Available on CMS information server **CMS IN 2007/025**

CMS Internal Note

The content of this note is intended for CMS internal use and distribution only

19 May 2007

Putting Luminosity and Trigger Scaler information into the Event Stream

W. Badgett, K. Maeshima

Fermilab, Batavia, Illionis, USA

V. Halyo, D. Marlow

Princeton, New Jersey, USA

Abstract

We propose putting the luminosity and some basic L1 trigger scaler information into the CMS event data stream in real-time. This proposal is not meant to replace the current existing design, but as an addition to enter the information into the event stream. The purpose is to support efficient and robust operation for data taking, handling and analysis by having these data readily and widely available. If we can achieve this with small cost at this stage (with no major hardware nor software modifications), the gain of having this information in the event stream would be significant. We have explored various options and we conclude that there is a relatively simple way to implementing it. We discuss the motivation, information contents, and a method we propose with some technical details.

1 Introduction

This write-up proposes putting the luminosity and some basic L1 trigger scaler information into the CMS event data stream in real-time and up-stream of the HLT. This is an addition and not meant to replace the current read-out, monitoring, information storage scheme. In the hadron collider environment, due to the high cross section and the high crossing rate, trigger, DAQ and data-handling operations are non-trivial. The ease of commissioning, debugging and maintaining the high quality performance of CMS as the accelerator delivers higher and higher luminosity will be assisted by having more places we can probe/monitor and obtain relevant information. This document consists of three main sections (section 2, 3 and 4) following the introduction. Section 2 describess motivation, Section 3 describes data-contents we wish to enter into the event data stream, and Section 4 describes the technical method we propose to get the asynchronous data into the data stream.

2 Motivation

This section discusses advantage of having luminosity (lumi) and level-1 (L1) trigger scaler information in the event stream. Both lumi and L1 trigger scaler information are essential for the experiment and are used in various places. Table 1 and Table 2 list examples of where this information would be used.

Table 1: Example of where/how lumi information may be used

type	what	comments
delivered lumi	inst.and integ. lumi to LHC and CMS (bunch by bunch)	critical LHC performance measure (independent readout from CMS DAQ)
live lumi	integ lumi when DAQ up	a) integ. live lumi and live-lumi/delivered-lumi are critical CMS performance measure which should be available in real time (need Live-Time info. from trig. system) b) need integ lumi for each trigger and data sets for analysis
	inst. lumi (average over bunches)	used in multiple places a) trigger x-section monitors/studies b) detector occupancy monitors/studies c) dependency studies for online/offline operations(event-size, cpu, stream-size etc.) d) dependency study for physics analysis (e.g. isolation efficiencies)

Table 2: Example of where/how L1 trigger scaler information may be used for monitoring, debugging and analysis

what	comments
event counts	monitor to ensure we are not loosing events unknowingly by providing convenient probe points in different DAQ stages
Trigger Rates (pre/post pre-scale)	basic quantities to monitor. need this info. to determine dynamic prescale factors
Trigger x-Section	basic quantities to monitor. (need also inst. lumi info as well as trig rate info.)
Live time (dead time) info.	need to figure out live integ. lum
pre-scale factors	need to calculate effective trigger cross sections
trigger rates (x-sections) vs inst. lum.	behaviour of each trigger rates learned from this can be used as the reference numbers for the trigger rate (x-section) monitor These information are also essential to construct a new trigger table for future with higher lumi. (see section 2.3 for more details)

2.1 Current Design

With the current design, both HF luminosity and L1 trigger scaler information are not stored in the event stream. In the online environment, it is planned that processes which needs lumi data can obtain it either via a xDAQ server when needed in the frequency of an order of several Hz (maximum available rate is approx. 5 Hz, defined by the lumi Nibble which is 2**11 orbits corresponding around 180 msec), or by reading the luminosity database where the. information is stored in each lumi. section, A lumi section is defined as (2**20 orbits)= (2**20*88.924 micro-sec) = approx. 93.2 sec. L1 GT (Global Trigger) scaler information is available from the L1 Global Trigger system (GT) and the real time monitoring is planned at the Trigger Supervisor (TS) implemented as a xDAQ application. The TS also plans to store the trigger scaler information into the trigger database. The format of the data base is yet to be determined. A real time monitoring application can also obtain the L1 trigger scaler information via a server as the lumi information. In addition to performing trigger monitoring, the main functionality of the TS is to control/navigate the L1 trigger system by communicating with the DAQ system. For example, the TS will propagate the proper keys from the run configuration to load appropriate LUT (look up tables) to each of the L1 systems, using the online database.

2.2 Advantages - General

As listed in tables 1 and 2, we often use lumi and L1 trigger scaler information together, hence it makes sense to make effort to put both information into the event stream, not one or the other. Obviously having these data in the event stream does not help all possible cases. For example, delivered lumi must be provided when beam is delivered even when events were not taken. However, there are various advantages to have lumi and L1 trigger scaler information in the event steam. Generally speaking, the more places in the data chain where we can probe and monitor easily, the better, all without interfering with the existing data acquisition. One can think of various different spots in the data chain to inject the lumi/L1 scaler information and these methods are outlined in the final section of this note. It is our advantage that the earlier in the data chain that we inject the information, the more we gain access points for monitoring, and this means we can debug faster and run the system more efficiently. Indeed, this proposal does introduce a redundancy in recording the lumi and L1 information, but considering the centrality and critical importance of these data to the experiment, a measure of reduncancy is desirable. Also, injecting in real-time (before the HLT) allows us to enter time-varying information more frequently (1 Hz level maybe reasonable) compared to the accessing the information in the luminosity database (per lumi-section - 93 sec) in the later stage in offline.

The information only available at the crate level in the online system tends to be restricted only to limited number of experts so that operations are not impaired. Once information are available in the event data, it opens up more flexibility to look at the data, since there are many useful common tools developed to read and analise data (CMSSW , DQM framework, etc.) beyond the expert tools. Once the information enters the event stream, we can retrieve the information in the same manner (without changing the code) from the event data in online/quasi-online or offline environment. That is typically not the case for getting the information from the crates/server which is only available in real-time. Once the information is written in the database, it is persistent. However, frequency of the information stored in the database, differences (contents and timing) in online/offline database, and accessibility may limit usefulness in certain cases, and it should only help (not harm) to have alternative and redundant way of getting the information. We expect quasi-online/offline (nearly real time, but not quite) type of debugging to take place by reading raw data off the look-area disk, especially during the early running period. If the lumi and trigger scaler information are contained in the events stream, one can imagine accessing those information tends to be simplar while analysing those events. In such case, a program which is developed to debug/monitor the event data in the offline environment can be put into a use as an online monitor easily, and vice-versa.

2.3 Advantages - some specific examples

- XMon (trigger cross section monitor in depth): Trigger rates and cross sections can be monitored from the crate level information at TS in real time. However, in order to monitor the rates quantitatively we must know what are the expected rate and range for each of the triggers at a given inst. lumi. In principle, this sounds easy after some experience with real beam data taking. However, in practice, doing this (figuring out expected trigger rate/range for all the triggers as a function of inst. lumi and automate the trigger rate monitor using the obtained reference quantities) is rather involved. There are a large number of triggers to keep track of and one must know the details of trigger tables and their history. Since most of triggers do not behave an ideal manner (flat trigger cross section), fitting a function can be tricky, also. Despite the difficulties, knowing the expected trigger cross sections are very useful not only to monitor the trigger rate in real time but also to prepare a realistic new trigger table for the higher luminosity. The task of XMon is suited to be done in the online environment, in order to feed the results back into the scaler monitor as soon as possible. Having trigger scaler and lumi information in the event stream allows to run XMon in real time easily by sampling events as a consumer monitor off the storage manager using standard DQM structure.

- The occupancy monitoring is one of the most basic detector event monitoring. The expected occupancy plots (reference plots) may depend on inst. lumi, and the lumi. value can be obtained easily if the limi info. is in the event stream, whether the monitoring program runs in online, quasi-online or offline.
- In the normal situation, the inst. luminosity should be decreasing smoothly, and we should be able to predict the behaviour within a lumi section (93 sec) very well, if we know the beginning and the ending inst. lumi of the section. However, we may have a situation; something strange in the detector happens, and we might want to see if it is correlated with sudden change in the inst. luminosity. (unexpected accel. magnet setting changes, etc.....). Having more fine grain (max rate of 5 Hz) lumi info. would give an extra handle to monitor trigger/detector online/offline.
- Prescale factors, trigger efficiencies, integ. lumi of the data sets, etc are all essential numbers for analysis. Having simpler and easier way to access those information should help performaing analysis. Having redundant ways of getting some of the information allows to perform crucial cross checks often times.
- There are many operational issues on online and offline which are sensitive to the inst. lumi. (for example, trigger rates, HLT exe. cpu time per events, reconstruction cpu time, data rate, data size). Studies for those parameters as a function of inst. lum will provide useful information to prepare for the higher luminosity period and easier access to this information should only help.
- Having the lumi info. in the event stream should also help to monitor the HF lumi monitoring system in online and offline. With the current scheme, the persistent data is only in the lumi DB every 93 seconds and the faster rate lumi monitoring using DQM tools relies on the real time server. Having the lumi information in the event stream allows to develop/debug monitoring code easier since it is a bit more transparent whether one is in the online or offline environment.

3 Proposed Data Format

The data format definition is straightforward, as the data are fixed length. In the case of sending data for every level one accept, we should keep in mind the 2 kByte limitation. We do not propose sending luminosity information on a bunch by bunch basis for every event,as this would exceed our allowed bandwidth. We leave open the possibility of a low rate readout of the full bunch by bunch data.

3.1 HF Luminosity Data

In table 3 we list the 32 bit data words and the proposed contents for the luminosity numbers provided by the HF luminosity calculation PC. We include both a fast time integration of a few seconds and a slow time integration of one luminosity section (approximately 93 seconds). The luminosity section numbers will be close to the official accounting for luminosity delivered to CMS. However, pathological beam behaviour and system malfunctions may be missed by such a long integration time. To keep track of the luminosity delivered by the LHC, we also include the integrated luminosity since the beginning of the fill. Note that the items 0, 1 and 2 are expected to represent integral numbers of orbits.

In the future, additional methods of calculating and normalizing the lumonisity will come online. We propose in this respect to adapt the data source as appropriate to obtain the best online luminosity numbers that we can at any given time.

3.2 Luminosity Normalization

There are several possible methods for providing the absolute luminosity normalization, as have been discussed elsewhere [2]. Each method can be appropriate for specific ranges of instantaneous luminosity. However, we do not expect such items such as trigger cross sections to be known to better than 15%, so online uses of the luminosity measurements are not so sensitive to the normalization method. We propose that the normalization method used

Table 3: Proposed Luminosity Event Stream Data

Word	Quantity	Δt time interval
0	Instantaneous Luminosity	short, seconds
1	Starting orbit number	for item 0
2	Ending orbit number	for item 0
3	Average Instantaneous Luminosity	previous lumi section
4	Integrated Luminosity	previous lumi section
5	Luminosity Section number	for items 3 and 4
6	Integrated Luminosity	since start of CMS run
7	Integrated Luminosity	since start of LHC fill

to compute the absolute numbers in table 3 use the most appropriate method for the instantaneous luminosity at hand. The method used should be recorded in the run conditions database.

3.3 Global Trigger Data

The actual luminosity recorded by the experiment must be corrected by the livetime efficiency of the trigger and data acquisition system. CMS will not be live for all crossings containing colliding bunches; i.e., there will be some good crossings that cannot be triggered upon and read out. The Global Trigger keeps track of the deadtime accounting, as well as the trigger accept rates for the 128 level one triggers, after presacaling is applied. The Global Trigger counters are needed for measurements of the trigger rates and trigger cross sections, as described in the previous section.

Table 4 enumeration the proposed trigger scaler information that should be placed into the data stream. Note that none of these data can be placed directly into the data stream by the Global Trigger readout as the scalers do not have access to their Slink data output.

Table 4: Proposed Global Trigger Scalers Event Stream Data

Word	Quantity	Δt time interval
0...127	Trigger counters Δ_N	short, seconds
128...255	Trigger counters Δ_N	previous lumi section
256...383	Trigger counters Δ_N	since beginning of run
384	Starting orbit number	for items 0...127
385	Ending orbit number	for item 0...127
386	Luminosity Section number	for items 128...255
387	Crossings live	items 384, 385
388	Crossings live	previous lumi section
389,390	Crossings live, 64 bits	since beginning of run

The total proposed data for the combined Global Trigger and HF Luminosity data represents 1596 bytes, within the 2kB limit even considering that some header information must be added to the raw data.

4 Technical Implementation

Several methods to inject the luminosity data into the main data acquisition system have been considered. We first review them and explain why some methods are not favored:

- Use an HCal DCC module to inject data, with the source data arriving via the VME bus and stored in on-board FPGA. The DCC has both a TTC interface and an Slink data output; special use would require an FPGA reprogram. While this method would achieve the desired results, these boards are expensive, and there are currently not enough spares to allow the expansion. Also, a dedicated VME 9U crate would be needed in order to isolate this special DCC from interfering with the primary HCal data flow. DCC cards from other detector subcomponents would have the same issues.

- Use an LTC module to inject data, in a similar mode to the DCC solution. The LTC also has a TTC interface and an Slink data output. A dedicated VME 6U crate would be needed in this case. However, these boards are in very high demand and there are no spares that could be devoted to this purpose. Experts have also predicted great difficulty in reprogramming the LTC for this purpose.
- Using software to inject the data at the HLT Filter Unit, or at a nearby point. With potentially thousands of HLT nodes, this would represent an unacceptable burden to whatever communication mechanism is used. If the luminosity calculation PC were to transmit the information via xDAQ messages, this would represent thousands of TCP/IP connections to that PC; if the HLT were to contact the database server, a similar number of concurrent connections would be required. The database and the luminosity PC are central and critical single points of failure for the experiment, and we cannot risk such a scheme.
- Using software to inject the data at the Storage Manager. This has been covered in the previous section. The Storage Manager does not unpack the event data and so cannot insert data.
- Using software downstream of P5. This has also been covered in the previous section. After the data leave P5, it is too late for the online data consumers to access it.

The method we propose here was suggested by the CMD group as being an appropriate general solution for injecting asynchronous data into the DAQ stream. The GIII PCI card [1] was developed as a general purpose tool for the data acquisition system. The GIII can be placed in any commercial PC, and can provide an Slink data output via a PMC mezzanine card. The GIII has on-board 32Mbytes of memory, which is more than enough for this purpose.

The missing piece is an interface to the TTC system, which can be plugged into the GIII PMC connectors to provide clock and trigger signals. Good news is that the CMD group has started developing a mezzanine with a TTCrx for the GIII The TTCrx chip on this mezzanine board would provide the GIII with the synchronized trigger signals that would tell it to pipe the onboard data out the Slink to the FRL. See figure 1 for a rough diagram of how the data would flow. Once the data are in the FRL, they can be treated like data from any other source.

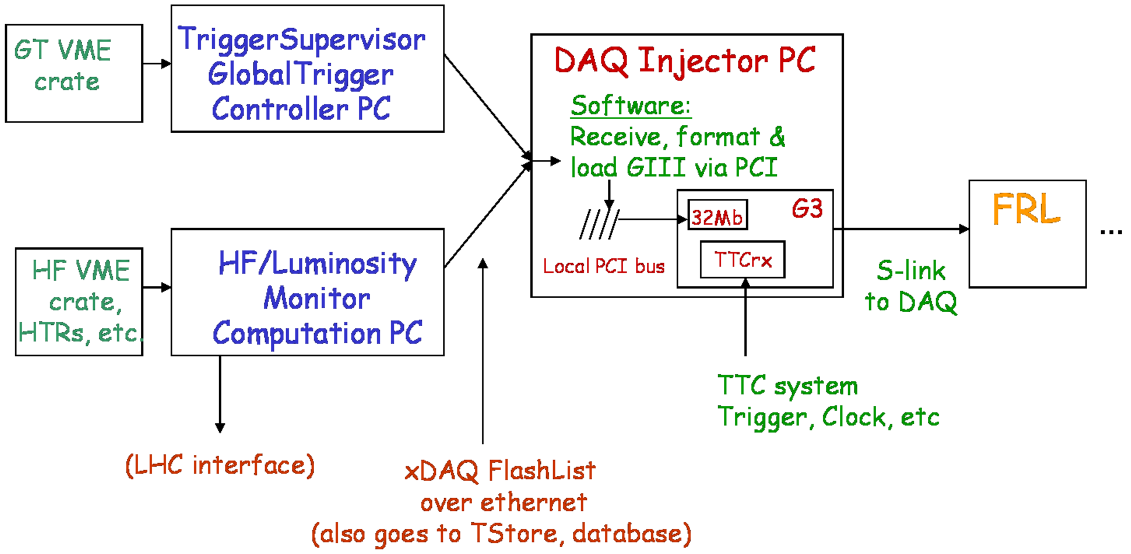

Figure 1: Proposed Luminosity Data Flow

Note that we have not specified the frequency of readout of the data. This system could send out data for every level one accept, or only on certain trigger types. With the TTCrx interface, we can distinguish between different types of triggers, and could consider placing the readout only on a special calibration type of trigger. However, the GIII must send a data record for each Level one accept, containing as a minimum an s-link header and trailer.

The FPGA of the GIII will have to be reprogrammed to provide this functionaliy. The Fermilab group has a number of engineers that would be able to code the new FPGA program. However, one serious concern is the schedule of the GIII TTCrx mezzanine card. To be most effective, this proposal should be implemented comfortably before the arrival of luminosity in late 2007. If needed, Fermilab may consider contributing engineering effort to the GIII TTCrx interface.

The expected latency of this system is below 1 second. Open issues include:

- the DAQ injector will belong to which TTC partition? to the trigger?, HF, a new one? (HF partition should just work)
- the DAQ injector fragments will belong to which super-fragment?
- software has to be developed to configure and control the DAQ injector PC. To which subsystem (in the run-control sense) does this belong?

Acknowledgements

Authors would like to thank all the people who provided us with useful inputs and allowed us to make this proposal. Bill and Kaori would especially like to thank: Dominique Gigi, Manfred Jeitler, Jeremiah Mans, Frans Meijers, Christoph Schwick, Westley Smith, Ichiro Suzuki, Chris Tully, and Claudia Wulz.

References

[1] For GIII information, see `http://dgigi.home.cern.ch/dgigi/fed_kit/fed_kit.htm` link.

[2] *CMS Lumonisotu, CMS Note in prepartion, V. Halyo, et al*