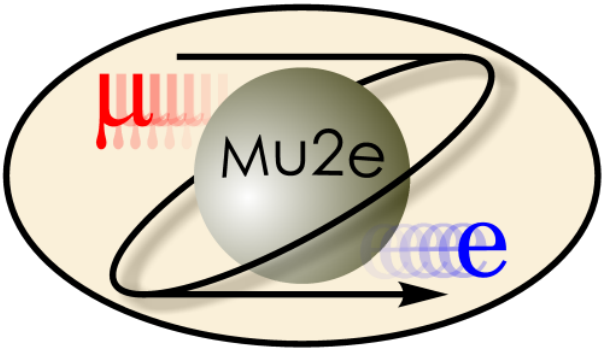
Commissioning of the Mu2e Data AcQuisition system and the Vertical Slice Test of the straw tracker

Italian Summer School at Fermilab

Sara Gamba

Internship carried out from July 2023 to September 2023



Internship tutors : Pavel Murat
Referent professor : Simone Donati

Educational institution : University of Pisa - Department of Physics
Internship host company : Fermilab

Abstract

The report discusses the commissioning of the DAQ system of the Mu2e experiment. We studied how the tracker works and especially the tracker readout. We used a generator to send pulses and we tried to understand the output and non-output of the DTC, in order to validate the event and run format and all the failure modes.
In section one, we describe Charged Lepton Flavour Violation, the main purpose of Mu2e, in section two we describe Mu2e experiment. Section three presents the tracker description and readout, meanwhile section four the DAQ system and event building. In section five, we start explaining the analysis we have done on events and all the failure modes we discovered. In section six, we have done an analysis of the logger and boardreader rate, we validated the generator frequency in section seven and studied the number of hits in function of the channel number in section eight. To conclude we investigated the channel to channel time differences in section nine and in section ten we explain our Monte Carlo simulation, reproducing ROCs readout.

Contents

1 Charged Lepton Flavour Violation

Three is the number of flavours of charged leptons in the Standard Model: electron (e), muon (μ) and tau (τ).
In the SM, lepton flavour is always conserved, so we should not see the transformation of a lepton of a certain flavour into another one.
CLFV is the violation of the lepton flavour number (n_l). If a muon decays to an electron, we should have neutrinos, in particular the reaction we expect is $\mu^- \to e^- \nu_\mu \nu_e$. The presence of the neutrinos conserves flavour.
Interactions of the SM fermions are non-diagonal in flavour, due to the quark mixing and neutrino oscillation. The charged leptons have not experimentally shown evidence of flavor violation yet.
If we extend SM to account for neutrino oscillation, we should see CLFV but it is highly suppressed ($< 10^{-50}$ BR), significantly lower than the sensitivity of any current or planned experiment. Therefore, experimental observation of any CLFV process would be unambiguous evidence of New Physics.
Many extensions of the SM predict much higher rates of CLFV processes, sensitivity levels which can be probed by experiments like Mu2e at Fermilab [al21a].

2 Mu2e experiment at Fermilab

Mu2e is an under construction experiment in Fermilab. The main purpose of this experiment is to find the neutrinoless coherent conversion of muon to electron in the presence of an atomic nucleus $\mu^- N \to e^- N$.
It will measure $R_{\mu e}$ of the event shown before, respect to all the possible muon capture events on Alluminium target.

$$R_{\mu e} = \frac{\Gamma(\mu^- + N(A,Z) \to e^- + N(A,Z))}{\Gamma(\mu^- + N(A,Z) \to \nu_\mu + N(A,Z-1))} \tag{1}$$

We expect about four orders of improved sensitivity with respect to the current best limit of $R_{\mu e} < 7 \times 10^{-13}$ $(90\% CL)$ set by the SINDRUM II experiment.
We expect that Mu2e will improve momentum resolution from 2 MeV/c, SINDRUM II, to 1 MeV/c. We expect to take data starting from 2026 for three years.

2.1 Mu2e apparatus

The structure of this experiment is shown in Fig.1. The Mu2e experiment is divided in three part, characterised by different magnetic fields. This three parts are called *solenoids*: Production Solenoid (PS), Transport Solenoid (TS) and Detector Solenoid (DS).

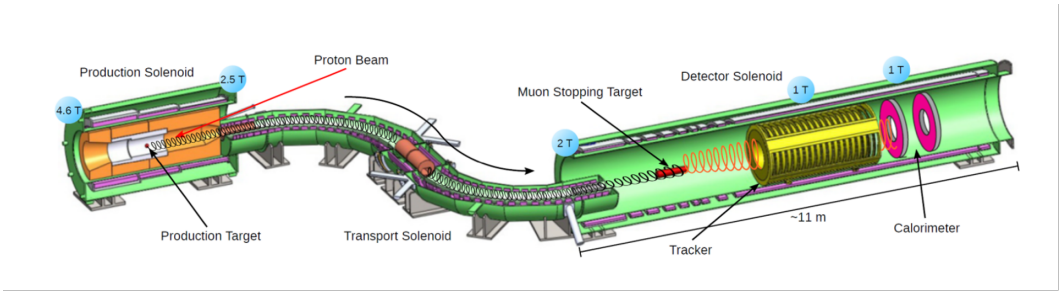


Fig. 1: Schematic view of the Mu2e experiment at Fermilab.

2.1.1 Production Solenoid

A 8 GeV pulsed proton beam is extracted from Fermilab Delivery Ring and it hits on the tungsten production target in the PS. The proton beam structure is 250 ns-wide proton pulses separated by a gap of 1695 ns. The reasons of the pulsed proton beam is given in the 2.2. Most of the particles produced in pW interactions are pions and the particles produced backwards are reflected by the PS magnetic mirror and pushed forward to the TS.

2.1.2 Transport Solenoid

Pions are produced in the PS and decay in flight by the following process $\pi^- \rightarrow \mu^- \nu_\mu$. The collimators at the entrance, center, and exit of the TS define the TS momentum acceptance, reducing the transport efficiency for particles with momenta above ~ 100 MeV/c. The curved magnetic field of the TS helps to remove the charged particles of opposite signs. Two absorber windows in the TS are used to reduce the antiproton background.

2.1.3 Detector Solenoid

The DS magnetic field has two regions: an upstream region with a graded magnetic field and a downstream end with a uniform field of 1 T, where the main detector elements, the tracker and the calorimeter, are present. The stopping target (ST) is positioned in the upstream region of the DS. The average momentum of the muons entering the DS is ~ 50 MeV/c, and about 1/3 of them stop in the ST. The ST is made of 37 Al annular foils spaced 2.2 cm apart. The segmented geometry of the ST helps to reduce electron energy losses and the central hole helps to reduce radiation in the detector.

2.2 Signal and Backgrounds

The muons stopped in the target foils can form muonic atoms and rapidly cascade to the 1s orbit in the Al atoms. After that there could be the muon-to-electron conversion resulting in a monochromatic peak with energy:

$$E_{CE} = m_\mu - E_{recoil} - E_{bind} \tag{2}$$

where $m_\mu = 105.7$ MeV/c^2 is the muon mass, $E_{recoil} = 0.2$ MeV is the recoil energy of the target Al nucleus, and E_{bind} is the binding energy of the 1s state of the muonic atom. For the Aluminum, $E_{CE} = 104.97$ MeV. Since the conversion process is coherent, the outgoing nucleus remains in the ground state, the experimental signature is always a 105 MeV electron, unaffected by changes in nuclear energy levels or other secondary emission. Electrons with ∼105 MeV energy could also be produced by the following types of background. In figure 2, we can see the signal we are looking for and the backgrounds.

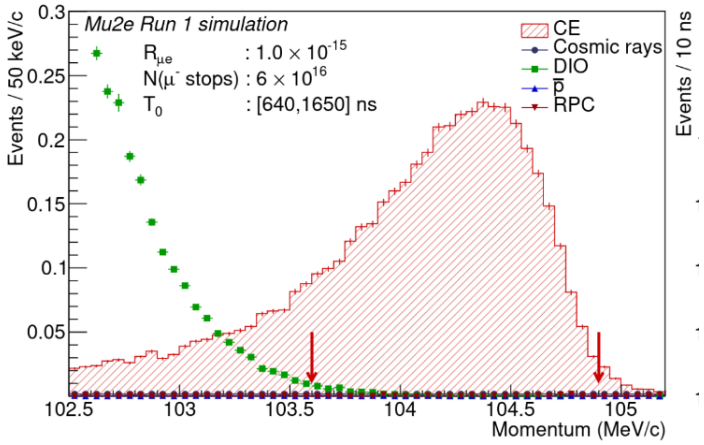


Fig. 2: Signal and backgrounds near 105 MeV.

- **Decay in Orbit** (DIO): muons that are stopped in the alluminium can decay in an electron and two neutrinos. The electron momentum distribution is similar to that of the Michel spectrum of the free muon decay, but the outgoing electron can exchange a photon with the nucleus. This recoil of the electron off the nucleus shifts the kinematic endpoint of the DIO electrons to be equivalent to the momentum of the CE signal. The tracker needs to have excellent momentum resolution and track reconstruction efficiency. The Mu2e tracker is expected to provide a momentum resolution of ∼100 keV/c at the CE momentum of ∼105 MeV/c;

- **Radiative capture of pions** (RPC): it occurs when pions contaminate the muon beam and stop in the Al target generating significant background which rapidly falls with time. The stopped pions undergo $\pi^- + N(A, Z) \rightarrow \gamma^* + N(A, Z-1)$, followed by an asymmetric $\gamma \rightarrow e^+e^-$ conversion which can yield electrons at the conversion energy. The RPC background can be suppressed by setting a delayed live-time window with respect to the proton pulse arrival at the production target, schematically shown in Figure 3. The data-taking begins at about 640 ns after the proton pulse arrival and proton pulses are separated by a time window of 1695 ns;

- **Radiative muon capture** (RMC): $\mu^- + N(A, Z) \rightarrow \gamma^* + \nu_\mu + N(A, Z-1)$ is a process analogous to RPC, but with a softer spectrum. For Al, the maximal energy of the RMC photon is ∼101.9 MeV, about 3 MeV below the expected conversion signal. The

timing dependence of the RMC electron rate is defined by the lifetime of the muonic atom unlike the RPC background;

- **Antiprotons** ($\bar{p}$): antiprotons are produced by the interactions of the proton beam at the tungsten Production Target. Antiprotons can then travel through the TS, unaffected by the charge selecting and annihilate in the ST in the DS, producing signal-like electrons. Antiprotons are much slower than the other beam particles so they cannot be efficiently suppressed by the delayed data taking window. Absorber elements will be placed at the entrance and at the center of the TS to reduce the antiprotons background;

- **Cosmic rays**: Cosmic muons can strike on the Mu2e ST and they can produce an electron with the conversion energy and exit the detector with no other trace. Such an electron is indistinguishable from the signal since it comes from the ST and has the right energy. Mu2e will have a **Cosmic Ray Veto** detector surrounding the Detector Solenoid to veto such events.

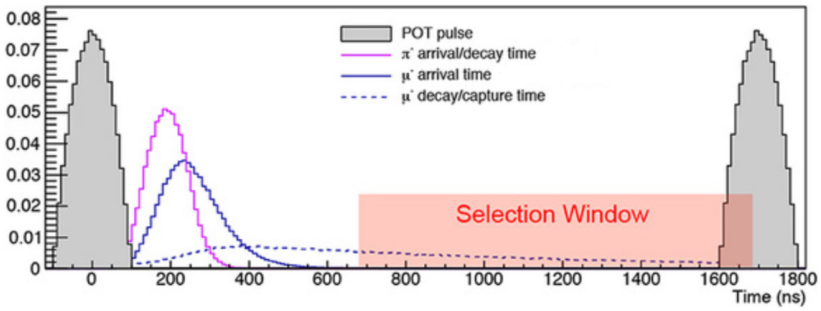


Fig. 3: Time window between proton pulses and selection window.

3 The tracker of Mu2e experiment

In the Detector Solenoid there is a 3 m long tracker and it is located 3 m downstream of the stopping target in the uniform 1 T region of the DS magnetic field.
It is composed by 5 mm diameter and 40-110 cm long straw tubes, filled with a 80%:20% Ar:CO_2 mixture at a pressure of 1 atm. The whole detector will be in vacuum and the covering radii will be between 38 cm and 68 cm. The inner 38 cm region of the tracker is not instrumented because it could be blind to muon beam and to the associated activity and also be blind to most of the lower energy electrons coming from Standard Model muon decay.
There are 96 straws per panel, 6 panels (rotated 30 degrees relative to each other) per plane, 2 planes per station and in total there are 18 tracking stations: 216 panels. In figure 4 and in figure 5, we can see a schematic view of our tracker [Lee16].

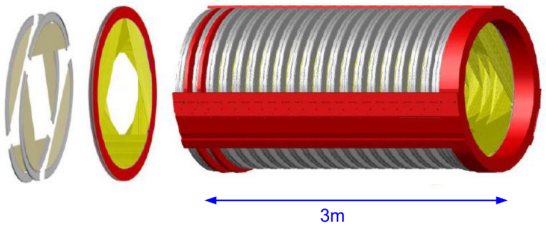


Fig. 4: The tracker structure, from the left panels, planes and the 18 stations.

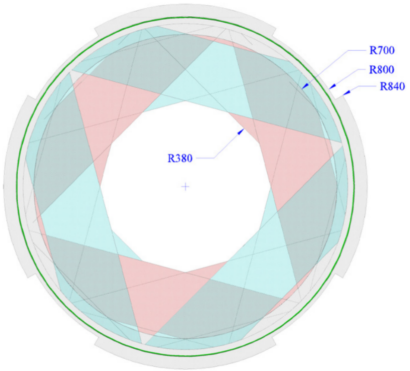


Fig. 5: The plane structure divided in planes rotated 30 degrees relative to each other.

3.1 Tracker readout

Front-end electronics are housed in the outer part of the panel (semicircular crown), as we can see in Figure 6. The tracker readout system provides information about the pulse timing at the end of each straw and total height. Each pulse timing is stored and both difference and sum contain useful information: the difference is computed in order to measure the longitudinal position along the wire, while the sum depends on the particle arrival time and the drift distance. The pulse height is also measured to do PID.
As we can see in Figure 7, signals are readout from the ends of each straw on the panel: an earlier signal and a later signal propagate to the stage of amplifiers. The amplifiers are installed vertically in a compact manner, as we can see in the analog sides of the motherboard in Figure 6. Amplifiers turn signal current to voltage, amplify, shape and bias signal, distribute high voltage (HV side), provide calibration charge injection (CAL side). Then signals are sent to DRAC. From the comparator outputs are sent to TDCs (∼20 ps/tick), a normal trigger that requires coincidence between two straw ends. Integrator adds the signals from both ends of a straw and the analogical sum of the signals is sent to an ADC. ADC digitizes the signal. TDCs are implemented in FPGAs.
All the digitized information are sent to a digitizer board that transmits the digitized waveform to the Data AcQuisition system.

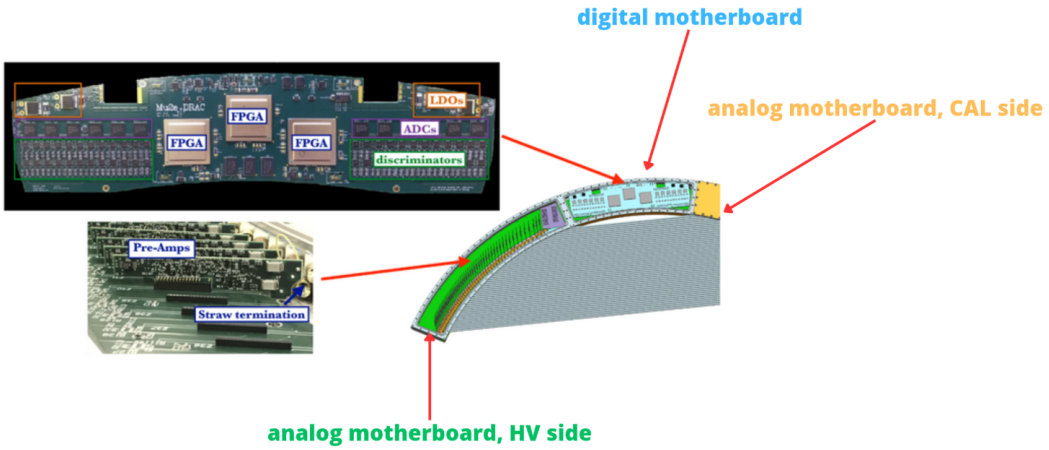


Fig. 6: Schematic view of a panel: a circular crown sorrounds the staws. It is divided in three parts: the green part is one of the two analog motherboards (High Voltage side), the blue one is the digital motherboard and the yellow one is the other analog motherboard (CALibration side).

4 Data AcQuisition system

The Mu2e Trigger and Data AcQuisition System -TDAQ- is the system to collect digitized data from the tracker, calorimeter, cosmic ray veto and monitor the beam status. The TDAQ uses 36 dual-CPU servers to handle a total rate of 192000 proton pulses per second.
The Run Control Host receives beam status and timing information from the Accelerator Control Network. The Control FanOut -CFO- module in the Run Control Host is responsible for reporting when the arrival of the first and the second proton pulses and also for the synchronization of DTCs. The Read Out Controller -ROC- continuously streams out the data collected between two proton pulses from the detectors to the DTCs, Data Transfer Controllers.
Data are stored and online processed, and then Trigger is activated, then added to the cosmic veto. The DAQ server filters the events and then they are sent to the offline storage.
TDAQ uses OTSDAQ as software solution, that is made by ARTDAQ and ART to filter events (data transfer, event building and event reconstruction) and processing frameworks. OTSDAQ uses a run control system using the data acquisition software XDAQ implemented at CMS. The OTSDAQ is developed in two main directions: server and web side. The server side is developed in C++. The web side is developed in HTML and JavaScript [al21b].

4.1 Event building

A schematic view of Mu2e DAQ components is shown in Figure 8. Considering a system with two DTCs and a CFO, where each DTC can have six ROCs attached and the two DTCs are connected to an Event Building Switch -EBS-, a Heartbeat Packet -HB- for Event 0 is generated by the CFO, and passed to the two DTCs via the timing links. The DTCs pass

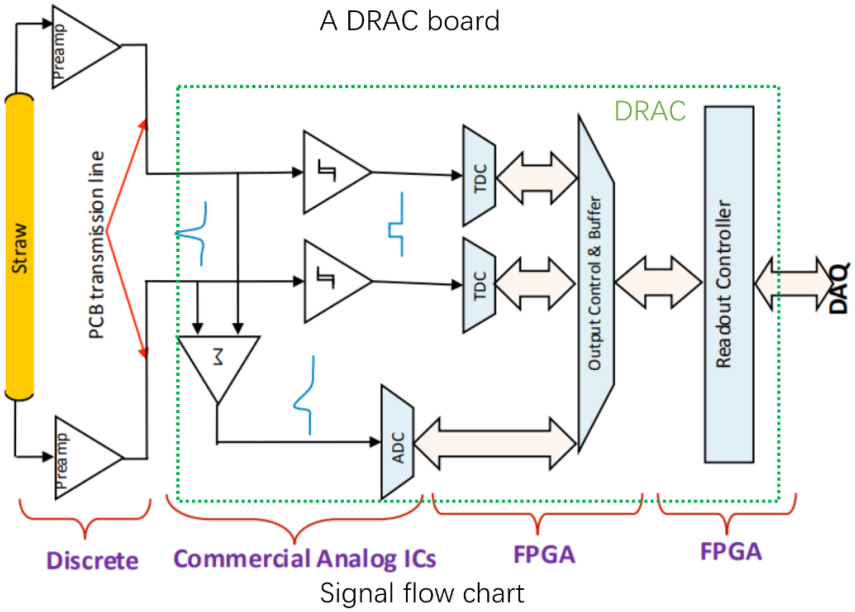


Fig. 7: ReadOut of the tracker and DRAC board.

the HB Packet to the ROCs and then generate Data Request Packets -DR- for the event and transmit those to the ROCs.

The Event Window Marker is generated by the CFO and sent to the DTCs and then DTCs pass EWM to ROCs. ROCs begin acquiring data for Event 0, based on information in the Event 0 HB Packet. A HB Packet for Event 1 is generated by the CFO, it is passed to the DTCs and then to the ROCs. The DTCs generate DR Packets for Event 1 and transmit those to the ROCs.

ROCs end acquisition for Event 0 and begin acquisition for Event 1. ROCs reply to Event 0 DR Packets by sending Event 0 data to their DTCs.

Now HB and DR for Event 2 are made. ROCs end acquisition for Event 1 and begin for Event 2. ROCs replay to DR for Event 1. DTCs now have Sub-Event data for Event 1. DTC 0 adds Sub Event Header to Event 1 data and transmits it via the Event Building Switch to DTC 1. DTC 1 now has all data for Event 1. It adds Sub Event Header to the Event 1 data it collected from it's ROC's, adds an Event Header and outputs the complete event over PCI. Then it continues also for Event 3.

In our analysis we are using only one DTC with the CFO Emulator, then the DTC builds events in exactly the same way, but an Event Building Switch isn't necessary.

5 Analysis of the events

We did an analysis of the readout teststands of the motherboard. We used a generator to send pulses and we tried to understand the output and non-output of the DTC, in order to validate the event format and all the failure modes. We were reading 1 ROC (96 channels), which is the equivalent of one panel. We varied the generator event window which is the

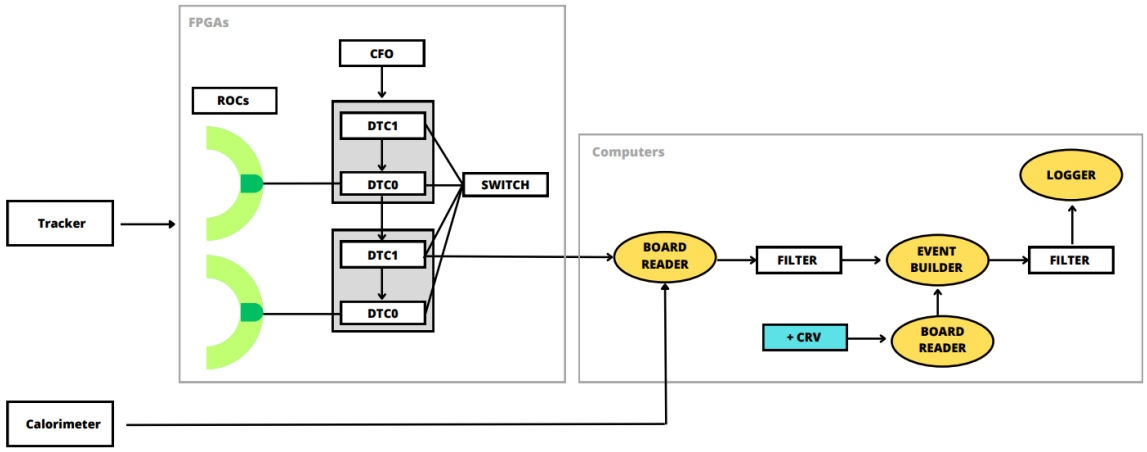


Fig. 8: Mu2e DAQ components diagram.

equivalent of the difference between proton pulses and the generator frequency. These two features define the number of *hits per event*, indeed ROC buffer has space for 255 hits:

- $N_{gen} < 255$: $N_{readout} = N_{gen}$;
- $N_{gen} \geq 255$: $N_{readout} = 255$, because the buffer is already full.

We were generating *hits* in each channel at 31.29 MHz/(2^7+1) and 31.29 MHz/(2^9+1), so $\sim$ 250 kHz and $\sim$ 60 kHz.
We used different time windows between the pulses during data acquisition. We tried to analyze data and understand errors or features.
We didn't undestood the event size distribution as soon as we decrease the time window.
We have discovered different types of failure mode in the data:

- Events marked as non-valid;
- Extra 16 bytes event;
- Non existent channel IDs.

5.1 Structure of an event

We have studied the main structure of an event. With the word event, we mean the 96 channels output given by a generator pulse. An example of an event is shown in Fig. 9. As we can see, lines of the event are called *packets* (orange). A packet is 16 bytes long. Every word in a packet is 2 bytes long.
Two lines form a hit. In one event we can have several hits from the same channel. Channel number is marked in pink and it is the first word of a hit.
The first packet of an event is called *data header packet* (black) and in particular, it contains some useful information to do data analysis. In green we have the *validity* of an event: when the event is marked as *valid* we can find 0x8050, when the event is marked as $non-valid$

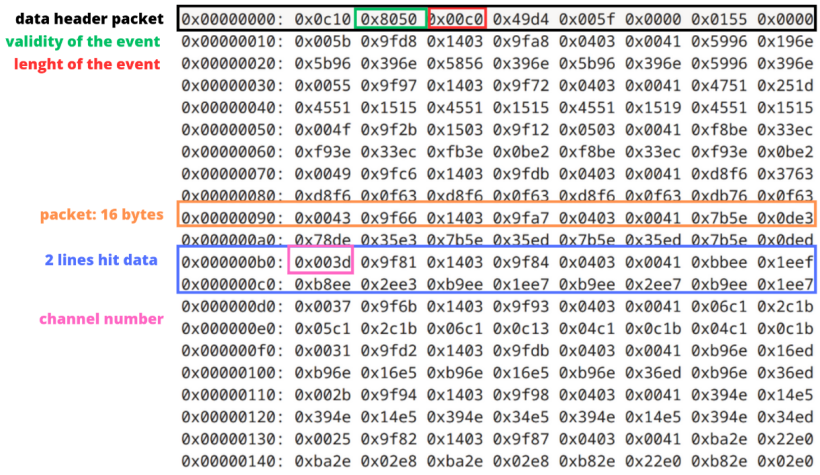


Fig. 9: Structure of an event.

we can find 0x0050. In red we have the length of the event.
The first column of the event stands for the line number, but the interesting parts of the event are on the right: we can compute the timing of a hit using the second and the third word in the packet.
For example, taking into account the line 0x000000b0, so channel number 0x003d, we compute timing hit using 0x9f81 and 0x1403: our timing information will be 039f81 in hex, so we should translate it in decimals and multiplied by 5/256 ns, which is the TDC bin width.

5.2 Event size distribution

Our first step of the analysis was to do event size distributions of different RUNs. Here, we show the results for RUN227 (Fig. 10) and RUN231 (Fig. 11) with a time window of 1000 $\times$ 25 ns and 300 $\times$ 25 ns. Here we show pictures of the two RUNs.

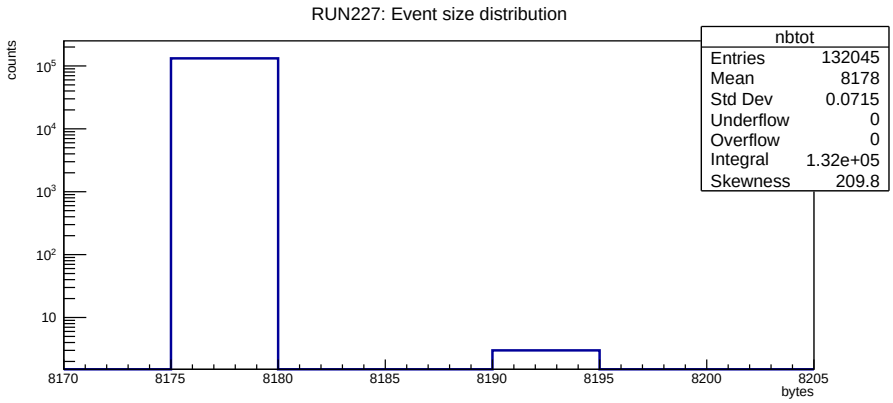


Fig. 10: Event size distribution for RUN227. Most of the events are peaked in 8.178 kB. We can see that there is one occurrence of 16 extra bytes.

As we can see in Fig. 12, decreasing time window, event size of an event decreases.

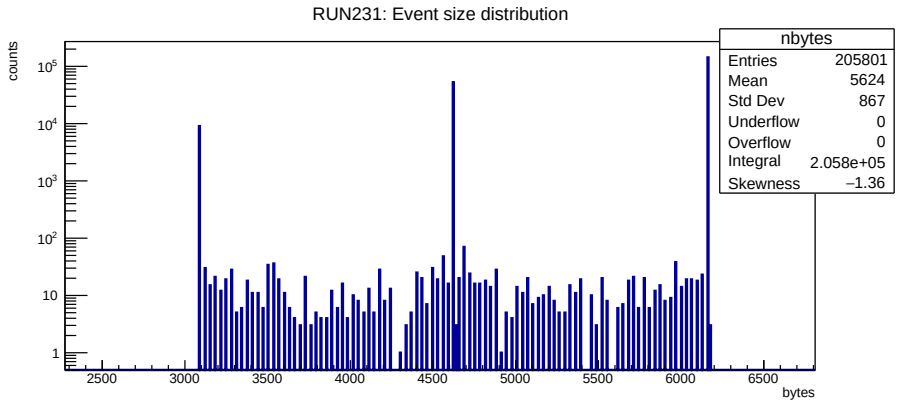


Fig. 11: Event size distribution for RUN231. We can see that, as we reduce the TW, the event size decreases, as we could expect, but the problem is that we don't understand the distribution.

At the moment, we don't understand why the event size distribution stops being peaked as we decrease the time window and it starts to have different event sizes.

5.3 Failure modes

5.3.1 Events marked as non valid

The first failure mode we have noticed concerns the validity of the events: some of the events we have studied are marked as non valid. The difference between a valid (Fig. 13) and a non valid (Fig. 14) event is, as we explained in subsection 5.1, the marker number in second position after the length of the event in the *data header packet*. In particular, a valid event is marked with 0x8050, while a non valid event is marked with 0x0050. In substance, the main differences between a valid and a non-valid event can be various: extra 16 bytes, non existent channel ID, but also events that seem to be good are sometimes marked as non-valid, as we can see, for example, in Fig. 15, where we can see that the non-valid events are distributed in the same way as the valid ones.

We report the event size distribution for RUN227 (Fig. 15) and RUN231 (Fig. 16): all valid events are in blue and all the non-valid events are red.

As we can see in the previous histograms, a few percentage of events is marked as invalid. In Tab. 5.3.1 we report a short resume of three RUNs characteristics and percentage of invalid events over all events.

5.3.2 Events with extra 16 bytes

We have noticed another type of failure mode: the one where the events present some 16 extra bytes. We can see this type of failure mode in RUN247 (Fig. 17).

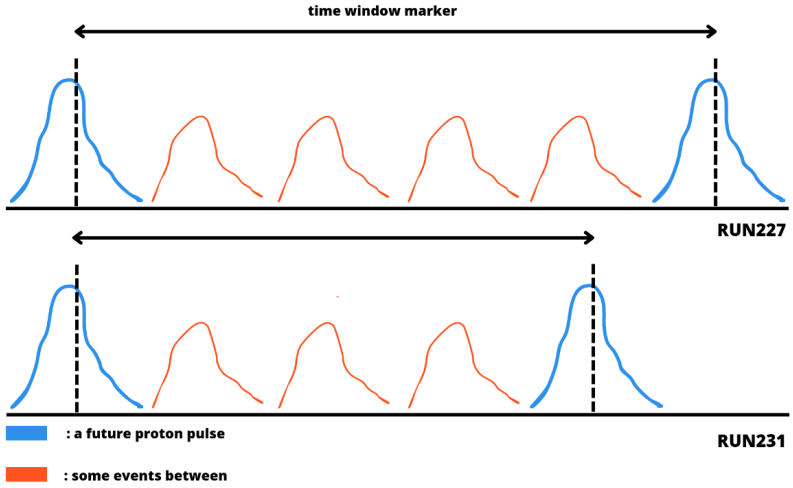


Fig. 12: Description of what happens as soon as we reduce the time window marker. The blue peaks are the future proton pulses and the time window marker is the time space between them, the red peaks are the events in a given time window. In a bigger TW more events can fit.

```
0x00000000: 0x0c10 0x8050 0x00c0 0x1ba6 0x0000 0x0000 0x0155 0x0000
0x00000010: 0x005b 0x0660 0x1409 0x062e 0x0409 0x0041 0x5a56 0x2565
0x00000020: 0x5a56 0x1565 0x5956 0x1565 0x5956 0x1565 0x5a56 0x1565
0x00000030: 0x0055 0x061b 0x1509 0x07fb 0x0409 0x0041 0x4751 0x0d13
0x00000040: 0x4751 0x351d 0x4751 0x351d 0x4751 0x351d 0x44d1 0x351d
0x00000050: 0x004f 0x07b3 0x1409 0x079b 0x0409 0x0041 0xfabe 0x2bea
0x00000060: 0xf9be 0x2bea 0xf9be 0x2bea 0xfabe 0x2bea 0xfabe 0x1be6
0x00000070: 0x0049 0x064f 0x1409 0x0660 0x0409 0x0041 0x380e 0x00e0
0x00000080: 0x3bf6 0x00e8 0x380e 0x00e0 0x380e 0x00e0 0x380e 0x3f60
```

Fig. 13: Example of valid event.

We show one of the events with 16 extra bytes in Fig. 18. In general, we would expect that channels are read in a defined sequence: first we read the first FPGA (CAL0 and CAL1), then the second one (HV0 and HV1) and all this sections contain 24 channels each. Analyzing the +16 bytes event, we can see that the lenght of a normal event in that RUN should be marked as 0x0060, but in that event is 0x0062: looking for anomalies, we have seen that, in Fig. 18, the yellow channel, from HV1, second section from the second FPGA, is read after the pink channel, from CAL1, second section in the first FPGA. In the middle some channels from CAL1, HV0 and HV1 miss, infact the pink channel is not the last channel of CAL1 and the yellow channel is not the first channel of the HV1 section. Analyzing the timing of pink and yellow channel, we see that the time of the yellow one is $\sim$ 5 ns, so it is a hit at the beginning, so we could think that we had a misalignment between channels, as we show in Fig. 19.

```
0x00000000: 0x0610 0x0050 0x0060 0x1bb7 0x0000 0x0000 0x0155 0x0000
0x00000010: 0x00de 0x5466 0x1408 0x5454 0x0408 0x0041 0x4431 0x231f
0x00000020: 0xc531 0x131c 0xc531 0x2318 0xc631 0x0318 0xc631 0x2310
0x00000030: 0x00d8 0x5475 0x1408 0x5474 0x0408 0x0041 0x87a1 0x1a16
0x00000040: 0x8461 0x2619 0x8561 0x2611 0x8661 0x0611 0x8461 0x0611
0x00000050: 0x00d2 0x5439 0x1408 0x5482 0x0408 0x0041 0xfbbe 0x1be2
0x00000060: 0xf87e 0x07e1 0xfbbe 0x07ee 0xf87e 0x3bee 0xf87e 0x3be1
0x00000070: 0x00cc 0x54cb 0x1408 0x549a 0x0408 0x0041 0xc6f1 0x1f13
0x00000080: 0x2409 0x3f10 0xc7f1 0x3f17 0xc409 0x009f 0xc7f1 0x1f17
```

Fig. 14: Example of non-valid event, with a non existent channel ID.

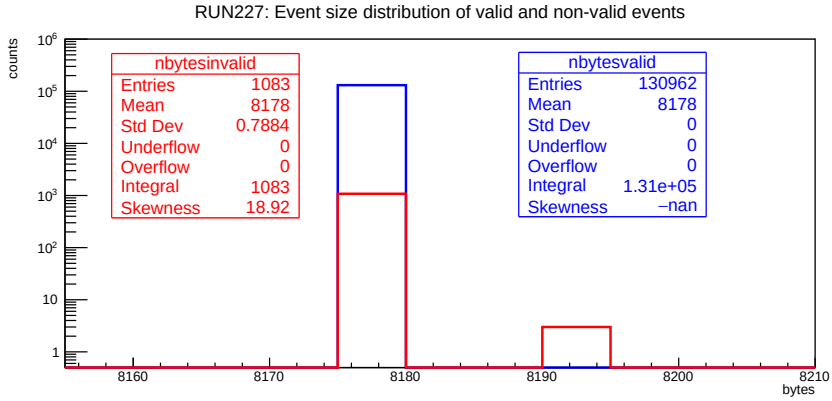


Fig. 15: Event size distribution of valid and non-valid events in RUN227. All the invalid events are peaked in 8.178 kB, but also the 16 extra bytes are marked as invalid.

5.3.3 Wrong channel ID

The last failure mode we have seen is an event with one hit with wrong channel ID. The events with this type of failure mode have hits with channels bigger than 96 (last channel). These are probably memory errors and, as we can see in Fig. 20, as soon as we have the wrong channel ID, we get a repeated pattern for the rest of the event.

There are few events with this type of failure mode (maximum 2 events per RUN), as we can see, for example, in RUN247 in Fig. 21. We think that there is something that should be fixed in the firmware, but it is now work in progress.

6 Analysis of logger and boardreader rate

We have studied rates reported by boardreader (DAQ input) and logger (DAQ output). We have done multiple runs, varying:

- number of ROCs : one or two;
- number of channels per ROC: 4-96;
- length of the time window 12.5 - 50 μs.

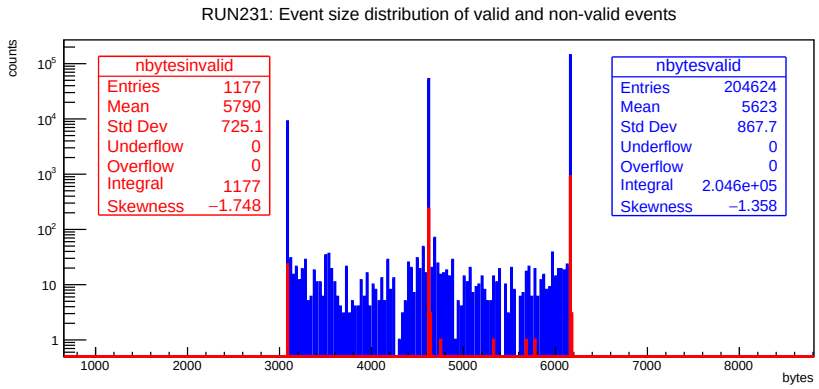


Fig. 16: Event size distribution of valid and non-valid events in RUN231. Both the valid and invalid events have a casual distribution that we don't understand yet.

RUN	nevents	TW [× 25ns]	R/O rate [MB/s]	nvalid [%]
227	132045	1000	0.94	0.82
231	205801	300	0.75	0.57
247	596522	500	2.5	0.19

Tab. 1: Resume of the operation condition and percentage of the non-validity of the events. As we can see, the valid events are less than the 1%, but the aim is to get no more non valid events.

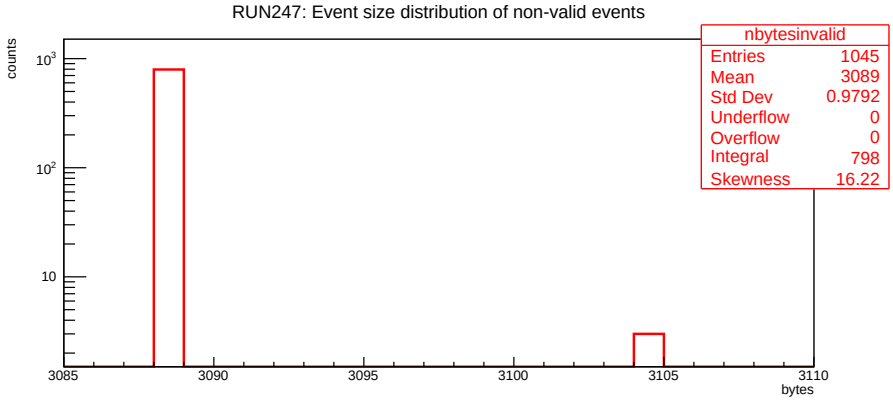


Fig. 17: Example of RUN with 16 extra bytes.

We expected that the rate of the boardreader would have been the same of the logger one, because in the middle we had no filter yet (Fig. 8). What we have seen in Fig. 22 is that $R_{br} < R_l$: we are working on understanding this.

6.1 Zoom on each RUN

We were interested to catch particular rate patterns in each RUN. As we can see in Fig. 23, in RUN281, the longest RUN we have succeeded in doing, there are repeated spikes: we have

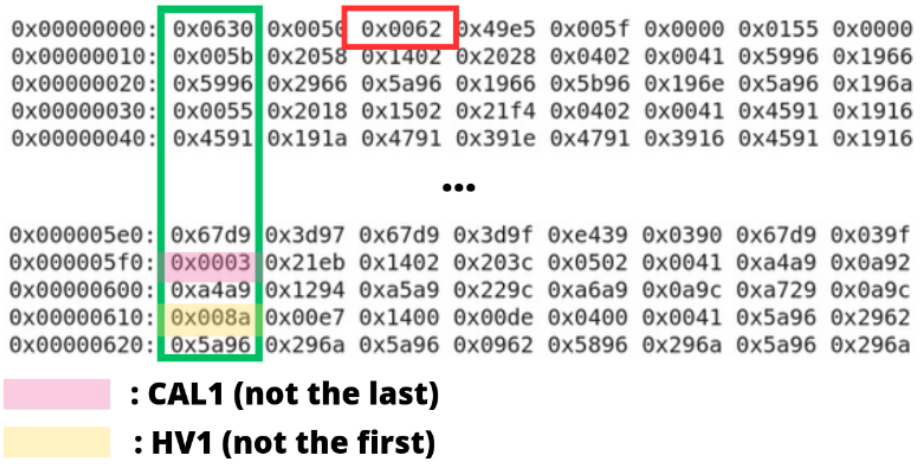


Fig. 18: Example of event with 16 extra bytes.

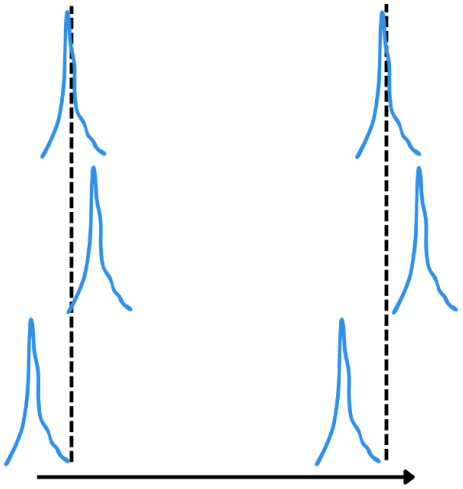


Fig. 19: Time misalignment between channels.

understood that these are the closing output files.

In RUN294 we read 2 ROCs with 96 channels each: we expected a double rate, but as we can see in Fig. 22, the rate dropped down by a factor of 3. Now it is work in progress.
Another thing we haven't understood yet is why we have acquired 500MB in 13 min, so 0.7MB/s, but ARTDAQ reports 7MB/s. There should be some errors on the ARTDAQ software and we need to understand why.

7 Validation of generator frequency

As we said in 5, we were generating *hits* in each channel at 31.29 MHz/(2^7+1) and 31.29 MHz/(2^9+1).
We wanted to validate the generator frequecy by looking at same channel hits in one event. We expected that the time difference between the i and $i-1$ pulse of each channel would be

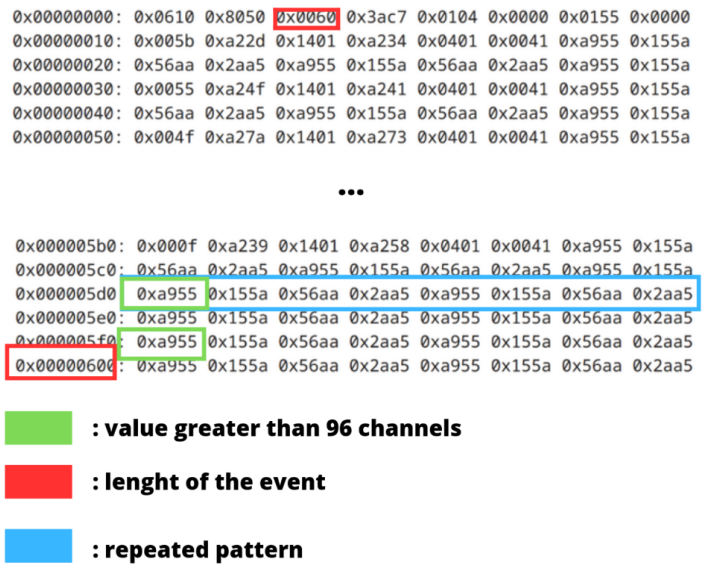


Fig. 20: An event with wrong channel ID failure mode.

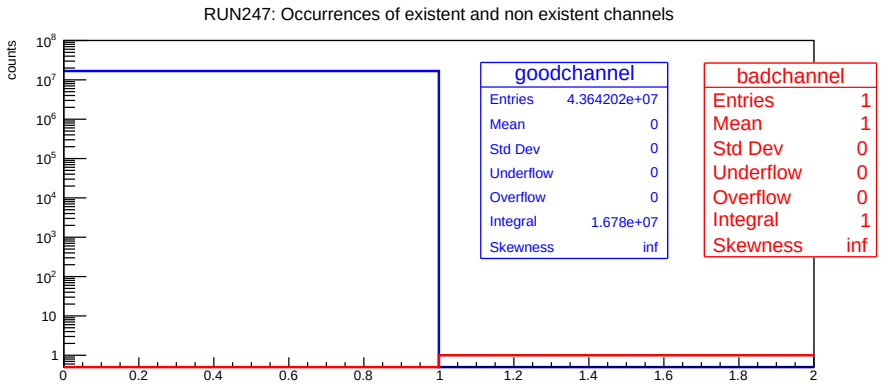


Fig. 21: Occurrences of correct and incorrect channel ID for RUN247.

exactly the inverse of the generator frequency.
In Fig. 24 and 25 we show the histograms of the i and $i-1$ hits time differences from two different RUNs.

We have validated generator frequency with an accuracy better than 20 ps, in order to understand the timing scale of our apparatus.

8 Occupancy: number of hits vs channel number

We have checked also how all the channels are filled, so how the buffer is filled. At first, we watched, as we can see in Fig. 26, the occupancy of channels ordered in ascending order, from 0 to 95. We have analyzed the occupancy in a mode of overflowing hit buffer and we

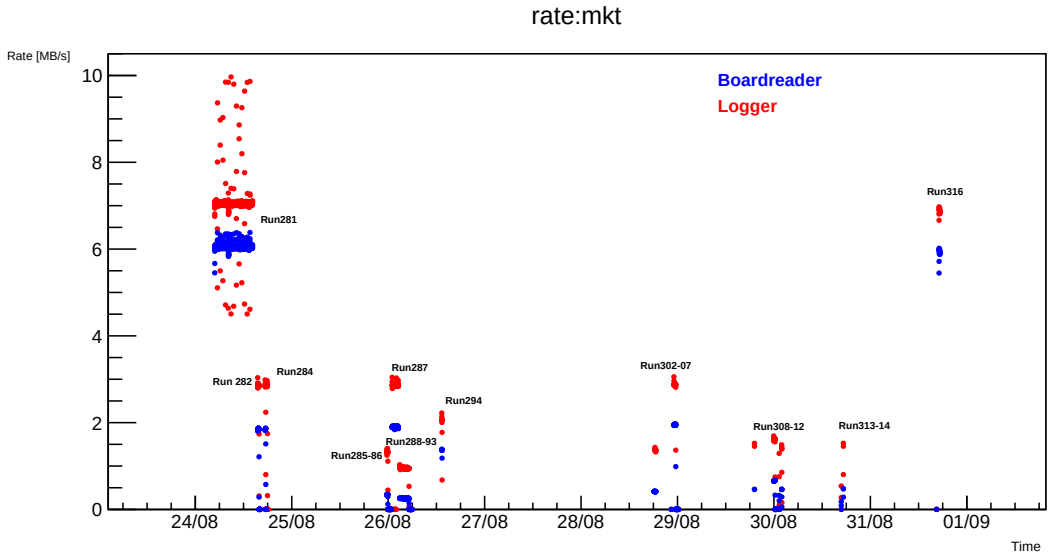


Fig. 22: Rate vs Time for the boardreader and for the logger.

have not seen a uniform distribution of number of hits vs channels: so we don't have the same occupancy for all channels. We tried to change the ordering of channels. We used the fill order we expected. We have done several RUNs with different time window and frequency, in order to understand if channels had a different order from which we expected. We show what we obtained in Fig. 28 and 27.

We expected that the buffer gradually fills with channels. On the contrary, we have seen that in RUN105023 the fill of channels is ordered, meanwhile in RUN105026 it is not. These RUNs have different frequencies, RUN105023 has a higher frequency than the RUN105026. RUN105023 fills immediately and it has a cutoff after channel 75 that is the 36th channel in the expexted order. Only the first 43 channels are filled and in this case we can think that the order of channels is correct. In RUN105026 there is a cutoff after channel 3 (the 48th channel filled) and then, after channel 5 (the 72th to be filled), we see that channels are filled again. We could suspect that, in the second FPGA, the first and the second lanes (groups of 24 channels each) are inverted with respect to the expected order.
Changing the frequency and the time window could help us understanding the correct order of channels. There are only two possible frequencies that we can use, so we preferred to change the time window.
In order to understand the correct order we have done a simulation that we will explain in section 10. In this simulation we specified all the features of our FPGAs and so we have studied the time differences between different channels as we will explain in section 9.

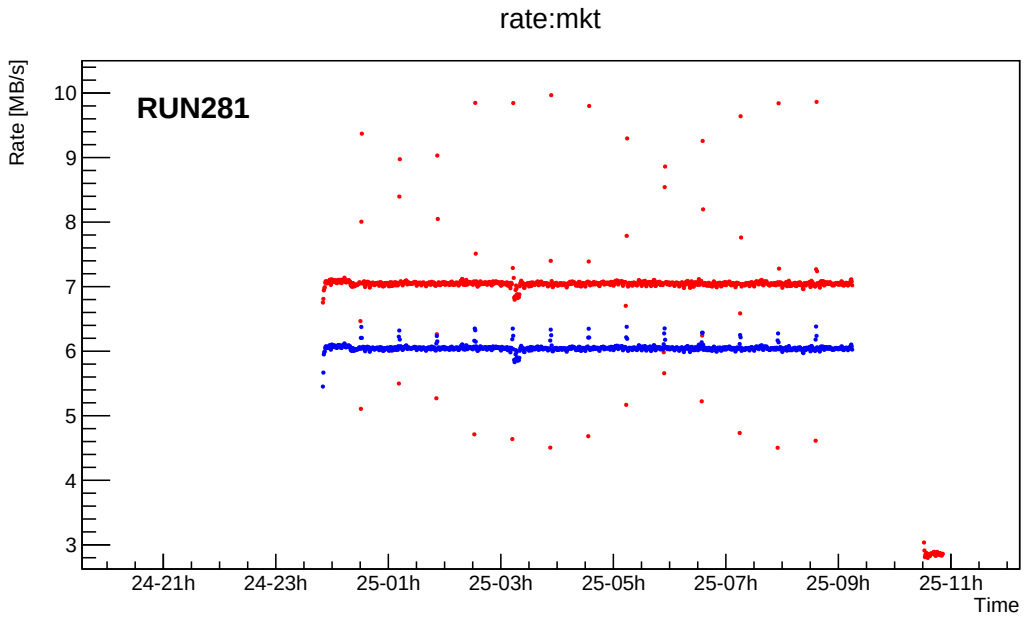


Fig. 23: Rate vs Time in RUN281.

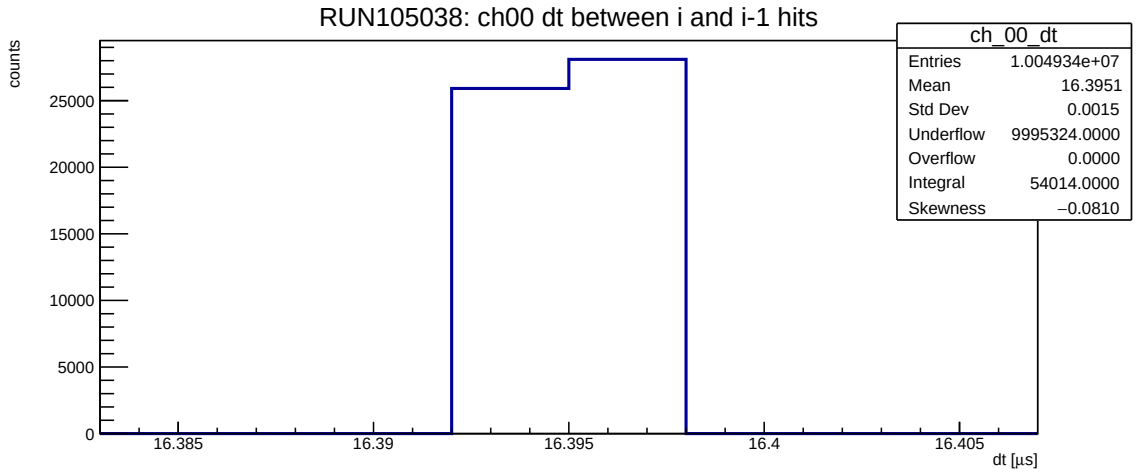


Fig. 24: Time difference between the i and $i-1$ pulse for RUN105038. As we can see the time difference is peaked in $\sim$16.4 μs, as we expected, because it is the inverse of one of the two generator frequencies.

9 Calibration of the time difference between channels

As we explained in the previous section, we were interested in studying the time difference between different channels, in order to reproduce our apparatus in the Monte Carlo simulation.

What we have done is saving the differences between all channels of one FPGA and one of

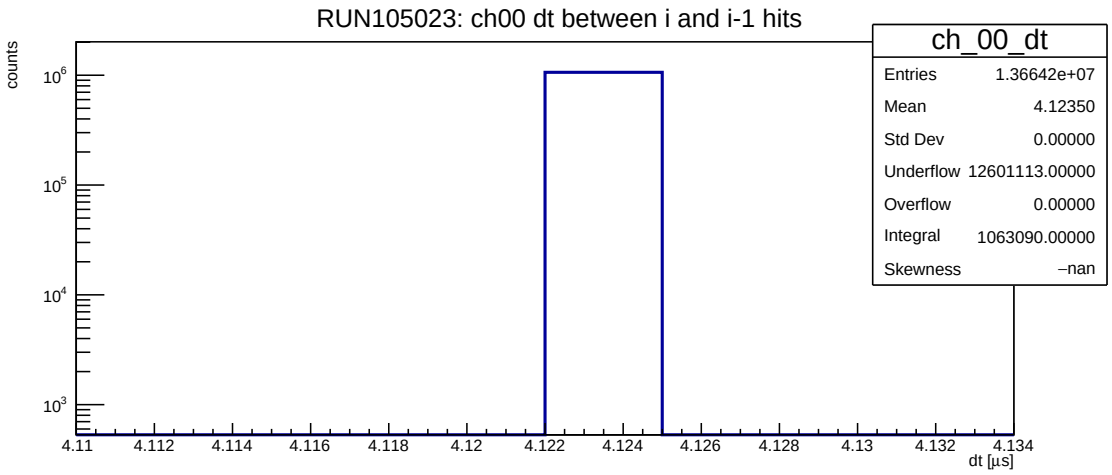


Fig. 25: Time difference between the i and $i-1$ pulse for RUN105023. As we can see the time difference is peaked in $\sim$4.1 μs, as we expected, because it is the inverse of one of the two generator frequencies.

the channel in the given FPGA. For example we chose channel 91 for the first FPGA and channel 94 for the second FPGA.
We have fitted that difference with the most natural distribution we could think to: a Gaussian.
All the absolute offsets between channels are included between 4 μs and 0.1 μs.
We show only two plots: the one that shows the time difference between channel 0 and channel 91, for the first FPGA, Fig. 29 and the one of the time difference between channel 44 and 94, for the second FPGA, Fig. 30.

10 Monte Carlo simulation

In order to make sure we fully understand our system we started doing a Monte Carlo Simulation of our DAQ system.
Given that the maximum number of hits per event is 255, the simulation is based on the following steps:

- in a time included between 0 μs and the inverse of the generator frequency, we create the first event (t_0), following a uniform distribution;
- we know that the hits from the same channel, in one event, are separated by the inverse of the generator frequency, so we create the second event (t_1), adding to t_0 this quantity;
- if the second pulse is cointained in the time window, we add this hit in the event we are building;
- as soon as we reach 255 hits we stop the fill.

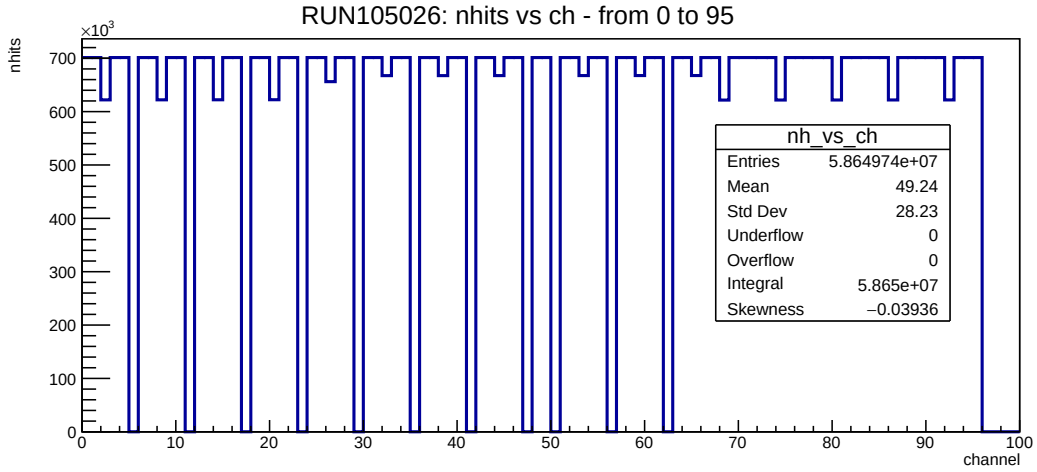


Fig. 26: Number of Hits vs Channel number for the RUN105026. In this case channels are ordered from 0 to 95 and we can see a distribution where not all the channels are filled equally and difficult to understand.

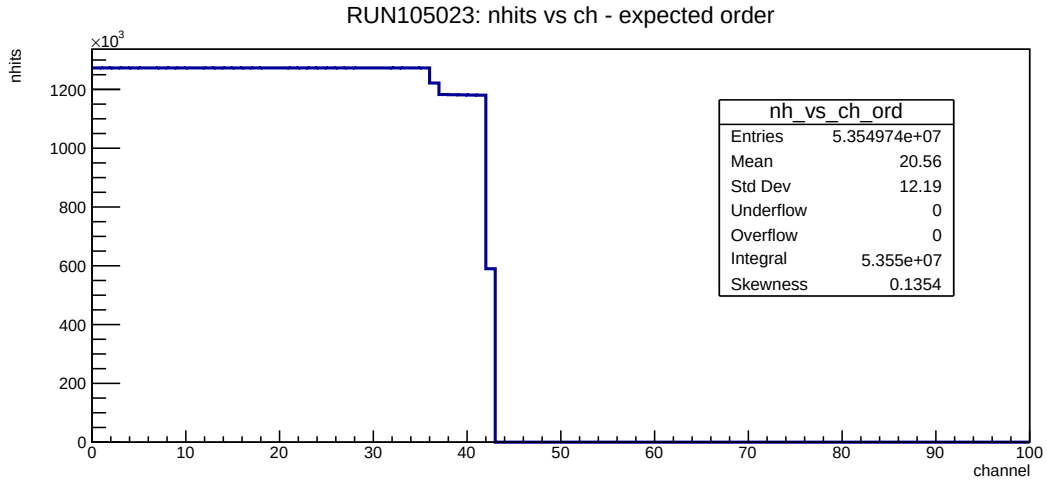


Fig. 27: Number of Hits vs Channel number for the RUN105026. In this case channels are filled in the order we expect.

We tried to understand the simulation when there aren't any delays, in order to understand how the cutoff looks like. We show this in Fig. 31. We can interpret this histogram: the first jump on the cutoff is due to the fact that the first channels are those where we collect 4 hits (first FPGA) and 3 hits (second FPGA) and the second ones those that collect 3 hits (second FPGA) and then 4 hits (first FPGA). Last group of channels is composed by those that collect 3 hits (first FPGA) and 3 hits (second FPGA). We have changed features, as the time window and generator frequency, to better compare each RUN with the simulation and we added FPGAs offsets and channel to channel offsets.
We show the occupancy histogram following the fill ordering of the Monte Carlo, that we call *estimated order of MC*. From the previous order, lane 2 of the second FPGA has changed

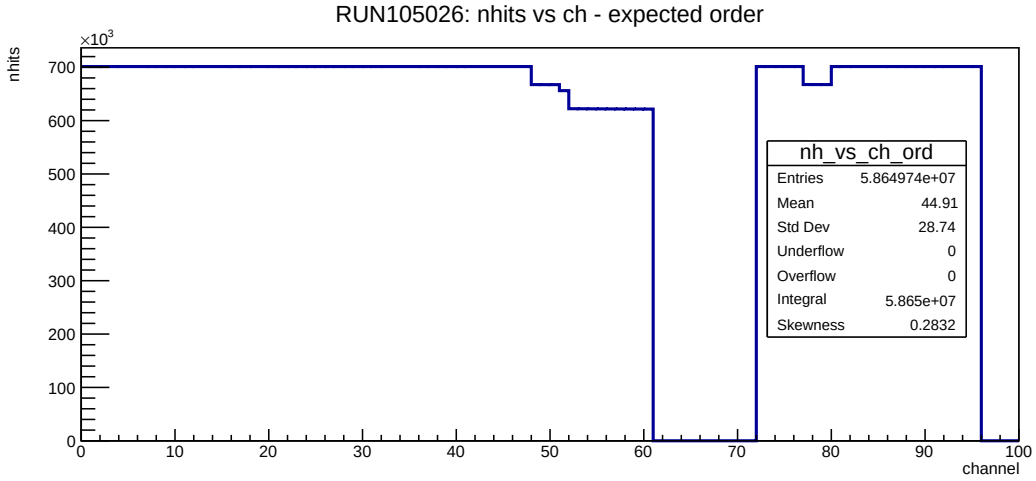


Fig. 28: Number of Hits vs Channel number for the RUN105026. In this case channels are filled in the order we expect.

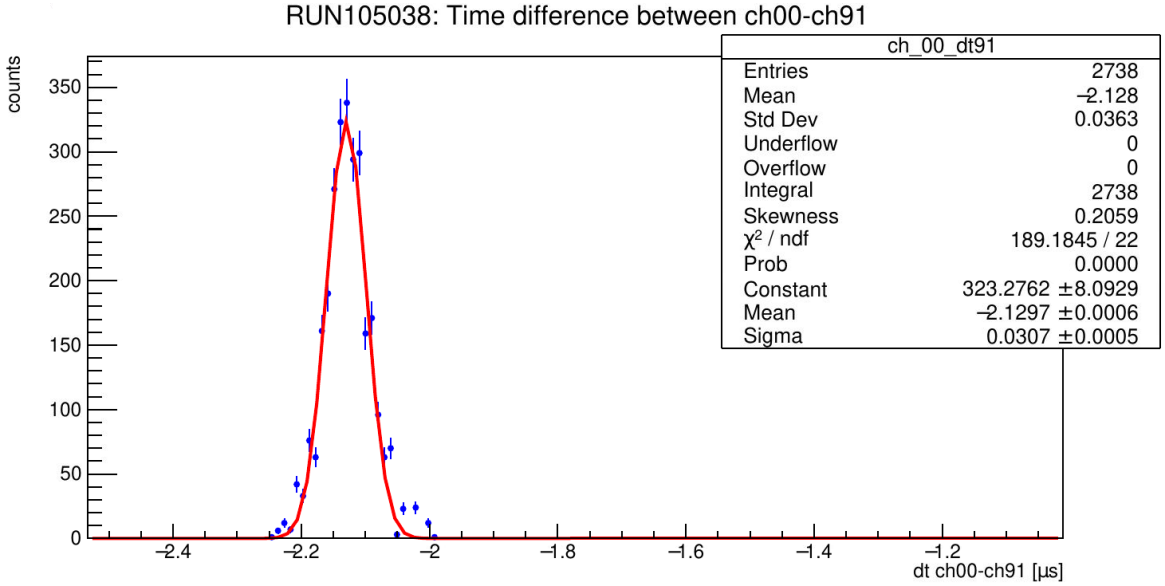


Fig. 29: Time difference between ch00 and ch91 (first channel of first FPGA). We have fitted with a Gaussian.

with lane 1 of the second FPGA and also some internal channels have changed position, in order to have a decreasing number of hits per channel and a cutoff. We show this new ordering in Fig. 32. We understood that as soon as we add delays and offsets in Monte Carlo, jumps appear, not so many because the delays are very small with respect to the time window (ns compared to μs), so in Fig. 32, we see only one jump more than in Fig. 31, but the hits logic is the same.

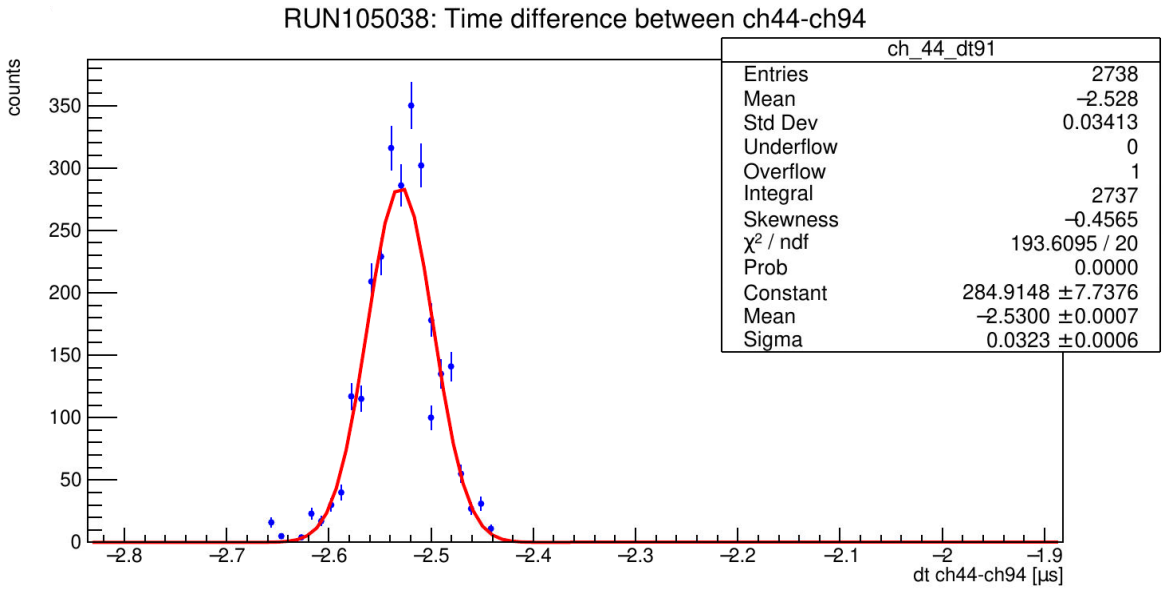


Fig. 30: Time difference between ch44 and ch94 (first channel of second FPGA). We have fitted with a Gaussian.

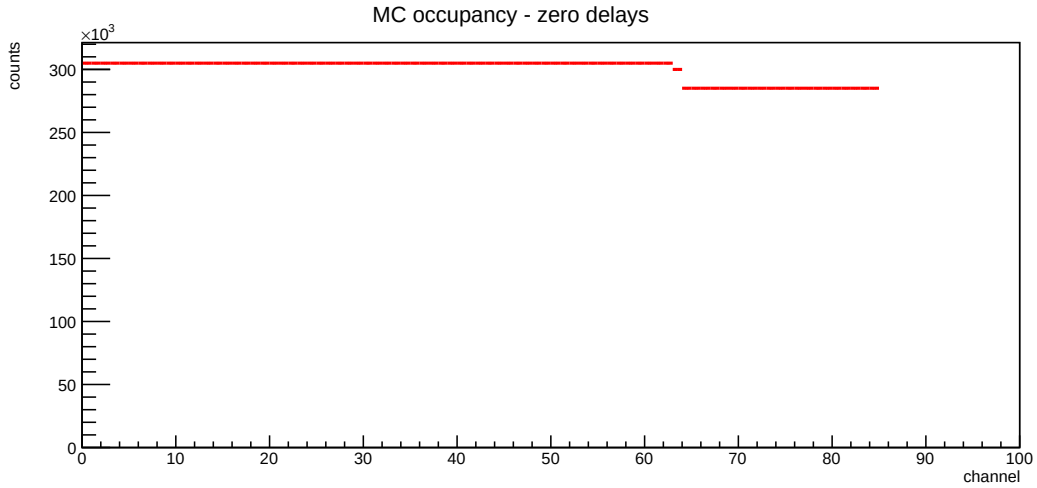


Fig. 31: Zero delays (both the FPGAs offsets and the channel to channel delays) occupancy. Channels are ordered following the MC occupancy.

11 Conclusions

In this Internship, we studied mainly the tracker and its readout. We used a generator to send pulses and we tried to understand the output and non-output of the DTC, in order to validate the event and run format and all the failure modes. We were reading 1 ROC (96 channels), which is the equivalent of one panel. We varied the generator event window that changed the number of hits per event. We could change the frequency from ∼250 kHz and ∼60 kHz. We studied the structure of an event in a run and its features, but also event size distribution, varying the time window, not understanding what happens when we reduce it.

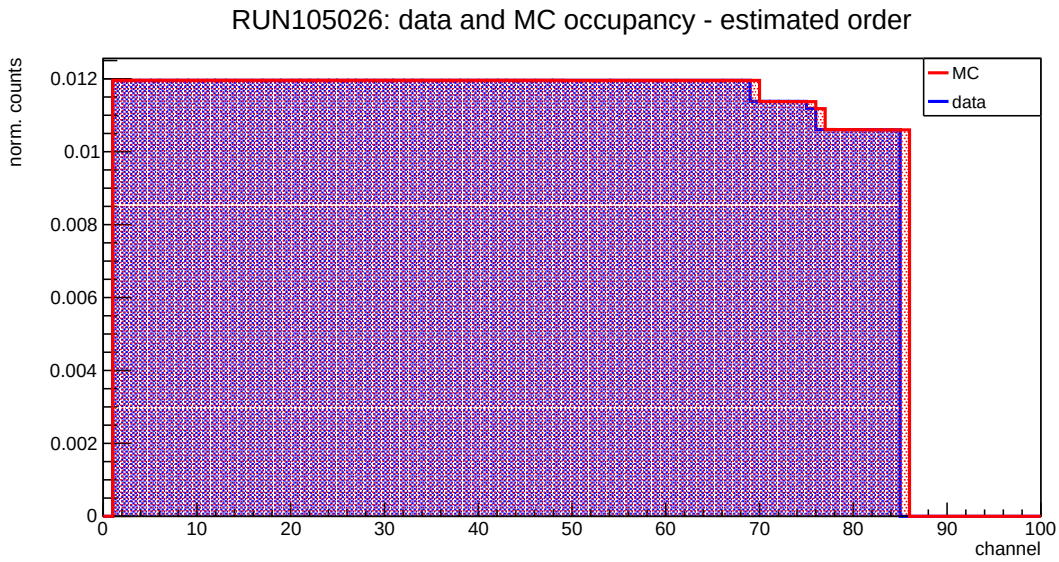


Fig. 32: Comparison between Monte Carlo (in red) and data (in blue) of channels occupancy.

We collected three types of failure mode: extra 16 bytes events, events marked as non-valid, non existent channel IDs.
We analyzed the logger and boardreader rate, changing ROCs number, channels per ROC and time window, observing some failures in OTSDAQ and not understanding why improving the number of ROCs, the rate drops down.
We validated the generator frequency studying the time difference between hits of the same channel, in the same event.
We have done histograms of the number of hits in function of the channel number, undestanding that the fill ordering is different from the one expected, so we simulated ROCs readout and reproducing plots, in order to understand which is the correct order. To do that we have analyzed the time differences between channels.
We are now able to read 1MB/s (1 DTC and 1 ROC) and our goal is to read 600MB/s with one DTC and in late fall-early winter we will do Vertical Slice Test.

Glossary

ARTDAQ An event building and filtering software toolset for DAQ. It is the toolkit to data transfer, event buiding, event reconstruction and analysis. 6

CFO Command FanOut (module). 6

DR Data Request Packets generated by DTCs. 7

DRAC Digitizer Readout & Assembler Controller. 5

DTC Data Transfer Controller (module). 6

EBS Event Building Switch. 6

Event Window Period of the time defined as mu2e event. 7

HB A packet sent to all ROCs before each Microbunch. 6

Microbunch 250ns proton pulse delivered to target (same as Proton Pulse and Pulse). 23

OTSDAQ Off The Shell Data AcQuisition. It is the online DAQ software framework of Mu2e. 6

ROC ReadOut Controller. 6

TDAQ Trigger Data AcQuisition and it is the software used to monitor, select, validate, calibrate experimental data of Mu2e Trigger . 6

XDAQ OTSDAQ uses a run control system using the data acquisition software XDAQ implemented for the development and calibration-mode runs at CMS. 6

References

[al21a] Abdi et al. "Mu2e Run I Sensitivity Projections for the Neutrinoless $\mu^- \to e^-$ Conversion Search in Aluminum". In: (2021).

[al21b] Gioiosa et al. "Slow control and data acquisition development in the Mu2e experiment". In: (2021).

[Lee16] MyeongJae Lee. "The Straw-tube Tracker for the Mu2e Experiment". In: (2016).