits [12..31] time between end of the event and the next event (multiple of 100ns)
    - bits [55..32] Event number
    - bit [63] Generate an Event# error (not used in this configuration)

Free bits can be used for future implementation.

The SEED number implemented in the Header of the event is used at the receiving part to recreate the data and to compare them with the real event data.

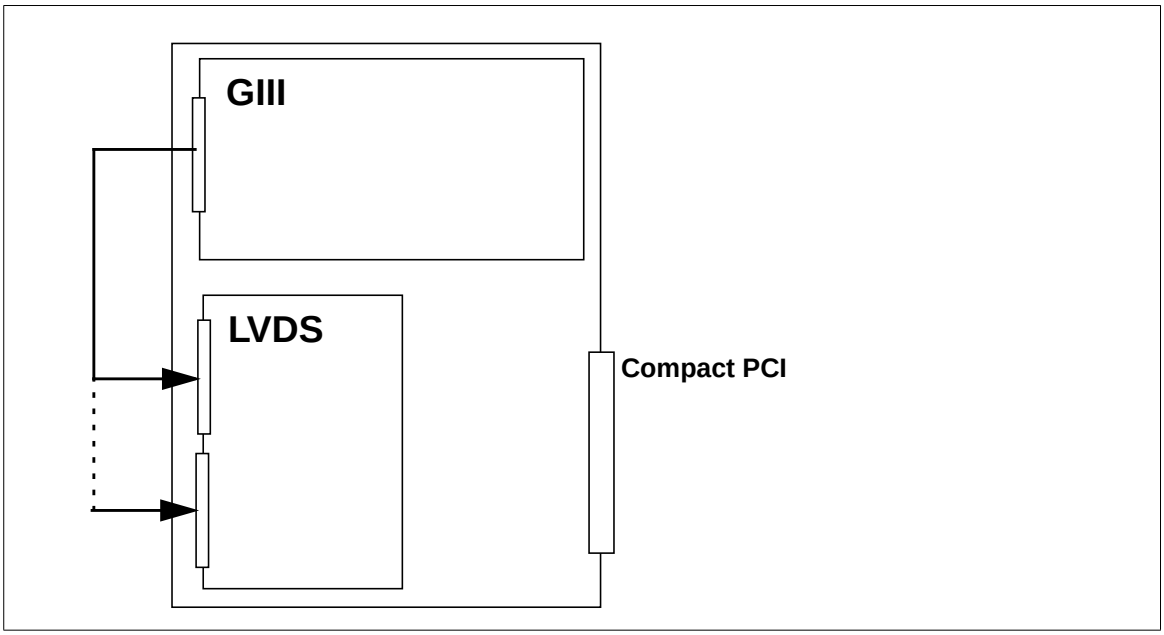

Figure 1 Test FRL Layout

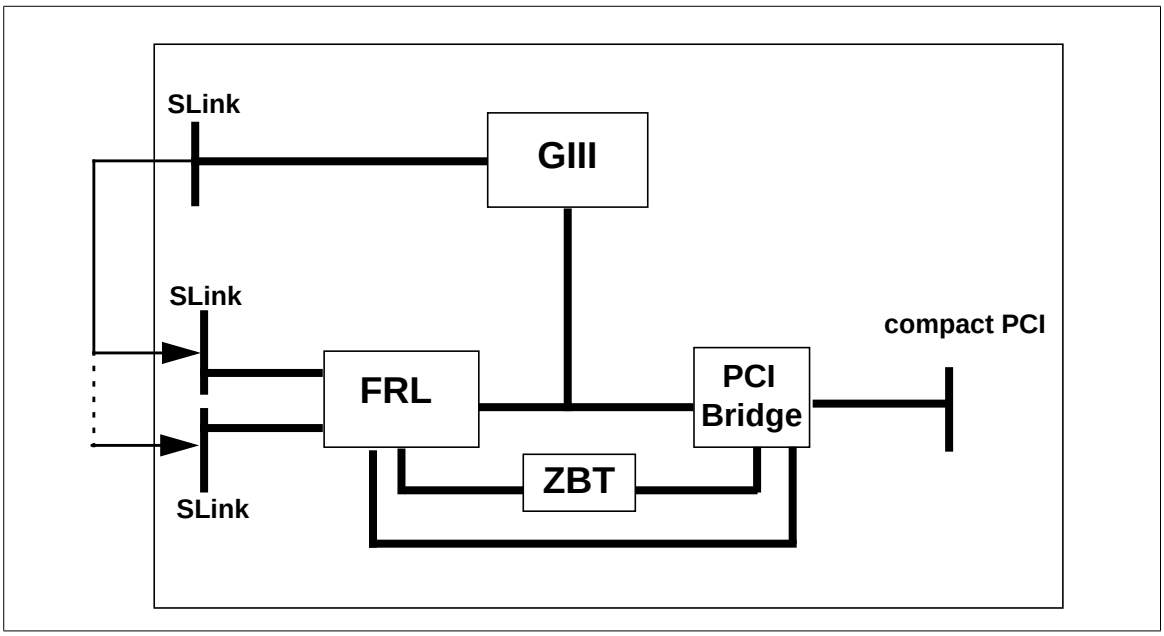

Figure 2 Test FRL Block Diagram

3 FRL Offsets

3.1 FRL Registers offsets (data swapped)

Table 1 FRL Registers description

Mnemonics	Register description	Bit	Mode	BA0 Offset
RESET	Software Reset	17	W	00h
ENABLE LINK0	Enable link 0	24	R/W	00h
ENABLE LINK1	Enable link 1	25	R/W	00h
ENABLE LINK2	Enable link 2	26	R/W	00h
ENABLE LINK3	Enable link 3	27	R/W	00h

Table 1 FRL Registers description

Mnemonics	Register description	Bit	Mode	BA0 Offset
SPY ALL EVENTS	Spy all events (acquisition mode through PC)	30	R/W	00h
ENABLE SPY	Enable spy mode (has to be done after the load of events to spy)	31	R/W	00h
LD_B_ADD	Load the data block address (done by Myrinet)	31..0	W	04h
LD_BS	Load the data block size (bytes)	31..0	R/W	80h
LD_HS	Load the FRL header size (number of 32-bits words) (usually 0x06)	31..0	R/W	84h
LD_EVT_SPY	Load the event number to spy (max. 1024 values)	23..0	W	88h

3.2 GIII Registers offsets (data swapped)

Table 2 GIII Registers description

Mnemonics	Register description	Bit	Mode	BA0 Offset
RESET	Software Reset	17	W	00h
STEP MODE	Enable the step mode (send one event at the time)	8	R/W	00h
AUTO EVT#	Generate event with a counter (usually set to 0)	9	R/W	00h
START	Start event generation	10	R/W	00h

Table 2 GIII Registers description

Mnemonics	Register description	Bit	Mode	BA0 Offset
STEP_ACC	Send one event (Bit 8 must be set)	11	W	00h
LD_MASK	Set the number of events prepared in SDRAM memory (0,1,3,7,F,1F,...)	31..0	R / W	04h
LD_A_PFF	Equivalent to FRL LD_B_ADD	31..0	R / W	08h
LD_BS	Load the data block size (bytes)	31..0	R / W	80h
ERR_DT	Number of Errors in data	31..0	R	40h
ERR_HD	Number of Errors in header	31..0	R	44h
ERR_BIT_L	Indiquates the wrong bit in data 31..0	31..0	R	48h
ERR_BIT_H	Indiquates the wrong bit in data 63..32	31..0	R	4Ch
ERR_CODE_A	Error Code (to be defined)	31..0	R	50h
ERR_CODE_B	Error Code (to be defined)	31..0	R	54h

3.3 ZBT Registers offsets

Table 3 ZBT Registers description

Mnemonics	Register description	Bit	Mode	BA0 Offset
RESET	Software Reset	17	W	00h
LD_ADD_B	Load the data block address (max. 64 blocks)	31..0	W	04h
LD_BS	Load the data block size (bytes)	31..0	R / W	80h
LD_ADD_FF	Load the FIFO base address for the Event WC	31..0	R / W	84h
LD_FF_PTR_A	Address to update the FIFO write ptr	31..0	W	88h

4 Test Software

4.1 Introduction

Since most of the checks are done by the hardware, the software is essentially reduced to

- bridges initialisation.
- spy events and check their structure.

In order to be flexible, a small set of commands has been defined to perform that functionnality.

4.2 Command Syntax

There are 4 types of commands:

- ctrl (controls commands)

- frl (commands for the FRL on all boards found in the crate)
- giii (commands for the GIII on all boards found in the crate)
- zbt (commands for thr ZBT on all boards found in the crate)

Notes:

- all displayed information is also logged into a file,
- comments start with a '#',
- comments can also be put at the end of line.

A value or offset can be defined as following:

- n (a decimal value)
- **0x**n (a hexadecimal value)
- **0w**n (a value defined by a bit position)
- **$**varname (a value defined by a variable)
- | (a value defined by a OR of several other values).

ctrl print config
prints the full config of all boards.

ctrl exec [command]
execute a command. The "Exit code" string is checked in the command output.

ctrl start nsecs spymode
start test for n seconds, spymode = 1 is spy enabled during test

ctrl echoln "a string"
ctrl echo "a string"
echo (with or without new line) a string

frl|giii|zbt set varname value
define a new variable and assigns a value to it

frl|giii|zbt write value offset
write a given value at location defined by offset

frl|giii|zbt display offset
display the value at location defined by offset

frl|giii|zbt assign varname offset
assign the value at location defined by offset to a variable

zbt setup
setup zbt structures for spy mode

4.3 Example

```
#######
# ZBT
#######
#
# OFFSETs
#
zbt set RESET_OFF 0x0
zbt set LD_OFF_1 0x04
zbt set LD_OFF_2 0x80
zbt set LD_OFF_3 0x84
zbt set LD_OFF_4 0x88
#
# FUNCTIONs
#
zbt set RESET 0w17               Software reset
zbt set LD_B_ADD 0x00000000      address data block (max 64 blocks)
zbt set LD_BS 0x00000000         data block size (bytes)
zbt set LD_ADD_FF 0x00000000     FIFO base address for Event WC
zbt set LD_FF_PTR_A 0x00000000   addr to update the FIFO write pointer
#
# PROG
#
zbt setup
ctrl echo "ZBT_blk_sz=”          display zbt block size
zbt display $LD_OFF_2
ctrl echo ",ZBT_fifo_badd=”      display zbt fifo base addr
zbt display $LD_OFF_3
ctrl echo ",ZBT_fifo_wptr=”      display zbt fifo write ptr
zbt display $LD_OFF_4
ctrl echoln ""
```

4.4 Spy mode

Figure 3 Spy Block Diagram shows the spy buffer layout.

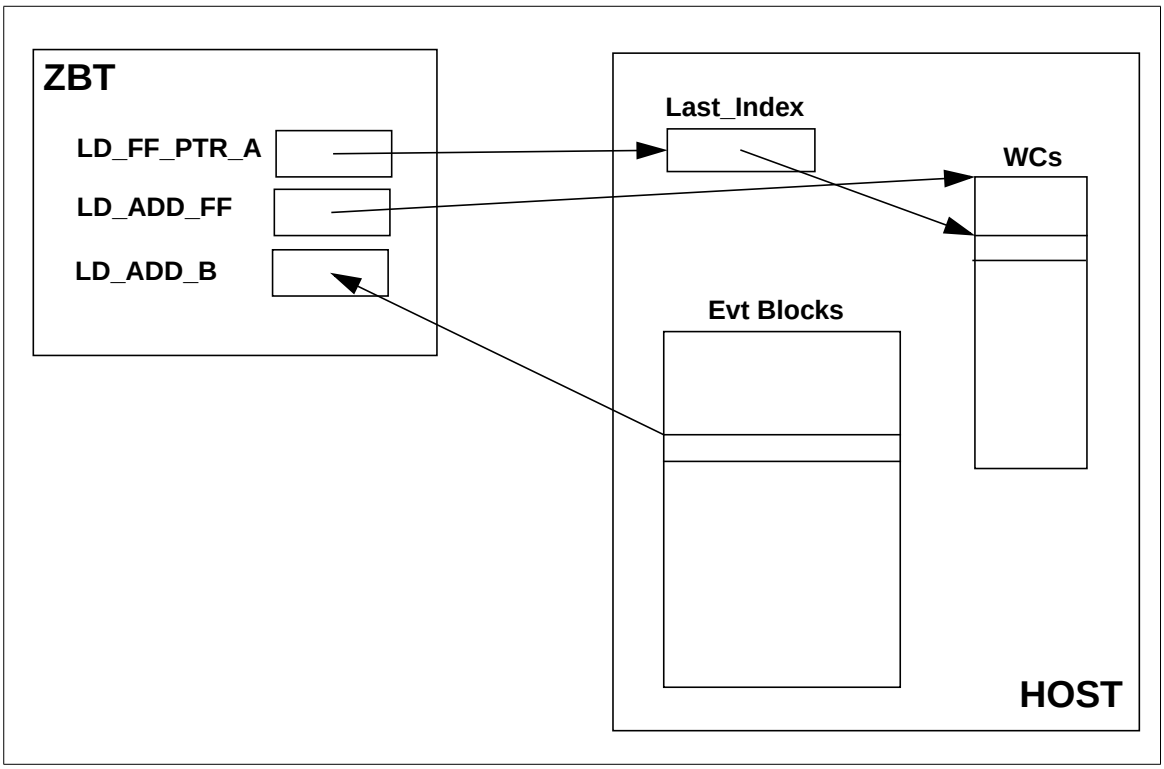

Figure 3 Spy Block Diagram

- the address of a 1024 WC table is loaded in the ZBT LD_ADD_FF register.
- the address of the Last Index location is loaded in the ZBT LD_FF_PTR_A register. That location is updated by the ZBT each time new WCs have been put in the WC table.
- the free Event Blocks pointers are put in the ZBT LD_ADD_B fifo.

4.4.1 Event structure