Set up and programming of an ALICE Time-Of-Flight trigger facility and software implementation for its Quality Assurance (QA) during LHC Run 2

Summer Student Report

Francesco Toschi

Alma Mater Studiorum, University of Bologna

Supervisor
Daniele De Gruttola

Co-supervisor
Andreas Morsch

23 September, 2016

Project Description

The Cosmic and Topology Trigger Module (CTTM) is the main component of a trigger based on the ALICE TOF detector. Taking advantage of the TOF fast response, this VME board implements the trigger logic and delivers several L0 trigger outputs, used since Run 1, to provide cosmic triggers and rare triggers in pp, p+Pb and Pb+Pb data taking. Due to TOF DCS architectural change of the PCs controlling the CTTM (from 32 bits to 64 bits) it is mandatory to upgrade the software related to the CTTM including the code programming the FPGA firmware. A dedicated CTTM board will be installed in a CERN lab (Meyrin site), with the aim of recreating the electronics chain of the TOF trigger, to get a comfortable porting of the code to the 64 bit environment. The project proposed to the summer student is the setting up of the CTTM and the porting of the software.

Moreover, in order to monitor the CTTM Trigger board during the real data taking, the implementation of a new Quality Assurance (QA) code is also crucial, together with the development of new macros that will allow to perform fast checks, such as comparing QA output for real data and Monte Carlo. This will provide a test tool for the new CTTM software and automatic, fast and efficient checks of the QA outputs for the ALICE Time-Of-Flight in the reconstruction of LHC Run 2 data, paying a particular attention to the plots dedicated to the TOF trigger monitoring.

ALICE TOF

ALICE (*A Large Ion Collider Experiment*) is one of the four main experiments at the Large Hadron Collider (LHC): it requires heavy ions collisions in order to study the properties of QGP (*Quark-Gluon Plasma*). Since Particle IDentification (PID) is essential for ALICE, the *Time-Of-Flight* detector (TOF) is crucial: it has a modular structure corresponding to 18 sectors (called supermodules) in φ each one composed by 5 modules in z direction. Each module contains 15 (central module) or 19 (outer modules) Multigap Resistive Plate Chambers (MRPCs) tilted by a certain angle in order to minimize the transversal path of the incident particles [1]. The MRPC detector is based on the same technology as RPC but the stack is diveded into gas gaps thanks to glass plates installed in the active volume. This allows to have smaller gaps where independent avalanches may form avoiding the risk of *streamer*. MRPCs developed and used by ALICE are 122×13 cm^2 10-gap double-stack strips with an active area of 120×7.4 cm^2 subdivided into two rows of 48 pads (3.7×2.5 cm^2) [2]. The total number of readout channels (pads) is 152 928 (three central modules in front of PHOton Spectrometer (PHOS) are not installed to reduce the amount of material crossed by photons).

The signal coming out from 24 pads is sent to a Front-End ASIC (FEA) equipped with 3 NINO ASIC chips as front-end electronics [3]: signals are sent to read-out boards TDC Readout Module (TRD) which host 30 High Performance TDC (HPTDC) for a total of 240 channels each TRM. A FEA performs the logic OR of the 24 pads and the logic signal is OR-summed with another daisy-chained FEA: the signals from 6 daisy-chained couples of FEAs are collected by a FEA Controller (FEAC), which monitors FEA low voltage, temperature and it sets the thresholds. A Local Trigger Module (LTM) board receives the signals from 8 FEACs for a total of 48 ORs: there are 4 LTMs for each supermodules for a total of 72 LTMs.

The read-out electronics is hosted by four crates on the edge of each supermodule (two on A side and two on C side) and they are composed as follows:

Slot #	**Left Crate**	**Right Crate**
1	DRM	DRM
2	LTM	LTM
3	TRM	CPDM
4	TRM	TRM
⋮	⋮	⋮
12	TRM	TRM

The outputs from the 72 LTMs are collected by the CTTM as a 2-fold

ORs: the basic element for the trigger logic is an OR-sum of 96 channels which corresponds to an area of ~ 1000 cm^2 (two half MRPCs). The CTTM is a large VME board (78×41 cm^2) hosting three piggy-backs, each one mounting a FPGA [4]. The top and bottom FPGAs receive the signals coming from the LTMs and apply the trigger logic, then the final decision is taken by the central FPGA which communicates through VME with a computer to the Central Trigger Processor (CTP).

CTTM: porting to a 64-bit environment

In the near future the computer monitoring the CTTM (`alitoftrg`) will be upgraded from a 32-bit architecture to 64-bit. This upgrade requires the porting of the code related to CTTM and compiling it in a 64-bit computer. A 64-bit architecture computer called `alitoftm00` and located in 29/R-003 was used for this purpose: the entire setup was assembled in this laboratory.

As first step the program *ltman* was compiled. It is used to monitor and control the daisy-chained FEACs connected to LTMs. In order to successfully compile it, the following objects were to be compiled as object files:

- `ltm_event` defining the functions acting on the LTM;
- `console` implementing the user interface;
- `v1392util` implementing the writing and reading operations on the registers;
- `CAENVMEDemoVme` implementing the possibility to read a single register knowing its address.

After compiling these sources it was possible to compile and build the executable program in the 64-bit environment. The only required change was adding the definition of Linux OS as preprocessor directives since the definition `LINUX` was not recognized while `__linux__` was.

The CTTM board (see Fig. 1) arrived on July 18, 2016 from Bologna section of INFN (*Istituto Nazionale di Fisica Nucleare*) in Italy. During the transportation the low voltage connections (3.3 V and 5.0 V) were broken and it was necessary to solder them again: using a multimeter the right connections were found and the solderings were done. Once the power supply was guaranteed inserting the low voltage boards in a *Wiener* crate (able to supply 3.3 V), the VME cable connection was plugged into the crate slot #4 since the base address was set to 0x400000. The CTTM registers were read using a CAEN V2718 bridge by means of the program *CAENVMEDemo*, already installed on `alitoftm00`. The reading the registers of the CTTM was impossible at the beginning because of *bus errors*: in order to check if the bridge was the cause of the problems it was tested on a different VME

Figure 1: *Front side of the CTTM board: the three FPGAs are visible. On the backside there are the LVDS connectors for the signals coming from the 72 LTMs.*

module. After changing a faulty bridge we used a new and tested one but the errors were still there. The new bridge was not able to read the registers of the tester VME module in the *Wiener* crate (where the low voltage boards were plugged), while it was correctly working if mounted in another crate. It was therefore decided to use two different crates: one to host the low voltage boards and another one to host the bridge and the CTTM VME connector. The final setup is shown in Fig. 2.

The clock could be sent to the CTTM through optical fiber as shown in Fig. 2 (the yellow cable connected on the front side of the CTTM) or using the one coming from the LTM; in the last case the connector is on the backside.

Once the setup was completed, the central FPGA firmware was reprogrammed using a *ByteBlaster* cable and the software *Quartus II Programmer*: the file used to reprogram it was a `.pof` file written in VHDL language. The reprogramming was necessary since it was impossible to read some FPGA registers which had to be accessible. After reprogramming every registers was readable.

The next step was to compile the software *epcs*. It is used to program the FPGAs on the CTTM using `.rbf` files with no need to use the byte-

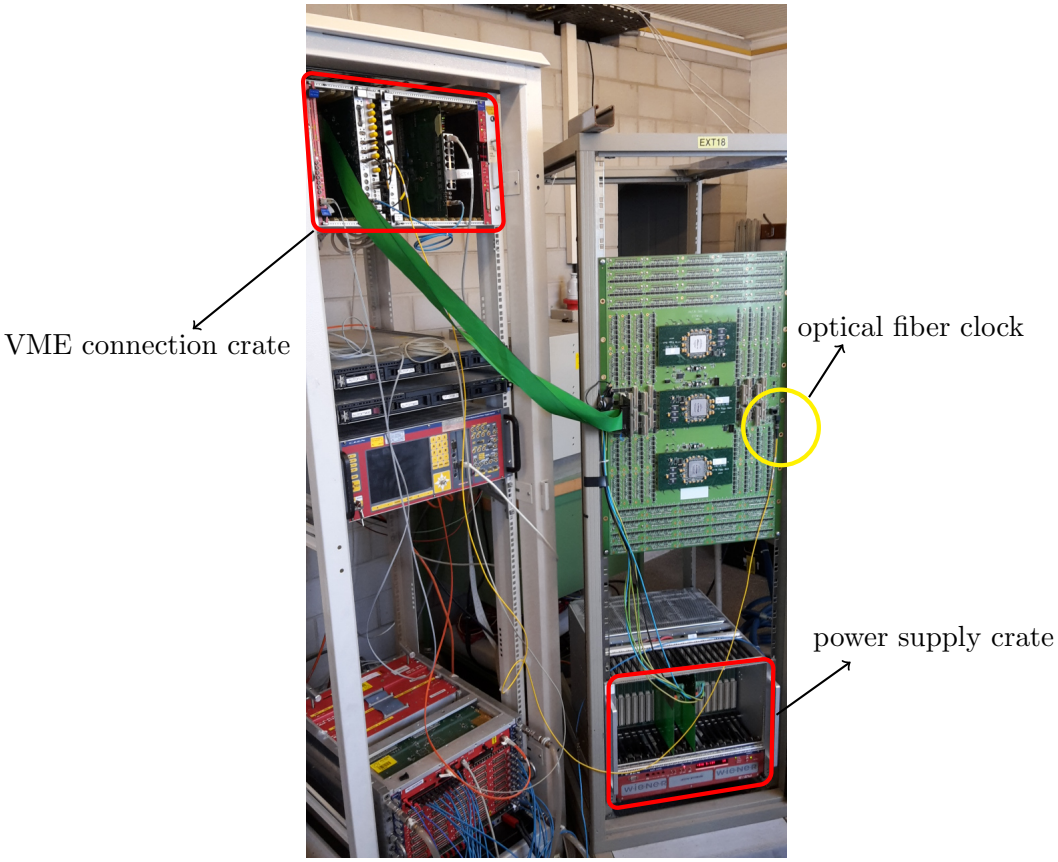

Figure 2: *Setup to test the CTTM board located in 29/R-003.*

blaster. Reprogramming using VHDL files and the byteblaster requires to be physically close to the CTTM board and thi is not always possible. In order to make an executable file the following object files were compiled:

- `libvme` defining the functions handling the communication through VME;
- `libepcs`, the library of functions to reprogram the FPGA firmware.

The other libraries used were `libncurses` and `libCAENVME`, already compiled as shared libraries. The software *epcs* was used to program once more the central FPGA using a `.rbf` file taken from `alitoftrg`, more recent than the `.pof` file.

The main part of the porting of the code in the 64-bit environment was to recompile the server of the CTTM. The source file used is `running_server_v8.c`. This program requires several other files in specified paths of `alitoftrg`: they were fetched and located in the same path in `alitoftm00` in order to simulate the environment present at P2 (ALICE location). The object files to be compiled in order to have the executable were:

- `utility`, a collection of functions handling the memory access, buffers and threads;
- `stools` implementing logbook messages tools;
- `simplelog` handling logbook functions;
- `libltm` implementing the functions to read LTMs from CTTM;
- `libInfo` and `infoLoggerConfig` to handle the *Logger*;
- `cfgFile` implementing some functions from `stools` and handling configuration files;
- `cttm_func` implementing the functions acting on the CTTM.

In addition to these files also `libvme`, `libepcs` and the library `libdim` (database functions) were required. A *makefile* was written to easily build the server.

The executable produced using the source from the computer `alitoftrg` had problems while running in the setup arranged in building 29 due to the fact that the environment is not completely the same as P2. The following changes had to be applied to run it without errors:

- setting manually the crate status as ON since it was detected as OFF. To detect the CTTM status the DIM communication system is used and the node is the server `alitofldc`, a computer in the laboratory. The server has to detect if the crate supplying the power is on. The cause of this misdetection may be that the low voltage crate and the VME reading one are different;
- removing thresholds setting because of problem accessing to the memory registers of bottom FPGA. These threshold are used to improve noisy channels;

Using the *infoBrowser* tool to check the server online and the *DIM* database to handle commands and services of the node `alitoftm00` it was possible to change the Detector Constrol System (DCS) status of the CTTM. The status identifies the current situation of each detector (or part of it): some functions are possible only for particular status. The CTTM status was set to STANDBY_CONFIGURED from STANDBY and then to READY. This means the CTTM was ready to work and it had to be checked.

In order to check if the CTTM works properly the board was pulsed. As first step the LTM located in 29/R-003 was connected to the backside of the CTTM through a LVDS cable and the clock from the LTM was passed to CTTM. Running the CTTM server (*running_server_v8*) the board was set to READY and using the software *ltman* the toggle status of the LTM was set

to TOGGLING. In this way it was possible to pulse the CTTM board: once stopped the toggling LTM it was possible to verify that the CTTM received the signal. This was possible because the server automatically updates the files *rate_pre* and *rate_after* with the trigger rates before and after setting the threshold for noisy channels. The files are updated as soon as the CTTM receives a pulse from LTMs. Since thresholds are null due to problems accessing the memory of the bottom FPGA only *rate_pre* should change. Pulsing the CTTM the file changed and moving the LVDS connection on the backside of the CTTM the *rate_pre* file changed accordingly.

AMORE framework

The *Data Quality Monitor* (DQM) framework used by ALICE experiment is called AMORE (*Automatic MOnitoRing Environment*). Several processes (analyzing a sample of raw data) called *agents* continously run during the data acquisition [5]. They are based on generic libraries such as `AliQAChecker` (used to check the histograms setting the quality flag) specialized by inheritance for each detector (e.g. for the TOF detector the class is `AliTOFQAChecker`). This choice of basing the DQM on *AliRoot* libraries is not common for all the detectors: a few of them use custom made functions.

A sample of data called *chunk* is taken and analyzed every $\sim 60\ s$ (MC, *Monitor Cycle*): each detector agent analyze this dataset over a loop, updating the QA plots and setting the quality flags. Some of these plots are checked during the data taking by the DQM *shifter* always present in the ALICE Control Room.

Every plot produced by each detector could be set as "shifter" plot or "expert" plot: in the first case it is shown by default in the amoreGUI window (the graphic user interface) and it must be constantly checked by the shifter. The expert plots instead are not shown by default but if there are problems concerning a detector the expert may need to check them. Each single detector expert has to set periodically the detector quality flag for each run: in order to summarize the performances a *summary image* is produced with the most significant plots.

Some histograms have a message box changing according to their quality flag: the colour and the message shown change as the flag changes. The possible flags are:

- *kINFO*, everything is working fine. The box is set to green and a reassuring message is displayed;
- *kWARNING*, not an error but something is not working as it should. In this case the box's colour is set to yellow and a message is shown usually suggesting to check what's the type of the run;

- *kERROR*, some parameters are out of range and this means the detector has problems. The box is set to red and a message telling the shifter what to do is shown (checking the TWiki or calling the *on-call* expert);
- *kFATAL*, the histogram is empty. In this case the box is empty and grey since there are no data to analyze online. This problem has to be fixed immediately.

On the left side of the amoreGUI window all the histograms produced are shown as elements of a tree. All the histogram names end with a coloured box representing the quality flag of the detector: if these are not green, the shifter has to check what is the problem and acts accordingly.

TOF DQM: QA histrograms split by trigger classes

The QA plots for the TOF detector are initialized, built and updated by the class `AliTOFQADataMakerRec`: data are taken from the raws stream and used to fill the plots. At the end of a monitor cycle the plots are checked by `AliTOFQAChecker` which sets the quality flag at the end of each MC.

Despite the number of raw histograms (>30) only 12 are not for experts: besides having to be checked by the DQM shifter those plots are included into the TOF summary image (see Fig. 3).
The most meaningful plots regarding the performances of the Time-Of-Flight detector are implemented with a message box [6]:

- `hTOF_Raws` plotting the multiplicity per event;
- `hTOF_RawsTime` plotting the arrival time per hit in ns;
- `hTOF_RawsToT` plotting the ToT (*Time-over-Threshold*) per hit in ns.

Splitting the histograms by trigger calsses is possible declaring it in the configuration file for the TOF Quality Assurance (`tofQA.config`). In order to add a copy of the clone to the summary image, it is needed to include the histogram name and the selected trigger classes in the config file. For example in the summary image of Fig. 3 the clone of `hTOF_Raws` and `hTOF_RawsTime` are present and indicated by the trigger class `kINT7` (the *minimum bias* trigger). Respectively, the original histograms are the second and the fourth in the first line while the clones are the first and the third.

Since most of the events have null multiplicity `hTOF_Raws` has a peak for null multiplicity events. Selecting the minimum bias trigger will canel this peak: indeed in this case the very low multiplicity events are not taken into account. As shown in Fig. 3 the message box for the clone is disabled. On the test machine `pcaldblade2` the latest version of *AliRoot* (version 4.4.7) was built: the changes were done on its libraries. In order to simulate the

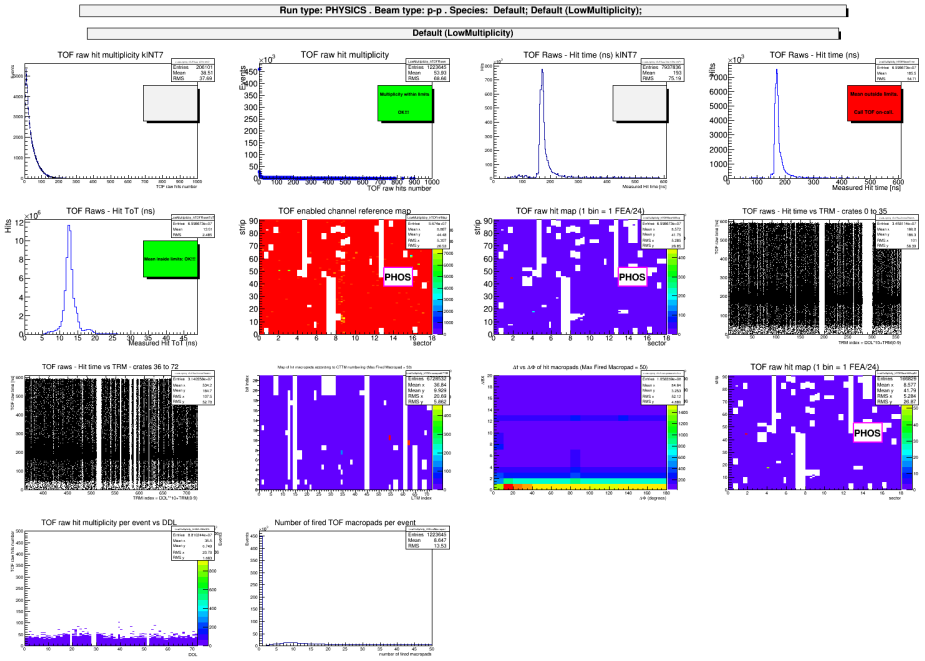

Figure 3: *Summary image from run* 260537. *The check of trigger class split histograms was not yet implemented.*

real data taking and the real DQM environment the data from run 254378. Working on the `AliTOFQAChecker` library the message box for the trigger class clones was enabled and a bug in the AMORE framework was found: the flag next to the clone name in the tree list was always purple. The bug was fixed by the AMORE expert Barthélémy von Haller.

A new feature for the summary image was implemented: it is possible to decide to show or not the original histograms (trigger blind) and to specify the trigger class clones wanted to be shown in the summary image. This may allow to show the trigger class clone in the amoreGUI window but not in the summary image. Even if this feature is not yet used it could be useful when the number of plots in the summary image increases and only few trigger classes will be required to set the detector quality flag at the end of run.

The amoreGUI window is shown in Fig. 4 and the final summary image is shown in Fig. 5. The message box was correctly implemented and the detector flags are well set. The trigger classes used are `kINT7`, `kLifeCENT` and `kCalibBarel` (different aliases for the minimum bias trigger).

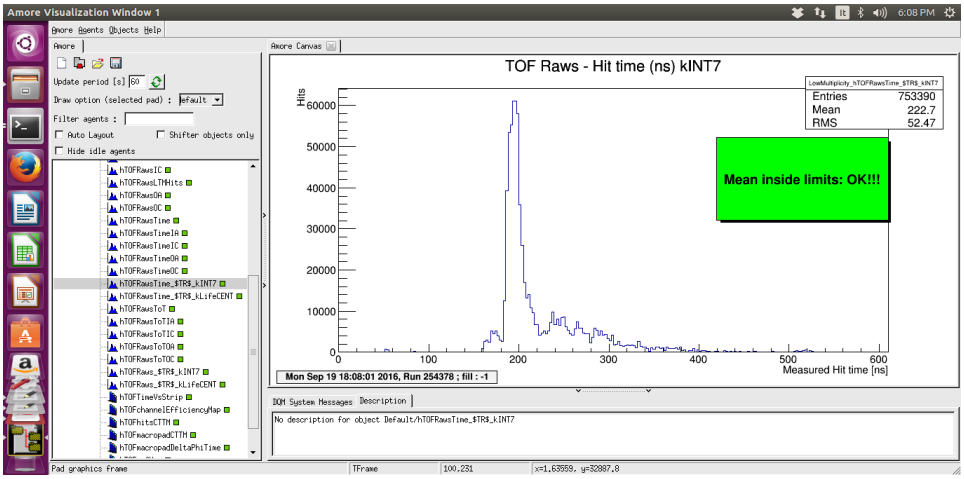

Figure 4: *The amoreGUI window showing the clone of* `hTOF_RawsTime` *for the trigger class* `kINT7`.

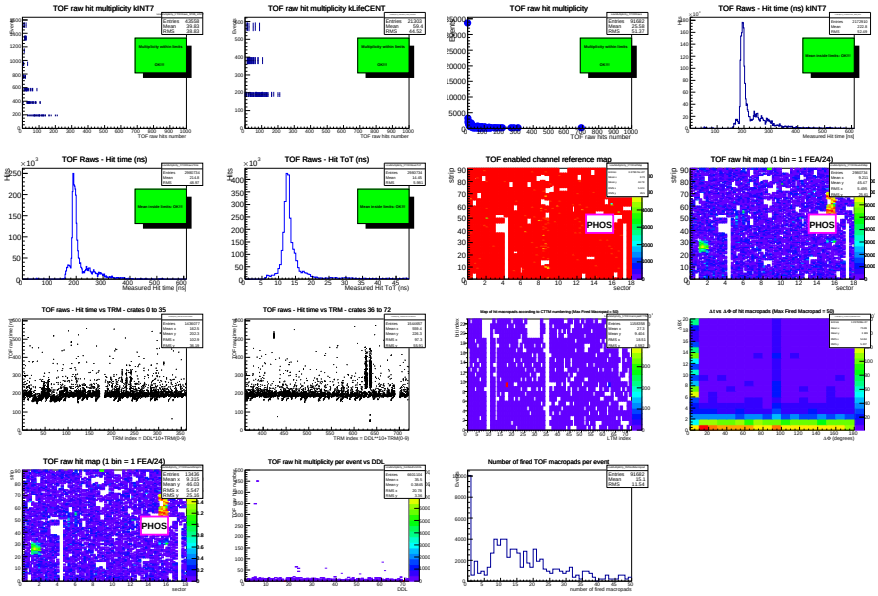

Figure 5: *Summary image with message box implemented for clones.*

Conclusions

The testing setup for the CTTM board was succesfully settled in 29/R-003. In order to avoid *bus errors* two different crates were used for VME connection and for power supply. At first the FPGAs firmware had to be reprogrammed using a byteblaster cable and a VHDL program, then they were reprogrammed using the *epcs* software. This software had to be recompiled in the 64-bit environment provided by `alitoftm00` computer. Once all the FPGAs were correctly reprogrammed the server (*running_server_v8*) was compiled in the 64-bit architecture. This was the crucial part of the work: several libraries had to be compiled in order to make it possible. As next step the DCS status of the CTTM board had to be set to READY: the differencies betwen the set up settled in 29/R-003 and P2 required some minor changes in the code. After the code was correctly changed it was possible to set the DCS status of the CTTM to READY. This is the status needed by CTTM to work correctly. The LTM was used to test the board: a pulse was sent to the CTTM to check if it worked properly. The test was successful. A *makefile* was written in order to compile properly the server when the upgrading of the architecture of `alitoftrg` will be done.

The code of the TOF amoreQA was changed in order to enable the message box also for trigger class clones. Using the test computer `pcaldblade2` it was possible to simulate the online DQM situation: for this purpouse the dataset from run 254378 (pp collision) was used. The tests returned positive results in several situations such as having more clones for the same histogram. The possibility to decide the trigger class clones to show in the summary image was implemented as new feature duing the coding. Moreover it's possible to draw on the summary image only the clones of an histogram with no need to show the orginal one. The changes to the code will be committed in the near future to the new AliRoot version. The message boxes enabled for clones in the summary image will help the TOF experts setting the quality flag for TOF at the end of each run. This will also help the DQM shifters to check if there are problems concerning TOF detector for some particular trigger classes: so far there was no message box helping them in this case.

References

[1] The ALICE Collaboration. The ALICE experiment at the CERN LHC. *Journal of Instrumentation*, 3(08):S08002, 2008.

[2] G. Dellacasa et al. ALICE technical design report of the time-of-flight system (TOF). 2000.

[3] A. Akindinov et al. A topological trigger based on the Time-of-Flight detector for the ALICE experiment. *Nuclear Instruments and Methods in Physics Research A*, 602:372–376, 2009.

[4] Alessandro Silenzi. *The topological trigger system of the TOF detector for the ALICE experimente at the LHC*. PhD thesis, Alma Mater Studiorum, Università di Bologna, 2010. Relatore Prof. Maurizio Basile.

[5] AMORE Modules Developer manual. `https://alice-daq.web.cern.ch/products/amore-modules-developer-manual`, 2015.

[6] AliROOT Reference Guide. `http://aliroot-docs.web.cern.ch/aliroot-docs`, 4 2016.