



Universiteit Gent
Faculteit Wetenschappen
Vakgroep Fysica en Sterrenkunde

² No title yet

³ No sub-title neither, obviously...

⁴ Alexis Fagot

5



Thesis to obtain the degree of
Doctor of Philosophy in Physics
Academic years 2012-2017





Universiteit Gent
Faculteit Wetenschappen
Vakgroep Fysica en Sterrenkunde

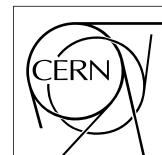
Promotoren: Dr. Michael Tytgat
Prof. Dr. Dirk Ryckbosch

Universiteit Gent
Faculteit Wetenschappen
Vakgroep Fysica en Sterrenkunde
Proeftuinstraat 86, B-9000 Gent, België
Tel.: +32 9 264.65.28
Fax.: +32 9 264.66.97

17



Thesis to obtain the degree of
Doctor of Philosophy in Physics
Academic years 2012-2017



Acknowledgements

¹⁹ Ici on remerciera tous les gens que j'ai pu croiser durant cette aventure et qui m'ont permis de passer
²⁰ un bon moment

²¹ *Gent, ici la super date de la mort qui tue de la fin d'écriture*
²² *Alexis Fagot*

Table of Contents

24	Acknowledgements	i
25	Nederlandse samenvatting	xvii
26	English summary	xix
27	1 Introduction	1-1
28	1.1 A story of High Energy Physics	1-1
29	1.2 Organisation of this study	1-1
30	2 Investigating the TeV scale	2-1
31	2.1 The Standard Model of Particle Physics	2-1
32	2.2 The Large Hadron Collider and the Compact Muon Solenoid	2-1
33	2.3 Muon Phase-II Upgrade	2-1
34	3 Amplification processes in gaseous detectors	3-1
35	3.1 Signal formation	3-1
36	3.2 Gas transport parameters	3-1
37	4 Resistive Plate Chambers	4-1
38	4.1 Principle	4-1
39	4.2 Rate capability of Resistive Plate Chambers	4-1
40	4.3 High time resolution	4-1
41	4.4 Resistive Plate Chambers at CMS	4-1
42	4.4.1 Overview	4-1
43	4.4.2 The present RPC system	4-2
44	4.4.3 Pulse processing of CMS RPCs	4-3
45	5 Longevity studies and Consolidation of the present CMS RPC subsystem	5-1
46	5.1 Testing detectors under extreme conditions	5-1
47	5.1.1 The Gamma Irradiation Facilities	5-3
48	5.1.1.1 GIF	5-3
49	5.1.1.2 GIF++	5-5
50	5.2 Preliminary tests at GIF	5-7
51	5.2.1 Resistive Plate Chamber test setup	5-7
52	5.2.2 Data Acquisition	5-9
53	5.2.3 Geometrical acceptance of the setup layout to cosmic muons	5-9
54	5.2.3.1 Description of the simulation layout	5-10
55	5.2.3.2 Simulation procedure	5-12
56	5.2.3.3 Results	5-13
57	5.2.4 Photon flux at GIF	5-13

58	5.2.4.1	Expectations from simulations	5-13
59	5.2.4.2	Dose measurements	5-18
60	5.2.5	Results and discussions	5-19
61	5.3	Longevity tests at GIF++	5-20
62	5.3.1	Description of the Data Acquisition	5-23
63	5.3.2	RPC current, environmental and operation parameter monitoring	5-24
64	5.3.3	Measurement procedure	5-25
65	5.3.4	Longevity studies results	5-25
66	6	Investigation on high rate RPCs	6-1
67	6.1	Rate limitations and ageing of RPCs	6-1
68	6.1.1	Low resistivity electrodes	6-1
69	6.1.2	Low noise front-end electronics	6-1
70	6.2	Construction of prototypes	6-1
71	6.3	Results and discussions	6-1
72	7	Conclusions and outlooks	7-1
73	7.1	Conclusions	7-1
74	7.2	Outlooks	7-1
75	A	A data acquisition software for CAEN VME TDCs	A-1
76	A.1	GIF++ DAQ file tree	A-1
77	A.2	Usage of the DAQ	A-2
78	A.3	Description of the readout setup	A-3
79	A.4	Data read-out	A-3
80	A.4.1	V1190A TDCs	A-4
81	A.4.2	DataReader	A-6
82	A.4.3	Data quality flag	A-10
83	A.5	Communications	A-12
84	A.5.1	V1718 USB Bridge	A-13
85	A.5.2	Configuration file	A-13
86	A.5.3	WebDCS/DAQ intercommunication	A-17
87	A.5.4	Example of inter-process communication cycle	A-18
88	A.6	Software export	A-18
89	B	Details on the offline analysis package	B-1
90	B.1	GIF++ Offline Analysis file tree	B-1
91	B.2	Usage of the Offline Analysis	B-2
92	B.2.1	Output of the offline tool	B-3
93	B.2.1.1	ROOT file	B-3
94	B.2.1.2	CSV files	B-5
95	B.3	Analysis inputs and information handling	B-6
96	B.3.1	Dimensions file and IniFile parser	B-6
97	B.3.2	TDC to RPC link file and Mapping	B-7
98	B.4	Description of GIF++ setup within the Offline Analysis tool	B-9
99	B.4.1	RPC objects	B-9
100	B.4.2	Trolley objects	B-10
101	B.4.3	Infrastructure object	B-11
102	B.5	Handeling of data	B-12
103	B.5.1	RPC hits	B-13

104	B.5.2 Clusters of hits	B-14
105	B.6 DAQ data Analysis	B-15
106	B.6.1 Determination of the run type	B-16
107	B.6.2 Beam time window calculation for efficiency runs	B-17
108	B.6.3 Data loop and histogram filling	B-18
109	B.6.4 Results calculation	B-19
110	B.6.4.1 Rate normalisation	B-19
111	B.6.4.2 Rate and activity	B-21
112	B.6.4.3 Strip masking tool	B-23
113	B.6.4.4 Output CSV files filling	B-26
114	C Structure of the hybrid simulation software	C-1
115	C.1 Introduction	C-1

List of Figures

117	2.1	Absorbed dose in the CMS cavern after an integrated luminosity of 3000 fb. R is the transverse distance from the beamline and Z is the distance along the beamline from the Interaction Point at Z=0.	2-2
118			
119			
120	2.2	A quadrant of the muon system, showing DTs (yellow), RPCs (light blue), and CSCs (green). The locations of new forward muon detectors for Phase-II are contained within the dashed box and indicated in red for GEM stations (ME0, GE1/1, and GE2/1) and dark blue for improved RPC (iRPC) stations (RE3/1 and RE4/1).	2-2
121			
122			
123			
124	2.3	RMS of the multiple scattering displacement as a function of muon p_T for the proposed forward muon stations. All of the electromagnetic processes such as bremsstrahlung and magnetic field effect are included in the simulation.	2-3
125			
126			
127	4.1	Signals from the RPC strips are shaped by the FEE described on Figure 4.1a. Output LVDS signals are then read-out by a TDC module connected to a computer or converted into NIM and sent to scalers. Figure 4.1b describes how these converted signals are put in coincidence with the trigger.	4-3
128			
129			
130			
131	4.2	Description of the principle of a CFD. A comparison of threshold triggering (left) and constant fraction triggering (right) is shown in Figure 4.2a. Constant fraction triggering is obtained thanks to zero-crossing technique as explained in Figure 4.2b. The signal arriving at the input of the CFD is split into three components. A first one is delayed and connected to the inverting input of a first comparator. A second component is connected to the noninverting input of this first comparator. A third component is connected to the noninverting input of another comparator along with a threshold value connected to the inverting input. Finally, the output of both comparators is fed through an AND gate.	4-4
132			
133			
134			
135			
136			
137			
138			
139			
140	5.1	(5.1a) Extrapolation from 2016 data of single hit rate per unit area in the barrel region. (5.1b) Extrapolation from 2016 data of single hit rate per unit area in the endcap region.	5-2
141			
142			
143	5.2	Background Fluka simulation compared to 2016 Data at $L = 10^{34} \text{ cm}^{-2} \cdot \text{s}^{-1}$ in the fourth endcap disk region. A mismatch in between simulation and data can be observed. [To be understood.]	5-3
144			
145			
146	5.3	Layout of the test beam zone called X5c GIF at CERN. Photons from the radioactive source produce a sustained high rate of random hits over the whole area. The zone is surrounded by 8 m high and 80 cm thick concrete walls. Access is possible through three entry points. Two access doors for personnel and one large gate for material. A crane allows installation of heavy equipment in the area.	5-4
147			
148			
149			
150			
151	5.4	^{137}Cs decays by β^- emission to the ground state of ^{137}Ba (BR = 5.64%) and via the 662 keV isomeric level of ^{137}Ba (BR = 94.36%) whose half-life is 2.55 min.	5-5
152			

153	5.5	Floor plan of the GIF++ facility. When the facility downstream of the GIF++ takes electron beam, a beam pipe is installed along the beam line (z-axis). The irradiator can be displaced laterally (its center moves from $x = 0.65$ m to 2.15 m), to increase the distance to the beam pipe.	5-5
154			
155			
156			
157	5.6	Simulated unattenuated current of photons in the xz plane (Figure 5.6a) and yz plane (Figure 5.6b) through the source at $x = 0.65$ m and $y = 0$ m. With angular correction filters, the current of 662 keV photons is made uniform in xy planes.	5-6
158			
159			
160	5.7	Description of the RPC setup. Dimensions are given in mm. A tent containing RPCs is placed at 1720 mm from the source container. The source is situated in the center of the container. RE-4-2-BARC-161 chamber is 160 mm inside the tent. This way, the distance between the source and the chambers plan is 2060 mm. Figure 5.7a provides a side view of the setup in the xz plane while Figure 5.7b shows a top view in the yz plane.	5-7
161			
162			
163			
164			
165			
166	5.8	RE-4-2-BARC-161 chamber is inside the tent as described in Figure 5.7. In the top right, the two scintillators used as trigger can be seen. This trigger system has an inclination of 10° relative to horizontal and is placed above half-partition B2 of the RPCs. PMT electronics are shielded thanks to lead blocks placed in order to protect them without stopping photons from going through the scintillators and the chamber.	5-8
167			
168			
169			
170			
171	5.9	Hit distributions over all 3 partitions of RE-4-2-BARC-161 chamber is showed on these plots. Top, middle and bottom figures respectively correspond to partitions A, B, and C. These plots show that some events still occur in other half-partitions than B2, which corresponds to strips 49 to 64, in front of which the trigger is placed, contributing to the inefficiency of detection of cosmic muons. In the case of partitions A and C, the very low amount of data can be interpreted as noise. On the other hand, it is clear that a little portion of muons reach the half-partition B1, corresponding to strips 33 to 48.	5-9
172			
173			
174			
175			
176			
177			
178			
179	5.10	Results are derived from data taken on half-partition B2 only. On the 18 th of June 2014, data has been taken on chamber RE-2-BARC-161 at building 904 (Prevessin Site) with cosmic muons providing us a reference efficiency plateau of $(97.54 \pm 0.15)\%$ represented by a black curve. A similar measurement has been done at GIF on the 21 st of July with the same chamber giving a plateau of $(78.52 \pm 0.94)\%$ represented by a red curve.	5-10
180			
181			
182			
183			
184			
185	5.11	Representation of the layout used for the simulations of the test setup. The RPC is represented as a yellow trapezoid while the two scintillators as blue cuboids looking at the sky. A green plane corresponds to the muon generation plane within the simulation. Figure 5.7a shows a global view of the simulated setup. Figure 5.7b shows a zommed view that allows to see the 2 scintillators as well as the full RPC plane.	5-11
186			
187			
188			
189			
190	5.12	γ flux $F(D)$ is plot using values from table 5.1. As expected, the plot shows similar attenuation behaviours with increasing distance for each absorption factors.	5-14
191			
192	5.13	Figure 5.13a shows the linear approximation fit done via formulae 5.7 on data from table 5.2. Figure 5.13b shows a comparison of this model with the simulated flux using a and b given in figure 5.13a in formulae 5.4 and the reference value $D_0 = 50\text{cm}$ and the associated flux for each absorption factor F_0^{ABS} from table 5.1	5-16
193			
194			
195			
196	5.14	Dose measurements has been done in a plane corresponding to the tents front side. This plan is 1900 mm away from the source. As explained in the first chapter, a lens-shaped lead filter provides a uniform photon flux in the vertical plan orthogonal to the beam direction. If the second line of measured fluxes is not taken into account because of lower values due to experimental equipments in the way between the source and the tent, the uniformity of the flux is well showed by the results.	5-18
197			
198			
199			
200			
201			

202	5.15	5-19
203	5.16 Evolution of the maximum efficiency for RE2 (5.16a) and RE4 (5.16b) chambers	
204	with increasing extrapolated γ rate per unit area at working point. Both irradiated	
205	(blue) and non irradiated (red) chambers are shown.	5-21
206	5.17 Evolution of the working point for RE2 (5.17a) and RE4 (5.17b) with increasing	
207	extrapolated γ rate per unit area at working point. Both irradiated (blue) and non	
208	irradiated (red) chambers are shown.	5-21
209	5.18 Evolution of the maximum efficiency at HL-LHC conditions, i.e. a background hit	
210	rate per unit area of 300 Hz/cm ² , with increasing integrated charge for RE2 (5.18a)	
211	and RE4 (5.18b) detectors. Both irradiated (blue) and non irradiated (red) chambers	
212	are shown. The integrated charge for non irradiated detectors is recorded during test	
213	beam periods and stays small with respect to the charge accumulated in irradiated	
214	chambers.	5-22
215	5.19 Comparison of the efficiency sigmoid before (triangles) and after (circles) irradiation	
216	for RE2 (5.19a) and RE4 (5.19b) detectors. Both irradiated (blue) and non irradiated	
217	(red) chambers are shown.	5-22
218	5.20 Evolution of the Bakelite resistivity for RE2 (5.20a) and RE4 (5.20b) detectors. Both	
219	irradiated (blue) and non irradiated (red) chambers are shown.	5-23
220	5.21 Evolution of the noise rate per unit area for the irradiated chamber RE2-2-BARC-9	
221	only.	5-23
222	A.1 (A.1a) View of the front panel of a V1190A TDC module [10]. (A.1b) View of the	
223	front panel of a V1718 Bridge module [11]. (A.1c) View of the front panel of a 6U	
224	6021 VME crate [12].	A-3
225	A.2 Module V1190A <i>Trigger Matching Mode</i> timing diagram [10].	A-4
226	A.3 Structure of the ROOT output file generated by the DAQ. The 5 branches (<code>EventNumber</code> ,	
227	<code>number_of_hits</code> , <code>Quality_flag</code> , <code>TDC_channel</code> and <code>TDC_TimeStamp</code>) are visible on	
228	the left panel of the ROOT browser. On the right panel is visible the histogram cor-	
229	responding to the variable <code>nHits</code> . In this specific example, there were approximately	
230	50k events recorded to measure the gamma irradiation rate on the detectors. Each	
231	event is stored as a single entry in the <code>TTree</code>	A-10
232	A.4 The effect of the quality flag is explained by presenting the content of <code>TBranch</code>	
233	<code>number_of_hits</code> of a data file without <code>Quality_flag</code> in Figure A.4a and the con-	
234	tent of the same <code>TBranch</code> for data corresponding to a <code>Quality_flag</code> where all TDCs	
235	were labelled as <code>GOOD</code> in Figure A.4b taken with similar conditions. It can be noted	
236	that the number of entries in Figure A.4b is slightly lower then in Figure A.4a due	
237	to the excluded events.	A-12
238	A.5 Using the same data as previously showed in Figure A.4, the effect of the quality	
239	flag is explained by presenting the reconstructed hit multiplicity of a data file with-	
240	out <code>Quality_flag</code> in Figure A.5a and the reconstructed content of the same RPC	
241	partition for data corresponding to a <code>Quality_flag</code> where all TDCs were labelled as	
242	<code>GOOD</code> in Figure A.5b taken with similar conditions. The artificial high content of bin	
243	0 is completely suppressed.	A-12
244	A.6 WebDCS DAQ scan page. On this page, shifters need to choose the type of scan	
245	(Rate, Efficiency or Noise Reference scan), the gamma source configuration at the	
246	moment of data taking, the beam configuration, and the trigger mode. These in-	
247	formation will be stored in the DAQ ROOT output. Are also given the minimal	
248	measurement time and waiting time after ramping up of the detectors is over before	
249	starting the data acquisition. Then, the list of HV points to scan and the number of	
250	triggers for each run of the scan are given in the table underneath.	A-14

- 251 B.1 Example of expected hit time distributions in the cases of efficiency (Figure B.1a)
252 and noise/gamma rate per unit area (Figure B.1b) measurements as extracted from
253 the raw ROOT files. The unit along the x-axis corresponds to ns. The fact that
254 "the" muon peak is not well defined in Figure B.1a is due to the contribution of all
255 the RPCs being tested at the same time that don't necessarily have the same signal
256 arrival time. Each individual peak can have an offset with the ones of other detectors.
257 The inconsistancy in the first 100 ns of both time distributions is an artefact of the
258 TDCs and are systematically rejected during the analysis. B-16
- 259 B.2 The effect of the quality flag is explained by presenting the reconstructed hit multi-
260 plicity of a data file without `Quality_flag`. The artificial high content of bin 0 is the
261 effect of corrupted data. B-19
- 262 B.3 Display of the masking tool page on the webDCS. The window on the left allows the
263 shifter to edit `ChannelsMapping.csv`. To mask a channel, it only is needed to set the
264 3rd field corresponding to the strip to mask to 0. It is not necessary for older mapping
265 file formats to add a 1 for each strip that is not masked as the code is versatile and
266 the default behaviour is to consider missing mask fields as active strips. The effect
267 of the mask is directly visible for noisy channels as the corresponding bin turns red.
268 The global effect of masking strips will be an update of the rate value showed on the
269 histogram that will take into consideration the rejected channels. B-24

List of Tables

270

271	5.1	Total photon flux ($E\gamma \leq 662$ keV) with statistical error predicted considering a	
272		^{137}Cs activity of 740 GBq at different values of the distance D to the source along	
273		the x-axis of irradiation field [6].	5-13
274	5.2	Correction factor c is computed thanks to formulae 5.5 taking as reference $D_0 =$	
275		50 cm and the associated flux F_0^{ABS} for each absorption factor available in table 5.1.	5-15
276	5.3	The data at D_0 in 1997 is taken from [6]. In a second step, using Equations 5.8	
277		and 5.9, the flux at D can be estimated in 1997. Then, taking into account the	
278		attenuation of the source activity, the flux at D can be estimated at the time of the	
279		tests in GIF in 2014. Finally, assuming a sensitivity of the RPC to γ $s = 2 \cdot 10^{-3}$,	
280		an estimation of the hit rate per unit area is obtained.	5-17
281	A.1	Inter-process communication cycles in between the webDCS and the DAQ through	
282		file string signals.	A-19

283

List of Acronyms

284

List of Acronyms

285

286

A

288

289 AFL

Almost Full Level

290

291

B

293

294 BARC

Bhabha Atomic Research Centre

295 BLT

Block Transfer

296 BR

Branching Ratio

297

298

C

300

301 CAEN

Costruzioni Apparecchiature Elettroniche Nucleari S.p.A.

302 CERN

European Organization for Nuclear Research

303 CFD

Constant Fraction Discriminator

304 CMS

Compact Muon Solenoid

305 CSC

Cathode Strip Chamber

306

307

D

309

310 DAQ

Data Acquisition

311 DCS

Detector Control Software

312 DQM

Data Quality Monitoring

313 DT

Drift Tube

314

315

F

317

318	FEE	Front-End Electronics
319	FEB	Front-End Board
320		
321	G	
323		
324	GE-/-	Find a good description
325	GE1/1	Find a good description
326	GE2/1	Find a good description
327	GEANT	GEometry ANd Tracking - a series of software toolkit platforms developed by CERN
328		
329	GEM	Gas Electron Multiplier
330	GIF	Gamma Irradiation Facility
331	GIF++	new Gamma Irradiation Facility
332		
333	H	
334		
335		
336	HL-LHC	High Luminosity LHC
337	HV	High Voltage
338		
339	I	
340		
341		
342	iRPC	improved RPC
343	IRQ	Interrupt Request
344		
345	L	
346		
347		
348	LHC	Large Hadron Collider
349	LS1	First Long Shutdown
350	LS3	Third Long Shutdown
351	LV	Low Voltage
352	LVDS	Low-Voltage Differential Signaling
353		
354	M	
355		
356		
357	MC	Monte Carlo
358	MCNP	Monte Carlo N-Particle
359	ME-/-	Find good description
360	ME0	Find good description

361		
362	N	
363		
364		
365	NIM	Nuclear Instrumentation Module logic signals
366		
367	P	
368		
369		
370	PMT	PhotoMultiplier Tube
371		
372	R	
373		
374		
375	RE-/-	Find a good description
376	RE2/2	Find a good description
377	RE3/1	Find a good description
378	RE3/2	Find a good description
379	RE4/1	Find a good description
380	RE4/2	Find a good description
381	RE4/3	Find a good description
382	RMS	Root Mean Square
383	ROOT	a framework for data processing born at CERN
384	RPC	Resistive Plate Chamber
385		
386	S	
387		
388		
389	SPS	Super Proton Synchrotron
390		
391	T	
392		
393		
394	TDC	Time-to-Digital Converter
395		
396	W	
397		
398		
399	webDCS	Web Detector Control System

401

Nederlandse samenvatting –Summary in Dutch–

402

403 Le resume en Neerlandais (j'aurais peut-être de apprendre la langue juste pour ca...).

English summary

⁴⁰⁵ Le meme résume mais en Anglais (on commencera par la hein!).

1

Introduction

406

407

⁴⁰⁸ **1.1 A story of High Energy Physics**

⁴⁰⁹ **1.2 Organisation of this study**

2

410

411

Investigating the TeV scale

412 2.1 The Standard Model of Particle Physics

413 2.2 The Large Hadron Collider and the Compact Muon Solenoid

414 2.3 Muon Phase-II Upgrade

415 After the more than two years lasting First Long Shutdown (LS1), the Large Hadron Collider (LHC)
416 delivered its very first Run-II proton-proton collisions early 2015. LS1 gave the opportunity to the
417 LHC and to the its experiments to undergo upgrades. The accelerator is now providing collisions
418 at center-of-mass energy of 13 TeV and bunch crossing rate of 40 MHz, with a peak luminosity
419 exceeding its design value. During the first and upcoming second LHC Long Shutdown, the Compact
420 Muon Solenoid (CMS) detector is also undergoing a number of upgrades to maintain a high system
421 performance [1].

422 From the LHC Phase-2 or High Luminosity LHC (HL-LHC) period onwards, i.e. past the Third
423 Long Shutdown (LS3), the performance degradation due to integrated radiation as well as the average
424 number of inelastic collisions per bunch crossing, or pileup, will rise substantially and become a
425 major challenge for the LHC experiments, like CMS that are forced to address an upgrade program
426 for Phase-II [2]. Simulations of the expected distribution of absorbed dose in the CMS detector
427 under HL-LHC conditions, show in figure 5.14 that detectors placed close to the beamline will have
428 to withstand high irradiation, the radiation dose being of the order of a few tens of Gy.

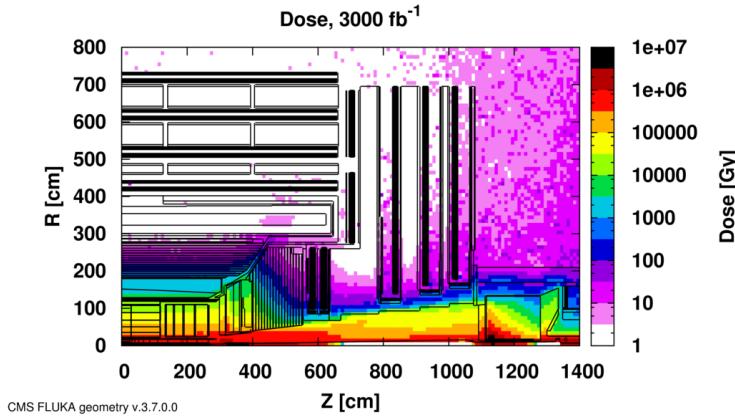


Figure 2.1: Absorbed dose in the CMS cavern after an integrated luminosity of 3000 fb^{-1} . R is the transverse distance from the beamline and Z is the distance along the beamline from the Interaction Point at $Z=0$.

The measurement of small production cross-section and/or decay branching ratio processes, such as the Higgs boson coupling to charge leptons or the $B_s \rightarrow \mu^+ \mu^-$ decay, is of major interest and specific upgrades in the forward regions of the detector will be required to maximize the physics acceptance on the largest possible solid angle. To ensure proper trigger performance within the present coverage, the muon system will be completed with new chambers. In figure 2.2 one can see that the existing Cathode Strip Chambers (CSCs) will be completed by Gas Electron Multipliers (GEMs) and Resistive Plate Chambers (RPCs) in the pseudo-rapidity region $1.6 < |\eta| < 2.4$ to complete its redundancy as originally scheduled in the CMS Technical Proposal [3].

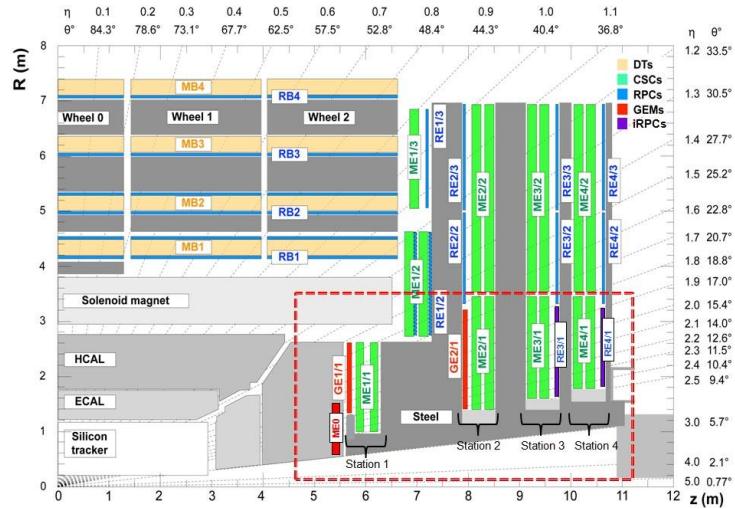


Figure 2.2: A quadrant of the muon system, showing DTs (yellow), RPCs (light blue), and CSCs (green). The locations of new forward muon detectors for Phase-II are contained within the dashed box and indicated in red for GEM stations ($ME0$, $GE1/1$, and $GE2/1$) and dark blue for improved RPC (iRPC) stations ($RE3/1$ and $RE4/1$).

RPCs are used by the CMS first level trigger for their good timing performances. Indeed, a very

438 good bunch crossing identification can be obtained with the present CMS RPC system, given their
 439 fast response of the order of 1 ns. In order to contribute to the precision of muon momentum mea-
 440 surements, muon chambers should have a spatial resolution less or comparable to the contribution
 441 of multiple scattering [1]. Most of the plausible physics is covered only considering muons with
 442 $p_T < 100$ GeV thus, in order to match CMS requirements, a spatial resolution of $\mathcal{O}(\text{few mm})$ the
 443 proposed new RPC stations, as shown by the simulation in figure 2.3. According to preliminary
 444 designs, RE3/1 and RE4/1 readout pitch will be comprised between 3 and 6 mm and 5 η -partitions
 445 could be considered.

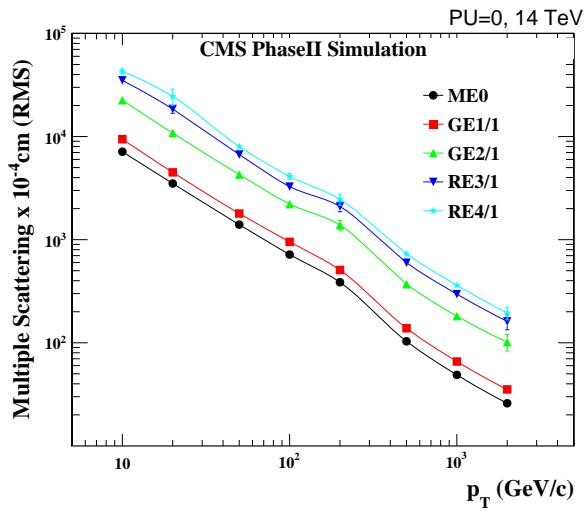


Figure 2.3: RMS of the multiple scattering displacement as a function of muon p_T for the proposed forward muon stations. All of the electromagnetic processes such as bremsstrahlung and magnetic field effect are included in the simulation.

3

446

447 Amplification processes in gaseous detectors

448 **3.1 Signal formation**

449 **3.2 Gas transport parameters**

4

450

451

Resistive Plate Chambers

452 **4.1 Principle**

453 **4.2 Rate capability of Resistive Plate Chambers**

454 **4.3 High time resolution**

455 **4.4 Resistive Plate Chambers at CMS**

456 **4.4.1 Overview**

457 The Resistive Plate Chambers (RPC) system, located in both barrel and endcap regions, provides a
458 fast, independent muon trigger with a looser p_T threshold over a large portion of the pseudorapidity
459 range ($|\eta| < 1.6$) [add reconstruction].

460

461 During High-Luminosity LHC (HL-LHC) operations the expected conditions in terms of back-
462 ground and pile-up will make the identification and correct P_T assignment a challenge for the Muon
463 system. The goal of RPC upgrade is to provide additional hits to the Muon system with precise tim-
464 ing. All these informations will be elaborated by the trigger system in a global way enhancing the
465 performance of the trigger in terms of efficiency and rate control. The RPC Upgrade is based on two
466 projects: an improved Link Board System and the extension of the RPC coverage up to $|\eta| = 2.4$.
467 [FIXME 2.4 or 2.5?]

468 The Link Board system, that will be described in section xxx, is responsible to process, syn-
469 chronize and zero-suppress the signals coming from the RPC front end boards. The Link Board
470 components have been produced between 2006 and 2007 and will be subjected to aging and failure
471 in the long term. The upgraded Link Board system will overcome the aging problems described in
472 section xxx and will allow for a more precise timing information to the RPC hits from 25 to 1 ns [ref
473 section xxx].

474 The extension of the RPC system up to $|\eta| = 2.1$ was already planned in the CMS TDR [ref
 475 cmstdr] and staged because of budget limitations and expected background rates higher than the rate
 476 capability of the present CMS RPCs in that region. An extensive R&D program has been done in
 477 order to develop an improved RPC that fulfills the CMS requirements. Two new RPC layers in the
 478 innermost ring of stations 3 and 4 will be added with benefits to the neutron-induced background
 479 reduction and efficiency improvement for both trigger and offline reconstruction.

480 4.4.2 The present RPC system

481 The RPC system is organized in 4 stations called RB1 to RB4 in the barrel region, and RE1 to RE4
 482 in the endcap region. The innermost barrel stations, RB1 and RB2, are instrumented with 2 layers
 483 of RPCs facing the innermost (RB1in and RB2in) and outermost (RB1out and RB2out) sides of the
 484 DT chambers. Every chamber is then divided from the read-out point of view into 2 or 3 η partitions
 485 called “rolls”. The RPC system consist of 480 barrel chambers and 576 endcap chambers. Details
 486 on the geometry are discussed in the paper [ref to geo paper].

487 The CMS RPC chamber is a double-gap, operated in avalanche mode to ensure reliable operation
 488 at high rates. Each RPC gap consists of two 2-mm-thick resistive High-Pressure Laminate (HPL)
 489 plates separated by a 2-mm-thick gas gap. The outer surface of the HPL plates is coated with a thin
 490 conductive graphite layer, and a voltage is applied. The RPCs are operated with a 3-component,
 491 non-flammable gas mixture consisting of 95.2% freon ($C_2H_2F_4$, known as R134a), 4.5% isobutane
 492 ($i-C_4H_{10}$), and 0.3% sulphur hexafluoride (SF_6) with a relative humidity of 40% - 50%. Readout
 493 strips are aligned in η between the 2 gas gaps. [\[Add a sentence on FEBs.\]](#)

494 The discriminated signals coming from the Front End boards feed via twisted cables (10 to 20 m
 495 long) the Link Board System located in UXC on the balconies around the detector. The Link System
 496 consist of the 1376 Link Boards (LBs) and the 216 Control Boards (CBs), placed in 108 Link Boxes.
 497 The Link Box is a custom crate (6U high) with 20 slots (for two CBs and eighteen LBs). The Link
 498 Box contains custom backplane to which the cables from the chambers are connected, as well as the
 499 cables providing the LBs and CBs power supply and the cables for the RPC FEBs control with use
 500 of the I2C protocol (trough the CB). The backplane itself contains only connectors (and no any other
 501 electronic devices).

502 The Link Board has 96 input channels (one channel corresponds to one RPC strip). The input
 503 signals are the ~ 100 ns binary pulses which are synchronous to the RPC hits, but not to the LHC
 504 clock (which drives the entire CMS electronics). Thus the first step of the FEB signals processing
 505 is synchronization, i.e. assignment of the signals to the BXes (25 ns periods). Then the data are
 506 compressed with a simple zero-suppressing algorithm (the input channels are grouped into 8 bit
 507 partitions, only the partitions with at least one nonzero bit are selected for each BX). Next, the non-
 508 empty partitions are time-multiplexed i.e. if there are more than one such partition in a given BX,
 509 they are sent one-by-one in consecutive BXes. The data from 3 neighbouring LBs are concentrated
 510 by the middle LB which contains the optical transmitter for sending them to the USC over a fiber at
 511 1.6 Gbps.

512 The Control Boards provide the communication of the control software with the LBs via the
 513 FEC/CCU system. The CBs are connected into token rings, each ring consists of 12 CBs of one
 514 detector tower and a FEC mezzanine board placed on the CCS board located in the VME crate in
 515 the USC. In total, there are 18 rings in the entire Link System. The CBs also perform automatic
 516 reloading of the LB's firmware which is needed in order to avoid accumulation of the radiation
 517 induced SEUs in the LBs firmware.

518 Both LBs and CB are based on the Xilinx Spartan III FPGAs, the CB additionally contains
 519 radiation-tolerant (FLASH based) FPGA Actel ProAsicPlus.

520 The High Voltage power system is located in USC, not exposed to radiation and easily accessible
 521 for any reparation. A single HV channel powers 2 RPC chambers both in the barrel and endcap
 522 regions. The Low Voltage boards are located in UXC on the balconies and provide the voltage to the
 523 front end electronics.

524 4.4.3 Pulse processing of CMS RPCs

525 Signals induced by cosmic particle in the RPC strips are shaped by standard CMS RPC Front-End
 526 Electronics (FEE) following the scheme of Figure 4.1. On a first stage, analogic signals are amplified
 527 and then sent to the Constant Fraction Discriminator (CFD) described in Figure 4.2. At the end of
 528 the chain, 100 ns long pulses are sent in the LVDS output. These output signal are sent on one side to
 529 a V1190A Time-to-Digital Converter (TDC) module from CAEN and on the other to an OR module
 530 to count the number of detected signals. Trigger and hit coïncidences are monitored using scalers.
 531 The TDC is used to store the data into ROOT files. These files are thus analysed to understand the
 532 detectors performance.

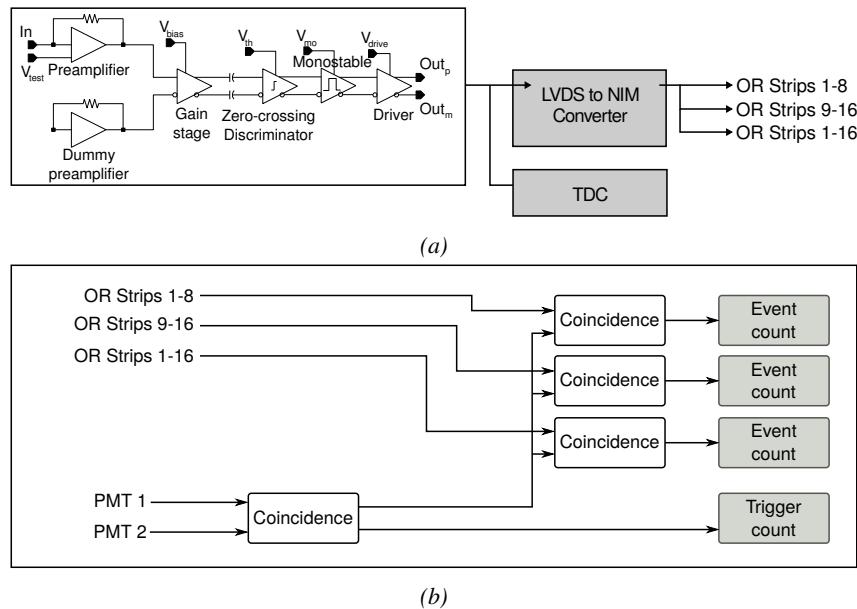


Figure 4.1: Signals from the RPC strips are shaped by the FEE described on Figure 4.1a. Output LVDS signals are then read-out by a TDC module connected to a computer or converted into NIM and sent to scalers. Figure 4.1b describes how these converted signals are put in coincidence with the trigger.

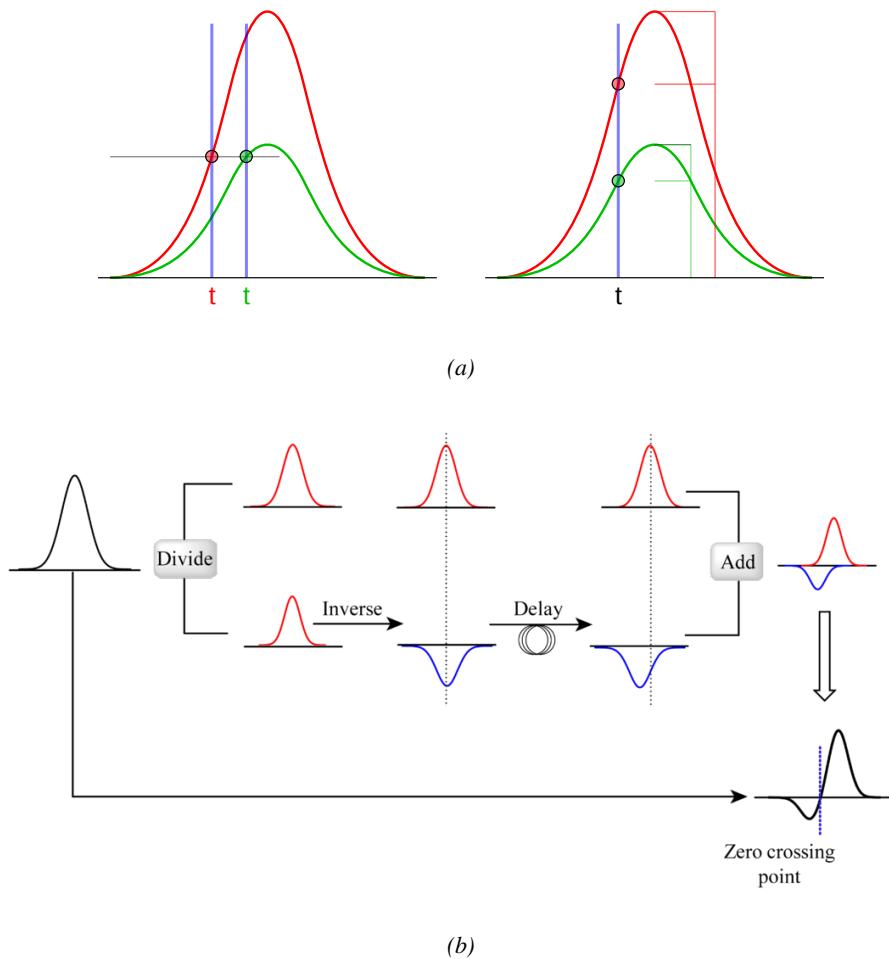


Figure 4.2: Description of the principle of a CFD. A comparison of threshold triggering (left) and constant fraction triggering (right) is shown in Figure 4.2a. Constant fraction triggering is obtained thanks to zero-crossing technique as explained in Figure 4.2b. The signal arriving at the input of the CFD is split into three components. A first one is delayed and connected to the inverting input of a first comparator. A second component is connected to the noninverting input of this first comparator. A third component is connected to the noninverting input of another comparator along with a threshold value connected to the inverting input. Finally, the output of both comparators is fed through an AND gate.

5

533

534
535

Longevity studies and Consolidation of the present CMS RPC subsystem

536

5.1 Testing detectors under extreme conditions

537
538
539
540
541
542
543

The upgrade from LHC to HL-LHC will increase the peak luminosity from $10^{34} \text{ cm}^{-2} \text{ s}^{-1}$ to reach $5 \times 10^{34} \text{ cm}^{-2} \text{ s}^{-1}$, increasing in the same way the total expected background to which the RPC system will be subjected to. Composed of low energy gammas and neutrons from $p\text{-}p$ collisions, low momentum primary and secondary muons, puch-through hadrons from calorimeters, and particles produced in the interaction of the beams with collimators, the background will mostly affect the regions of CMS that are the closest to the beam line, i.e. the RPC detectors located in the endcaps.

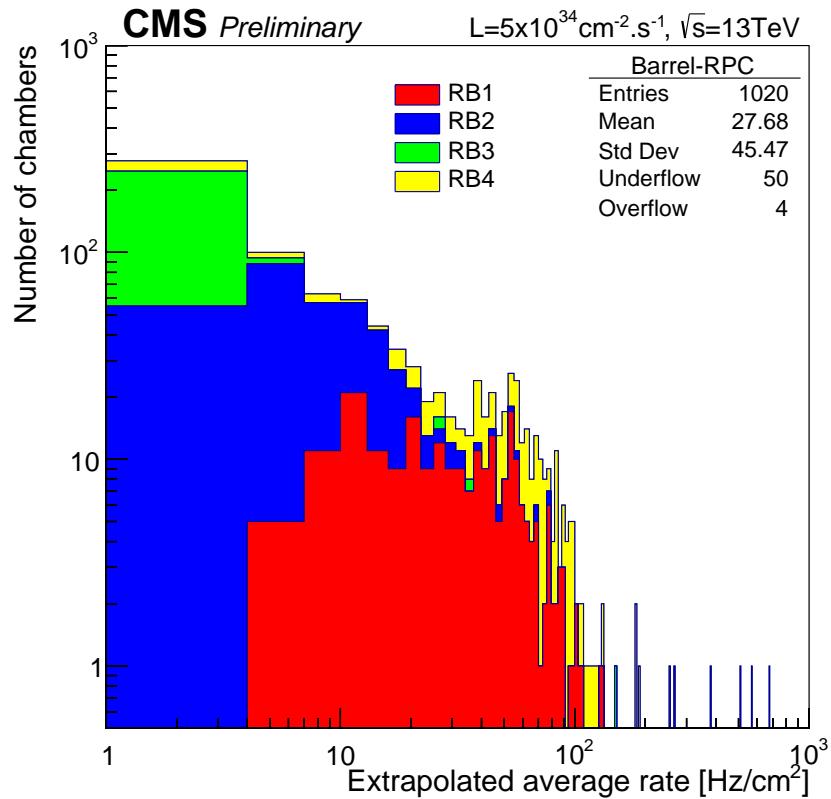
[To update.]

544

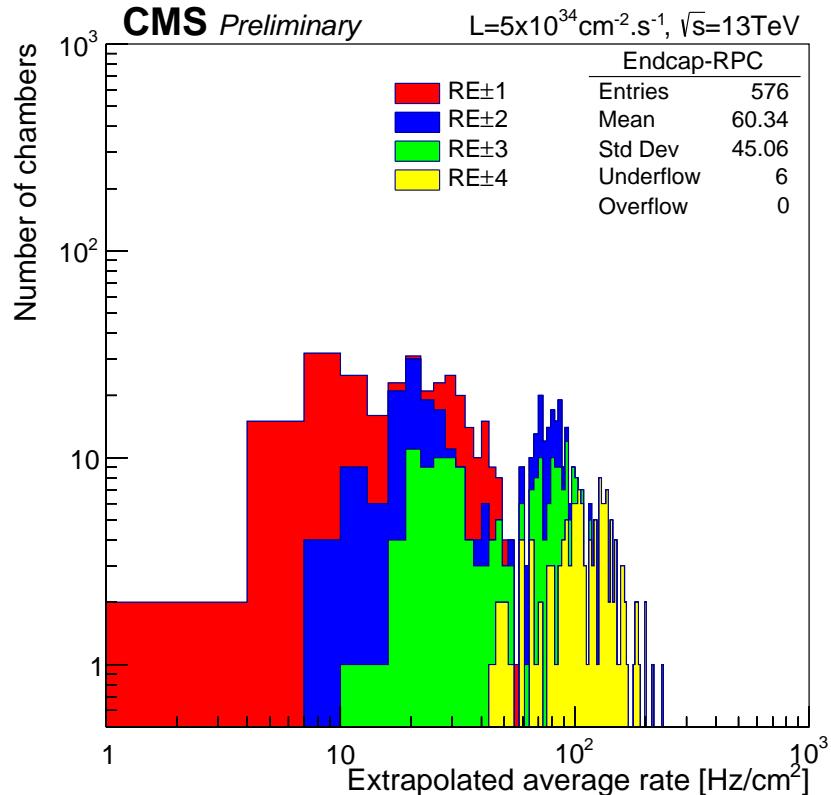
545
546
547
548
549
550
551
552

The 2016 data allowed to study the values of the background rate in all RPC system. In Figure 5.1, the distribution of the chamber background hit rate per unit area is shown at a luminosity of $5 \times 10^{34} \text{ cm}^{-2} \cdot \text{s}^{-1}$ linearly extrapolating from data collected in 2016 [ref mentioning the linear dependency of rate vs lumi]. The maximum rate per unit area at HL-LHC conditions is expected to be of the order of 600 Hz/cm^2 (including a safety factor 3). Nevertheless, Fluka simulations have conducted in order to understand the background at HL-LHC conditions. The comparison to the data has shown, in Figure 5.2, a discrepancy of a factor 2 even though the order of magnitude is consistent. [Understand mismatch.]

553



(a)



(b)

Figure 5.1: (5.1a) Extrapolation from 2016 data of single hit rate per unit area in the barrel region. (5.1b) Extrapolation from 2016 data of single hit rate per unit area in the endcap region.

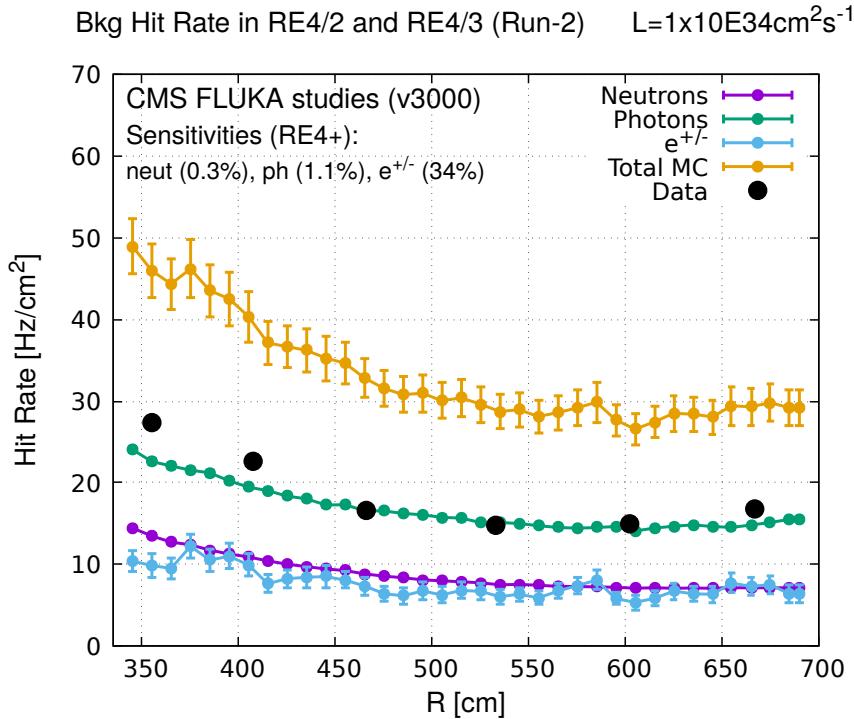


Figure 5.2: Background Fluka simulation compared to 2016 Data at $L = 10^{34} \text{ cm}^{-2} \cdot \text{s}^{-1}$ in the fourth endcap disk region. A mismatch in between simulation and data can be observed. [\[To be understood.\]](#)

554 In the past, extensive long-term tests were carried out at several gamma and neutron facilities
 555 certifying the detector performance. Both full size and small prototype RPCs have been irradiated
 556 with photons up to an integrated charge of $\sim 0.05 \text{ C/cm}^2$ and $\sim 0.4 \text{ C/cm}^2$, respectively [4, 5].
 557 During Run-I, the RPC system provided stable operation and excellent performance and did not
 558 show any aging effects for integrated charge of the order of 0.01 C/cm^2 . Projections on currents
 559 from 2016 Data, has allowed to determine that the total integrated charge, by the end of HL-LHC,
 560 would be of the order of 1 C/cm^2 (including a safety factor 3). [\[Corresponding figure needed.\]](#)
 561

562 5.1.1 The Gamma Irradiation Facilities

563 5.1.1.1 GIF

564 Located in the SPS West Area at the downstream end of the X5 test beam, the Gamma Irradiation
 565 Facility (GIF) was a test area in which particle detectors were exposed to a particle beam in presence
 566 of an adjustable gamma background [6]. Its goal was to reproduce background conditions these
 567 detectors would suffer in their operating environment at LHC. GIF layout is shown in Figure 5.3.
 568 Gamma photons are produced by a strong ^{137}Cs source installed in the upstream part of the zone
 569 inside a lead container. The source container includes a collimator, designed to irradiate a $6 \times 6 \text{ m}^2$
 570 area at 5 m maximum to the source. A thin lens-shaped lead filter helps providing with a uniform
 571 outcoming flux in a vertical plane, orthogonal to the beam direction. The principal collimator hole
 572 provides a pyramidal aperture of $74^\circ \times 74^\circ$ solid angle and provides a photon flux in a pyramidal vol-

ume along the beam axis. The photon rate is controled by further lead filters allowing the maximum rate to be limited and to vary within a range of four orders of magnitude. Particle detectors under test are then placed within the pyramidal volume in front of the source, perpendicularly to the beam line in order to profit from the homogeneous photon flux. Adjusting the background flux of photons can then be done by using the filters and choosing the position of the detectors with respect to the source.

578

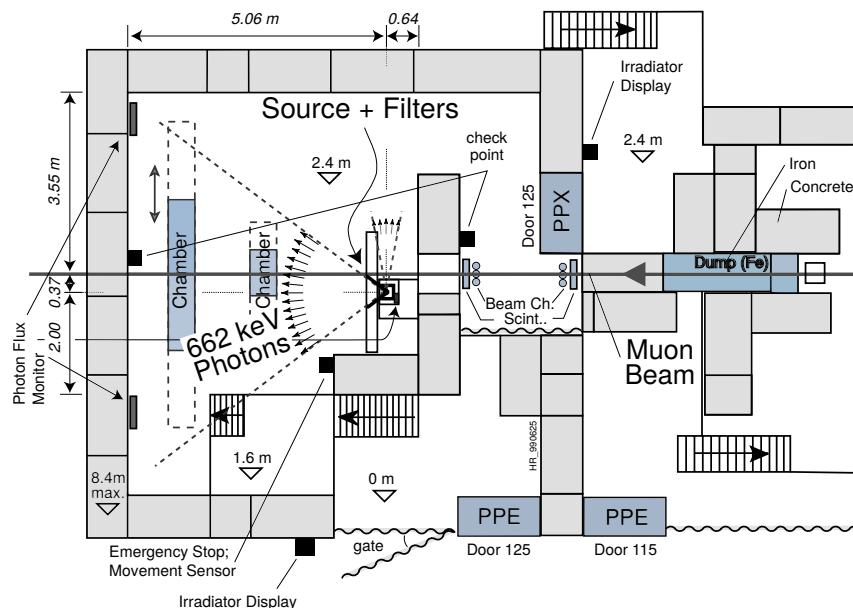


Figure 5.3: Layout of the test beam zone called X5c GIF at CERN. Photons from the radioactive source produce a sustained high rate of random hits over the whole area. The zone is surrounded by 8 m high and 80 cm thick concrete walls. Access is possible through three entry points. Two access doors for personnel and one large gate for material. A crane allows installation of heavy equipment in the area.

579 As described on Figure 5.4, the ^{137}Cs source emits a 662 keV photon in 85% of the decays. An
 580 activity of 740 GBq was measured on the 5th March 1997. To estimate the strength of the flux in
 581 2014, it is necessary to consider the nuclear decay through time assiciated to the Cesium source
 582 whose half-life is well known ($t_{1/2} = (30.05 \pm 0.08)$ y). The GIF tests where done in between the
 583 20th and the 31st of August 2014, i.e. at a time $t = (17.47 \pm 0.02)$ y resulting in an attenuation of
 584 the activity from 740 GBq in 1997 to 494 GBq in 2014.

585

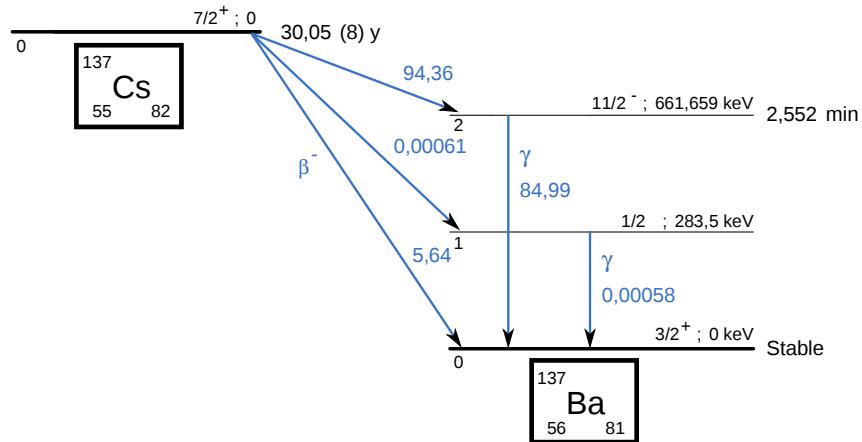


Figure 5.4: ^{137}Cs decays by β^- emission to the ground state of ^{137}Ba ($BR = 5.64\%$) and via the 662 keV isomeric level of ^{137}Ba ($BR = 94.36\%$) whose half-life is 2.55 min.

5.1.1.2 GIF++

The new Gamma Irradiation Facility (GIF++), located in the SPS North Area at the downstream end of the H4 test beam, has replaced its predecessor during LS1 and has been operational since spring 2015 [7]. Like GIF, GIF++ features a ^{137}Cs source of 662 keV gamma photons, their fluence being controlled with a set of filters of various attenuation factors. The source provides two separated large irradiation areas for testing several full-size muon detectors with continuous homogeneous irradiation, as presented in Figure 5.5.

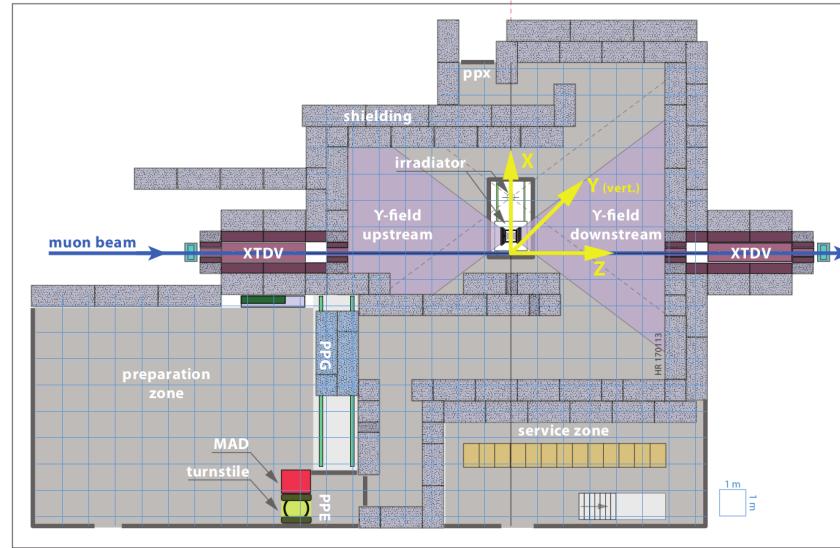


Figure 5.5: Floor plan of the GIF++ facility. When the facility downstream of the GIF++ takes electron beam, a beam pipe is installed along the beam line (z -axis). The irradiator can be displaced laterally (its center moves from $x = 0.65$ m to 2.15 m), to increase the distance to the beam pipe.

594 The source activity was measured to be about 13.5 TBq in March 2016. The photon flux being
 595 far greater than HL-LHC expectations, GIF++ provides an excellent facility for accelerated aging
 596 tests of muon detectors.

597

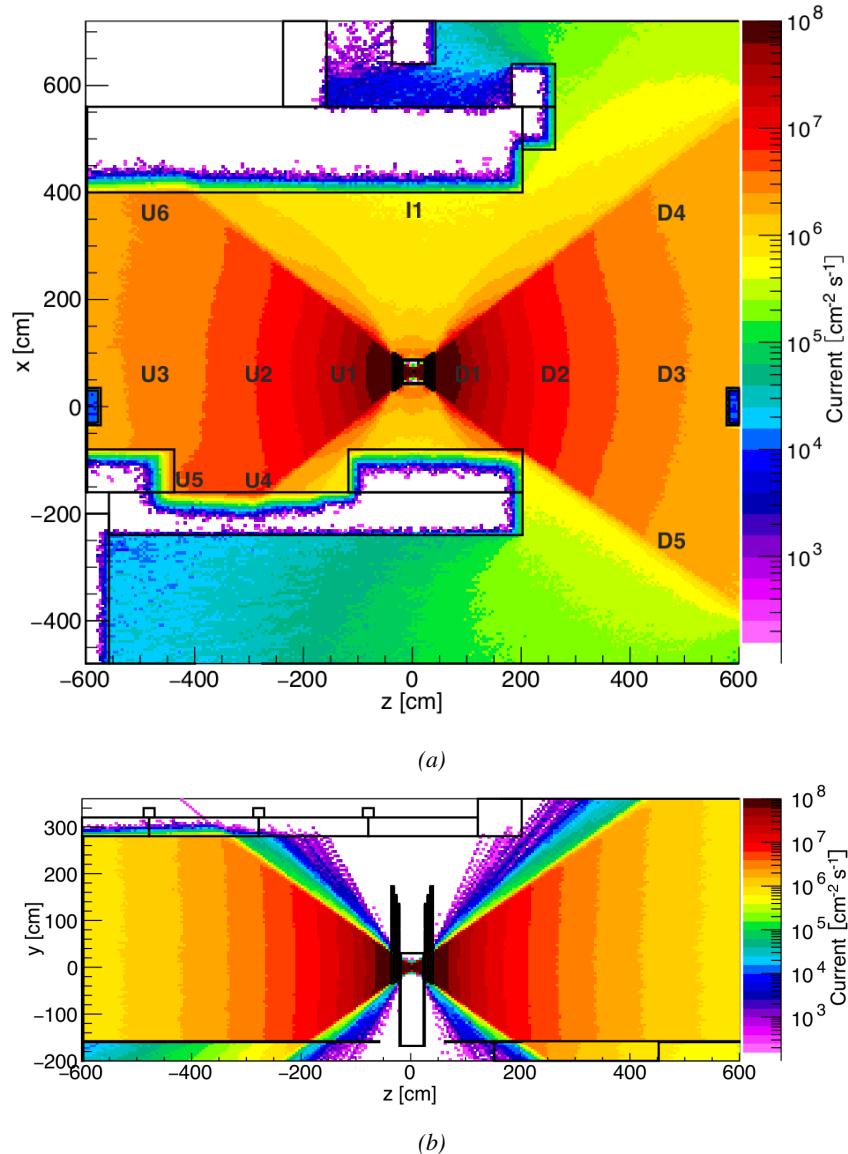


Figure 5.6: Simulated unattenuated current of photons in the xz plane (Figure 5.6a) and yz plane (Figure 5.6b) through the source at $x = 0.65 \text{ m}$ and $y = 0 \text{ m}$. With angular correction filters, the current of 662 keV photons is made uniform in xy planes.

598 The source is situated in the muon beam line with the muon beam being available a few times a
 599 year. The H4 beam, composed of muons with a momentum of about $150 \text{ GeV}/c$, passes through the
 600 GIF++ zone and is used to study the performance of the detectors. Its flux is of $104 \text{ particles}/\text{s cm}^2$

601 focused in an area similar to $10 \times 10 \text{ cm}^2$. Therefore, with properly adjusted filters, one can imitate
 602 the HL-LHC background and study the performance of muon detectors with their trigger/readout
 603 electronics in HL-LHC environment.

604

605 5.2 Preliminary tests at GIF

606 5.2.1 Resistive Plate Chamber test setup

607 During summer 2014, preliminary tests have been conducted in the GIF area on a newly produced
 608 RE4/2 chamber labelled RE-4-2-BARC-161. This chamber has been placed into a trolley covered
 609 with a tent. The position of the RPC inside the tent and of the tent related to the source is described
 610 in Figure 5.7. To test this CMS RPC, three different absorber settings were used. First of all,
 611 measurements were done with fully opened source. Then, to complete this preliminary study, the
 612 gamma flux has been attenuated from a factor 2 and a factor 5. The expected gamma flux at the level
 613 of our detector will be discussed in subsection ??.

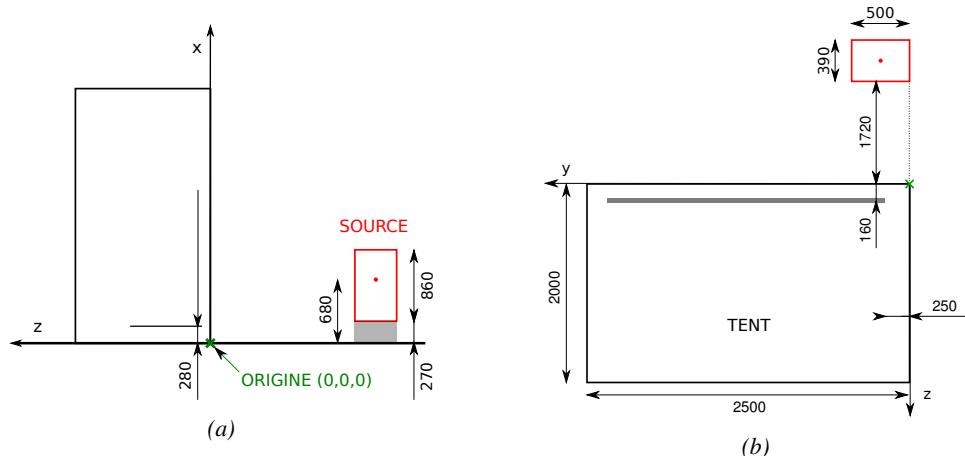


Figure 5.7: Description of the RPC setup. Dimensions are given in mm. A tent containing RPCs is placed at 1720 mm from the source container. The source is situated in the center of the container. RE-4-2-BARC-161 chamber is 160 mm inside the tent. This way, the distance between the source and the chambers plan is 2060 mm. Figure 5.7a provides a side view of the setup in the xz plane while Figure 5.7b shows a top view in the yz plane.



Figure 5.8: RE-4-2-BARC-161 chamber is inside the tent as described in Figure 5.7. In the top right, the two scintillators used as trigger can be seen. This trigger system has an inclination of 10° relative to horizontal and is placed above half-partition B2 of the RPCs. PMT electronics are shielded thanks to lead blocks placed in order to protect them without stopping photons from going through the scintillators and the chamber.

614 At the time of the tests, the beam not being operational anymore, a trigger composed of 2
615 plastic scintillators has been placed in front of the setup with an inclination of 10 deg with respect to
616 the detector plane in order to look at cosmic muons. Using this particular trigger layout, shown on
617 Figure 5.8, leads to a cosmic muon hit distribution into the chamber similar to the one in Figure 5.9.
618 Measured without gamma irradiation, two peaks can be seen on the profil of partition B, centered
619 on strips 52 and 59. Section ?? will help us understand that these two peaks are due respectively to
620 forward and backward coming cosmic particles where forward coming particles are first detected by
621 the scintillators and then the RPC while the backward coming muons are first detected in the RPC.

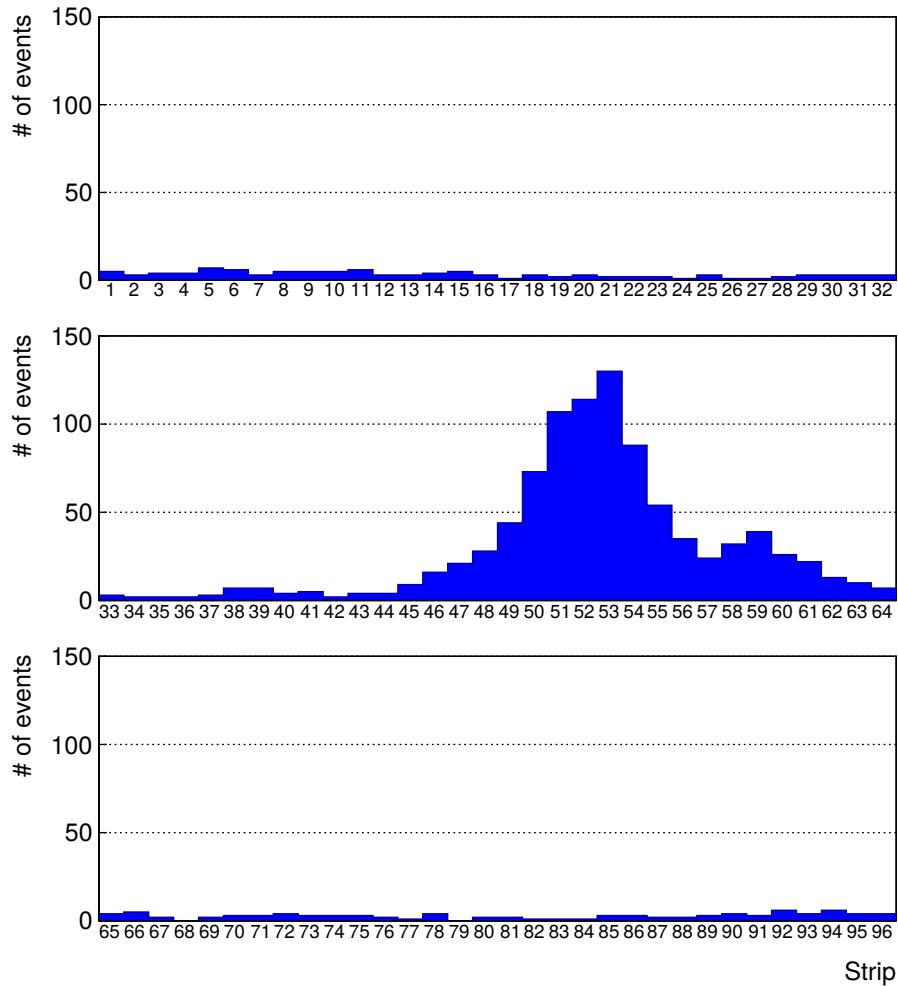


Figure 5.9: Hit distributions over all 3 partitions of RE-4-2-BARC-161 chamber is showed on these plots. Top, middle and bottom figures respectively correspond to partitions A, B, and C. These plots show that some events still occur in other half-partitions than B2, which corresponds to strips 49 to 64, in front of which the trigger is placed, contributing to the inefficiency of detection of cosmic muons. In the case of partitions A and C, the very low amount of data can be interpreted as noise. On the other hand, it is clear that a little portion of muons reach the half-partition B1, corresponding to strips 33 to 48.

5.2.2 Data Acquisition

5.2.3 Geometrical acceptance of the setup layout to cosmic muons

In order to profit from a constant gamma irradiation, the detectors inside of the GIF bunker need to be placed in a plane orthogonal to the beam line. The muon beam that used to be available was meant to test the performance of detectors under test. This beam not being active anymore, another solution to test detector performance had to be used. Thus, it has been decided to use cosmic muons detected through a telescope composed of two scintillators. Lead blocks were used as shielding to

629 protect the photomultipliers from gammas as can be seen from Figure 5.8.

630 An inclination has been given to the cosmic telescope to maximize the muon flux. A good com-
 631 promise had to be found between good enough muon flux and narrow enough hit distribution to
 632 be sure to contain all the events into only one half partitions as required from the limited available
 633 readout hardware. Nevertheless, a consequence of the misplaced trigger, that can be seen as a loss
 634 of events in half-partition B1 in Figure 5.9, is an inefficiency. Nevertheless, the inefficiency of ap-
 635 proximately 20 % highlighted in Figure 5.10 by comparing the performance of chamber BARC-161
 636 in 904 and at GIF without irradiation seems too important to be explained only by the geometri-
 637 cal acceptance of the setup itself. Simulations have been conducted to show how the setup brings
 638 inefficiency.

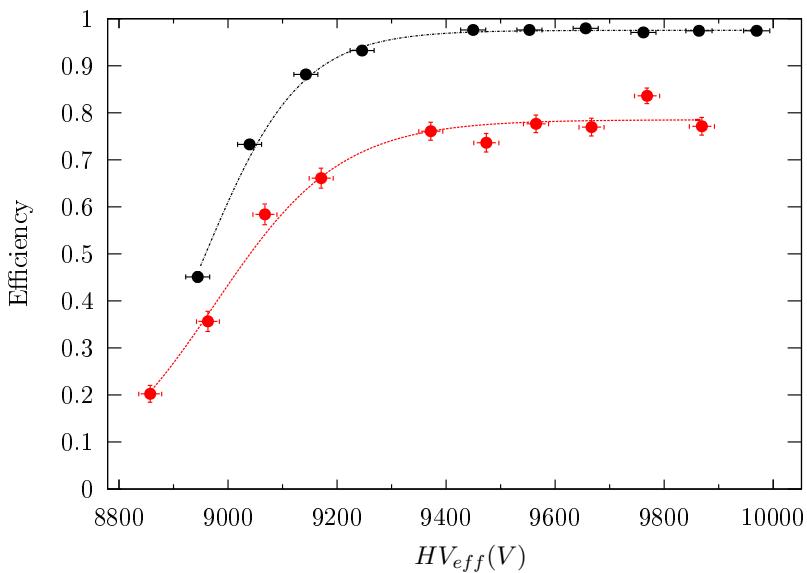


Figure 5.10: Results are derived from data taken on half-partition B2 only. On the 18th of June 2014, data has been taken on chamber RE-2-BARC-161 at building 904 (Prevessin Site) with cosmic muons providing us a reference efficiency plateau of $(97.54 \pm 0.15)\%$ represented by a black curve. A similar measurement has been done at GIF on the 21st of July with the same chamber giving a plateau of $(78.52 \pm 0.94)\%$ represented by a red curve.

639 5.2.3.1 Description of the simulation layout

640 The layout of GIF setup has been reproduced and incorporated into a Monte Carlo (MC) simulation
 641 to study the influence of the disposition of the telescope on the final distribution measured by the
 642 RPC. A 3D view of the simulated layout is given into Figure 5.11. Muons are generated randomly
 643 in a horizontal plane located at a height corresponding to the lowest point of the PMTs. This way,
 644 the needed size of the plane in order to simulate events happening at very big azimuthal angles (i.e.
 645 $\theta \approx \pi$) can be kept relatively small. The muon flux is designed to follow the usual $\cos^2\theta$ distribution
 646 for cosmic particle. The goal of the simulation is to look at muons that pass through the muon
 647 telescope composed of the two scintillators and define their distribution onto the RPC plane. During
 648 the reconstruction, the RPC plane is then divided into its strips and each muon track is assigned to a
 649 strip.

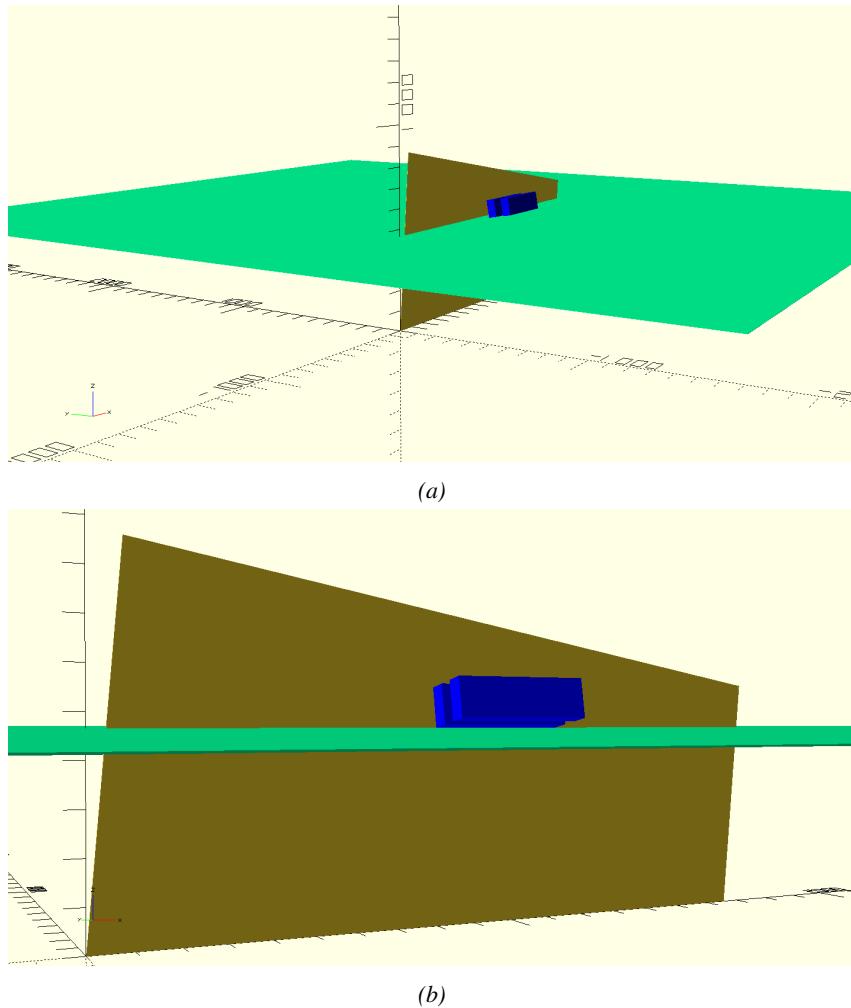


Figure 5.11: Representation of the layout used for the simulations of the test setup. The RPC is represented as a yellow trapezoid while the two scintillators as blue cuboids looking at the sky. A green plane corresponds to the muon generation plane within the simulation. Figure 5.7a shows a global view of the simulated setup. Figure 5.7b shows a zoomed view that allows to see the 2 scintillators as well as the full RPC plane.

650 In order to further refine the quality of the simulation and understand deeper the results the
 651 dependance of the distribution has been studied for a range of telescope inclinations. Moreover,
 652 the threshold applied on the PMT signals has been included into the simulation in the form of a
 653 cut. In the approximation of uniform scintillators, it has been considered that the threshold can be
 654 understood as the minimum distance particles need to travel through the scintillating material to give
 655 a strong enough signal. Particles that travel a distance smaller than the set "threshold" are thus not
 656 detected by the telescope and cannot trigger the data taking. Finally, the FEE threshold also has
 657 been considered in a similar way. The mean momentum of horizontal cosmic rays is higher than
 658 those of vertical ones but the stopping power of matter for momenta ranging from 1 GeV to 1 TeV
 659 stays comparable. It is then possible to assume that the mean number of primary e^-/ion pairs per
 660 unit length will stay similar and thus, depending on the applied discriminator threshold, muons with

661 the shortest path through the gas volume will deposit less charge and induce a smaller signal on the
 662 pick-up strips that could eventually not be detected. These two thresholds also restrain the overall
 663 geometrical acceptance of the system.

664 **5.2.3.2 Simulation procedure**

665 The simulation software has been designed using C++ and the output data is saved into ROOT
 666 histograms. Simulations start for a threshold T_{scint} varying in a range from 0 to 45 mm in steps
 667 of 5 mm, where $T_{scint} = 0$ mm corresponds to the case where there isn't any threshold apply on
 668 the input signal while $T_{scint} = 45$ mm, which is the scintillator thickness, is the case where muons
 669 cannot arrive orthogonally onto the scintillator surface. For a given T_{scint} , a set of RPC thresholds
 670 are considered. The RPC threshold, T_{RPC} varies from 2 mm, the thickness of the gas volume, to
 671 3 mm in steps of 0.25 mm. For each $(T_{scint}; T_{RPC})$ pair, $N_\mu = 10^8$ muons are randomly generated
 672 inside the muon plane described in the previous paragraph with an azimuthal angle θ chosen to follow
 673 a $\cos^2\theta$ distribution.

674 Planes are associated to each surface of the scintillators. Knowing muon position into the muon
 675 plane and its direction allows us, by assuming that muons travel in a straight line, to compute the
 676 intersection of the muon track with these planes. Applying conditions to the limits of the surfaces
 677 of the scintillator faces then gives us an answer to whether or not the muon passed through the
 678 scintillators. In the case the muon has indeed passed through the telescope, the path through each
 679 scintillator is computed and muons whose path was shorter than T_{scint} are rejected and are thus
 680 considered as having not interacted with the setup.

681 On the contrary, if the muon is labeled as good, its position within the RPC plane is computed
 682 and the corresponding strip, determined by geometrical tests in the case the distance through the
 683 gas volume was enough not to be rejected because of T_{RPC} , gets a hit and several histograms
 684 are filled in order to keep track of the generation point on the muon plane, the intersection points
 685 of the reconstructed muons within the telescope, or on the RPC plane, the path traveled through
 686 each individual scintillator or the gas volume, as well as other histograms. Moreover, muons fill
 687 different histograms whether they are forward or backward coming muons. They are discriminated
 688 according to their direction components. When a muon is generated, an (x, y, z) position is assigned
 689 into the muon plane as well as a $(\theta; \phi)$ pair that gives us the direction it's coming from. This way,
 690 muons satisfying the condition $0 \leq \phi < \pi$ are designated as backward coming muons while muons
 691 satisfying $\pi \leq \phi < 2\pi$ as forward coming muons.

692 This simulation is then repeated for different telescope inclinations ranging in between 4 and 20°
 693 and varying in steps of 2° . Due to this inclination and to the vertical position of the detector under
 694 test, the muon distribution reconstructed in the detector plane is asymmetrical. The choice as been
 695 made to chose a skew distribution formula to fit the data built as the multiplication of gaussian and
 696 sigmoidal curves together. A typical gaussian formula is given as 5.1 and has three free parameters
 697 as A_g , its amplitude, \bar{x} , its mean value and σ , its root mean square. Sigmoidal curves as given by
 698 formula 5.2 are functions converging to 0 and A_s as x diverges. The inflection point is given as x_i
 699 and λ is proportional to the slope at $x = x_i$. In the limit where $\lambda \rightarrow \infty$, the sigmoid becomes a
 700 step function.

$$g(x) = A_g e^{-\frac{(x-\bar{x})^2}{2\sigma^2}} \quad (5.1)$$

$$s(x) = \frac{A_s}{1 + e^{-\lambda(x-x_i)}} \quad (5.2)$$

Finally, a possible representation of a skew distribution is given by formula 5.3 and is the product of 5.1 and 5.2. Naturally, here $A_{sk} = A_g \times A_s$ and represents the theoretical maximum in the limit where the skew tends to a gaussian function.

$$sk(x) = g(x) \times s(x) = A_{sk} \frac{e^{\frac{-(x-\bar{x})^2}{2\sigma^2}}}{1 + e^{-\lambda(x-x_i)}} \quad (5.3)$$

5.2.3.3 Results

Influence of T_{scint} on the muon distribution

Influence of T_{RPC} on the muon distribution

Influence of the telescope inclination on the muon distribution

Comparison to data taken at GIF without irradiation

5.2.4 Photon flux at GIF

5.2.4.1 Expectations from simulations

In order to understand and evaluate the γ flux in the GIF area, simulations had been conducted in 1999 and published by S. Agosteo et al [6]. Table 5.1 presented in this article gives us the γ flux for different distances D to the source. This simulation was done using GEANT and a Monte Carlo N-Particle (MCNP) transport code, and the flux F is given in number of γ per unit area and unit time along with the estimated error from these packages expressed in %.

Nominal ABS	Photon flux F [$s^{-1}cm^{-2}$]			
	at $D = 50$ cm	at $D = 155$ cm	at $D = 300$ cm	at $D = 400$ cm
1	$0.12 \cdot 10^8 \pm 0.2\%$	$0.14 \cdot 10^7 \pm 0.5\%$	$0.45 \cdot 10^6 \pm 0.5\%$	$0.28 \cdot 10^6 \pm 0.5\%$
2	$0.68 \cdot 10^7 \pm 0.3\%$	$0.80 \cdot 10^6 \pm 0.8\%$	$0.25 \cdot 10^6 \pm 0.8\%$	$0.16 \cdot 10^6 \pm 0.6\%$
5	$0.31 \cdot 10^7 \pm 0.4\%$	$0.36 \cdot 10^6 \pm 1.2\%$	$0.11 \cdot 10^6 \pm 1.2\%$	$0.70 \cdot 10^5 \pm 0.9\%$

Table 5.1: Total photon flux ($E\gamma \leq 662$ keV) with statistical error predicted considering a ^{137}Cs activity of 740 GBq at different values of the distance D to the source along the x -axis of irradiation field [6].

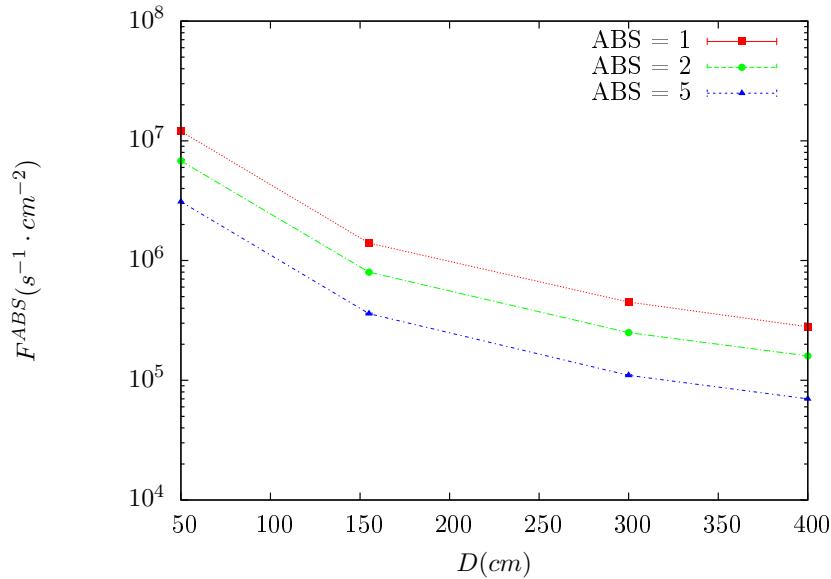


Figure 5.12: γ flux $F(D)$ is plot using values from table 5.1. As expected, the plot shows similar attenuation behaviours with increasing distance for each absorption factors.

The simulation doesn't directly provide us with an estimated flux at the level of our RPC. First of all, it is needed to extract the value of the flux from the available data contained in the original paper and then to estimate the flux in 2014 at the time the experimentation took place. Figure 5.12 that contains the data from Table 5.1. In the case of a pointlike source emitting isotropic and homogeneous gamma radiations, the gamma flux F at a distance D to the source with respect to a reference point situated at D_0 where a known flux F_0 is measured will be expressed like in Equation 5.4, assuming that the flux decreases as $1/D^2$, where c is a fitting factor.

$$F^{ABS} = F_0^{ABS} \times \left(\frac{cD_0}{D} \right)^2 \quad (5.4)$$

By rewriting Equation 5.4, it comes that :

$$c = \frac{D}{D_0} \sqrt{\frac{F^{ABS}}{F_0^{ABS}}} \quad (5.5)$$

$$\Delta c = \frac{c}{2} \left(\frac{\Delta F^{ABS}}{F^{ABS}} + \frac{\Delta F_0^{ABS}}{F_0^{ABS}} \right) \quad (5.6)$$

Finally, using Equation 5.5 and the data in Table 5.1 with $D_0 = 50$ cm as reference point, we can build Table 5.2. It is interesting to note that c for each value of D doesn't depend on the absorption factor.

Nominal ABS	Correction factor c		
	at $D = 155$ cm	at $D = 300$ cm	at $D = 400$ cm
1	$1.059 \pm 0.70\%$	$1.162 \pm 0.70\%$	$1.222 \pm 0.70\%$
2	$1.063 \pm 1.10\%$	$1.150 \pm 1.10\%$	$1.227 \pm 0.90\%$
5	$1.056 \pm 1.60\%$	$1.130 \pm 1.60\%$	$1.202 \pm 1.30\%$

Table 5.2: Correction factor c is computed thanks to formulae 5.5 taking as reference $D_0 = 50$ cm and the associated flux F_0^{ABS} for each absorption factor available in table 5.1.

727 For the range of D/D_0 values available, it is possible to use a simple linear fit to get the evolution
 728 of c . The linear fit will then use only 2 free parameters, a and b , as written in Equation 5.7. This gives
 729 us the results showed in Figure 5.13. Figure 5.13b confirms that using only a linear fit to extract c is
 730 enough as the evolution of the rate that can be obtained superimposes well on the simulation points.

$$c \left(\frac{D}{D_0} \right) = a \frac{D}{D_0} + b \quad (5.7)$$

$$F^{ABS} = F_0^{ABS} \left(a + \frac{bD_0}{D} \right)^2 \quad (5.8)$$

$$\Delta F^{ABS} = F^{ABS} \left[\frac{\Delta F_0^{ABS}}{F_0^{ABS}} + 2 \frac{\Delta a + \Delta b \frac{D_0}{D}}{a + \frac{bD_0}{D}} \right] \quad (5.9)$$

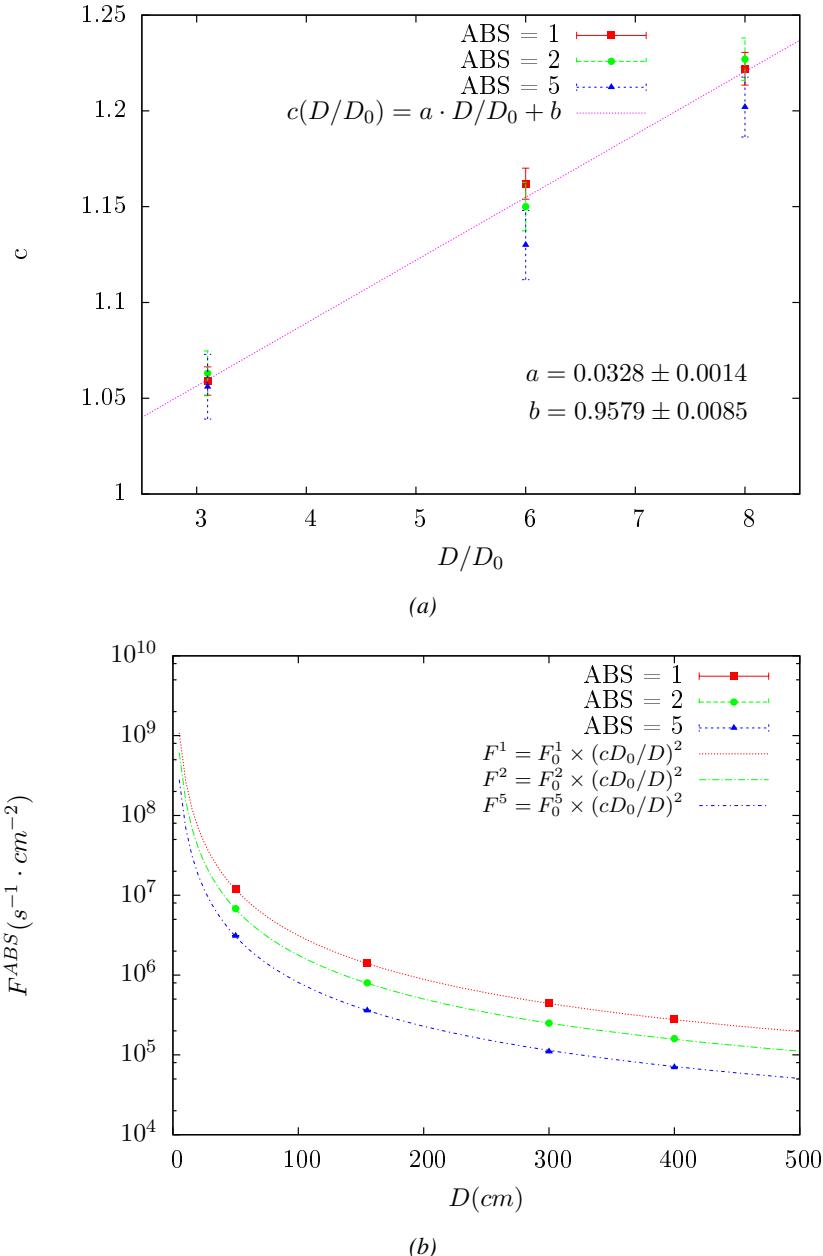


Figure 5.13: Figure 5.13a shows the linear approximation fit done via formulae 5.7 on data from table 5.2. Figure 5.13b shows a comparison of this model with the simulated flux using a and b given in figure 5.13a in formulae 5.4 and the reference value $D_0 = 50\text{cm}$ and the associated flux for each absorption factor F_0^{ABS} from table 5.1

In the case of the 2014 GIF tests, the RPC plane is located at a distance $D = 206\text{ cm}$ to the source.
 Moreover, to estimate the strength of the flux in 2014, it is necessary to consider the nuclear decay through time associated to the Cesium source whose half-life is well known ($t_{1/2} = (30.05 \pm 0.08)\text{ y}$).
 The very first source activity measurement has been done on the 5th of March 1997 while the GIF

⁷³¹ In the case of the 2014 GIF tests, the RPC plane is located at a distance $D = 206\text{ cm}$ to the source.

⁷³² Moreover, to estimate the strength of the flux in 2014, it is necessary to consider the nuclear decay

⁷³³ through time associated to the Cesium source whose half-life is well known ($t_{1/2} = (30.05 \pm 0.08)\text{ y}$).

⁷³⁴ The very first source activity measurement has been done on the 5th of March 1997 while the GIF

735 tests where done in between the 20th and the 31st of August 2014, i.e. at a time $t = (17.47 \pm 0.02)$ y
 736 resulting in an attenuation of the activity from 740 GBq in 1997 to 494 GBq in 2014. All the needed
 737 information to extrapolate the flux through our detector in 2014 has now been assembled, leading
 738 to the Table 5.3. It is interesting to note that for a common RPC sensitivity to γ of $2 \cdot 10^{-3}$, the
 739 order of magnitude of the estimated hit rate per unit area is of the order of the kHz for the fully
 740 opened source. Moreover, taking profit of the two working absorbers, it will be possible to scan
 741 background rates at 0 Hz, ~ 300 Hz as well as ~ 600 Hz. Without source, a good estimate of the
 742 intrinsic performance will be available. Then at 300 Hz, the goal will be to show that the detectors
 743 fulfill the performance certification of CMS RPCs. Then a first idea of the performance of the
 744 detectors at higher background will be provided with absorption factors 2 (~ 600 Hz) and 1 (no
 745 absorption). *[Here I will also put a reference to the plot showing the estimated background rate at
 746 the level of RE3/1 in the case of HL-LHC but this one being in another chapter, I will do it later.]*

Nominal ABS	Photon flux F [$s^{-1}cm^{-2}$]			Hit rate/unit area [$Hz cm^{-2}$] at $D^{2014} = 206$ cm
	at $D_0^{1997} = 50$ cm	at $D_0^{1997} = 206$ cm	at $D^{2014} = 206$ cm	
1	$0.12 \cdot 10^8 \pm 0.2\%$	$0.84 \cdot 10^6 \pm 0.3\%$	$0.56 \cdot 10^6 \pm 0.3\%$	1129 ± 32
2	$0.68 \cdot 10^7 \pm 0.3\%$	$0.48 \cdot 10^6 \pm 0.3\%$	$0.32 \cdot 10^6 \pm 0.3\%$	640 ± 19
5	$0.31 \cdot 10^7 \pm 0.4\%$	$0.22 \cdot 10^6 \pm 0.3\%$	$0.15 \cdot 10^6 \pm 0.3\%$	292 ± 9

Table 5.3: The data at D_0 in 1997 is taken from [6]. In a second step, using Equations 5.8 and 5.9, the flux at D can be estimated in 1997. Then, taking into account the attenuation of the source activity, the flux at D can be estimated at the time of the tests in GIF in 2014. Finally, assuming a sensitivity of the RPC to γ $s = 2 \cdot 10^{-3}$, an estimation of the hit rate per unit area is obtained.

⁷⁴⁷ 5.2.4.2 Dose measurements

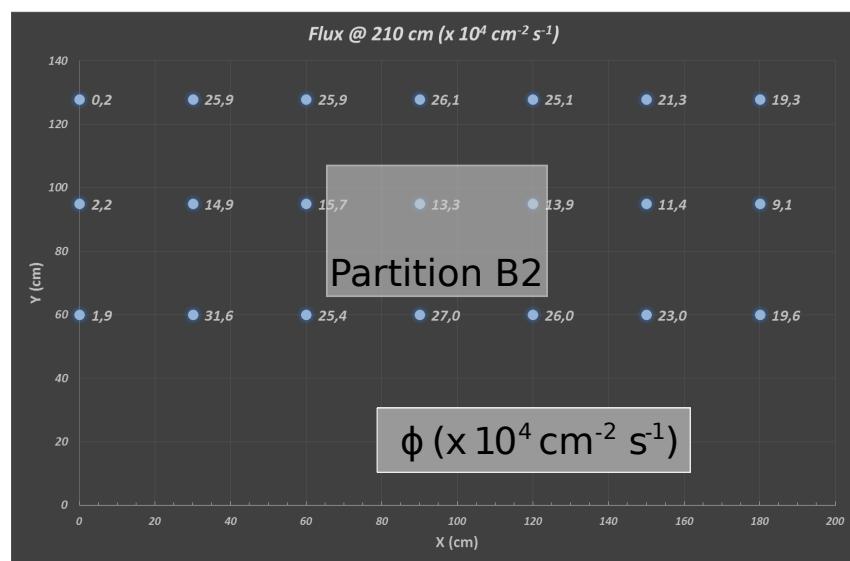


Figure 5.14: Dose measurements has been done in a plane corresponding to the tents front side. This plan is 1900 mm away from the source. As explained in the first chapter, a lens-shaped lead filter provides a uniform photon flux in the vertical plan orthogonal to the beam direction. If the second line of measured fluxes is not taken into account because of lower values due to experimental equipments in the way between the source and the tent, the uniformity of the flux is well showed by the results.

⁷⁴⁸ **5.2.5 Results and discussions**

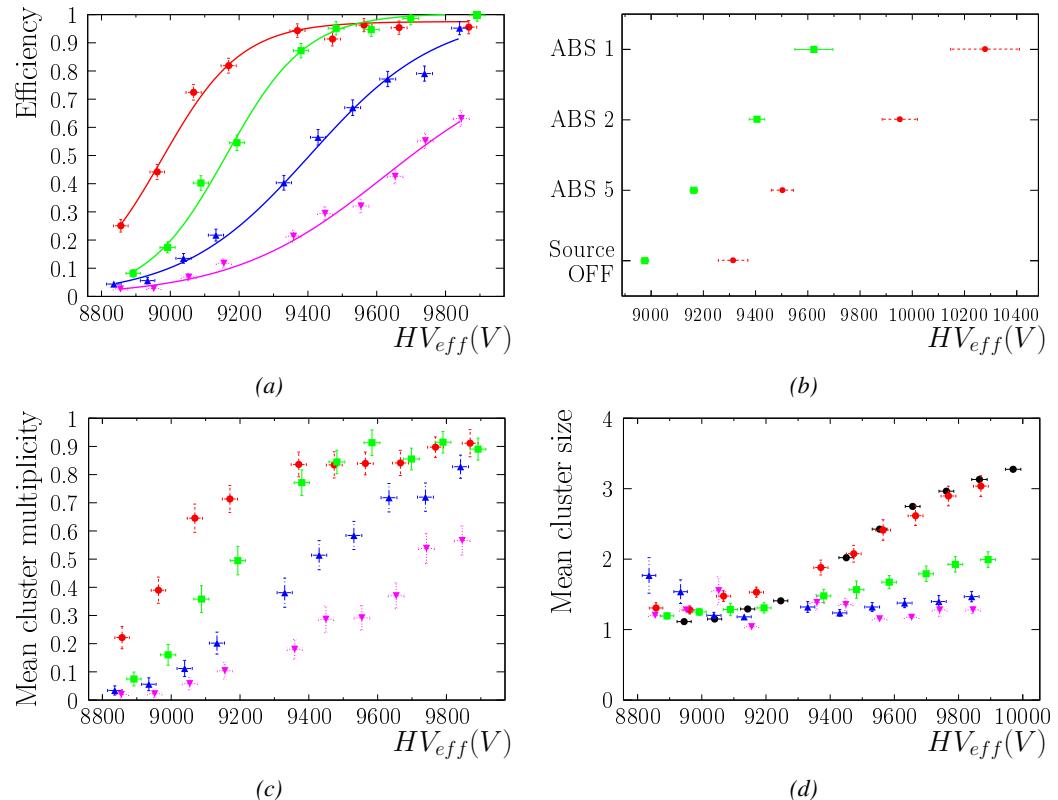


Figure 5.15

⁷⁴⁹ **5.3 Longevity tests at GIF++**

⁷⁵⁰ Longevity studies imply a monitoring of the performance of the detectors probed using a high inten-
⁷⁵¹ sity muon beam in a irradiated environment by periodically measuring their rate capability, the dark
⁷⁵² current running through them and the bulk resistivity of the Bakelite composing their electrodes.
⁷⁵³ GIF++, with its very intense ^{137}Cs source, provides the perfect environment to perform such kind
⁷⁵⁴ of tests. Assuming a maximum acceleration factor of 3, it is expected to accumulate the equivalent
⁷⁵⁵ charge in 1.7 years.

⁷⁵⁶ As the maximum background is found in the endcap, the choice naturally was made to focus the
⁷⁵⁷ GIF++ longevity studies on endcap chambers. Most of the RPC system was installed in 2007. Nev-
⁷⁵⁸ ertheless, the large chambers in the fourth endcap (RE4/2 and RE4/3) have been installed during
⁷⁵⁹ LS1 in 2014. The Bakelite of these two different productions having different properties, four spare
⁷⁶⁰ chambers of the present system were selected, two RE2,3/2 spares and two RE4/2 spares. Having
⁷⁶¹ two chambers of each type allows to always keep one of them non irradiated as reference, the per-
⁷⁶² formance evolution of the irradiated chamber being then compared through time to the performance
⁷⁶³ of the non irradiated one.

⁷⁶⁴ The performance of the detectors under different level of irradiation is measured periodically dur-
⁷⁶⁵ ing dedicated test beam periods using the H4 muon beam. In between these test beam periods, the
⁷⁶⁶ two RE2,3/2 and RE4/2 chambers selected for this study are irradiated by the ^{137}Cs source in order
⁷⁶⁷ to accumulate charge and the gamma background is monitored, as well as the currents. The two
⁷⁶⁸ remaining chambers are kept non-irradiated as reference detectors. Due to the limited gas flow in
⁷⁶⁹ GIF++, the RE4 chamber remained non-irradiated until end of November 2016 where a new mass
⁷⁷⁰ flow controller has been installed allowing for bigger volumes of gas to flow in the system.

⁷⁷¹ Figures 5.16 and 5.17 give us for different test beam periods, and thus for increasing integrated
⁷⁷² charge through time, a comparison of the maximum efficiency, obtained using a sigmoid-like func-
⁷⁷³ tion, and of the working point of both irradiated and non irradiated chambers [8]. No aging is yet to
⁷⁷⁴ see from this data, the shifts in γ rate per unit area in between irradiated and non irradiated detec-
⁷⁷⁵ tors and RE2 and RE4 types being easily explained by a difference of sensitivity due to the various
⁷⁷⁶ Bakelite resistivities of the HPL electrodes used for the electrode production.

⁷⁷⁷ Collecting performance data at each test beam period allows us to extrapolate the maximum effi-
⁷⁷⁸ ciency for a background hit rate of $300\text{ Hz}/\text{cm}^2$ corresponding to the expected HL-LHC conditions.
⁷⁷⁹ Aging effects could emerge from a loss of efficiency with increasing integrated charge over time,
⁷⁸⁰ thus Figure 5.18 helps us understand such degradation of the performance of irradiated detectors in
⁷⁸¹ comparison with non irradiated ones. The final answer for an eventual loss of efficiency is given in
⁷⁸² Figure 5.19 by comparing for both irradiated and non irradiated detectors the efficiency sigmoids
⁷⁸³ before and after the longevity study. Moreover, to complete the performance information, the Bake-
⁷⁸⁴ lite resistivity is regularly measured thanks to Ag scans (Figure 5.20) and the noise rate is monitored
⁷⁸⁵ weekly during irradiation periods (Figure 5.21). At the end of 2016, no signs of aging were observed
⁷⁸⁶ and further investigation is needed to get closer to the final integrated charge requirements proposed
⁷⁸⁷ for the longevity study of the present CMS RPC sub-system.

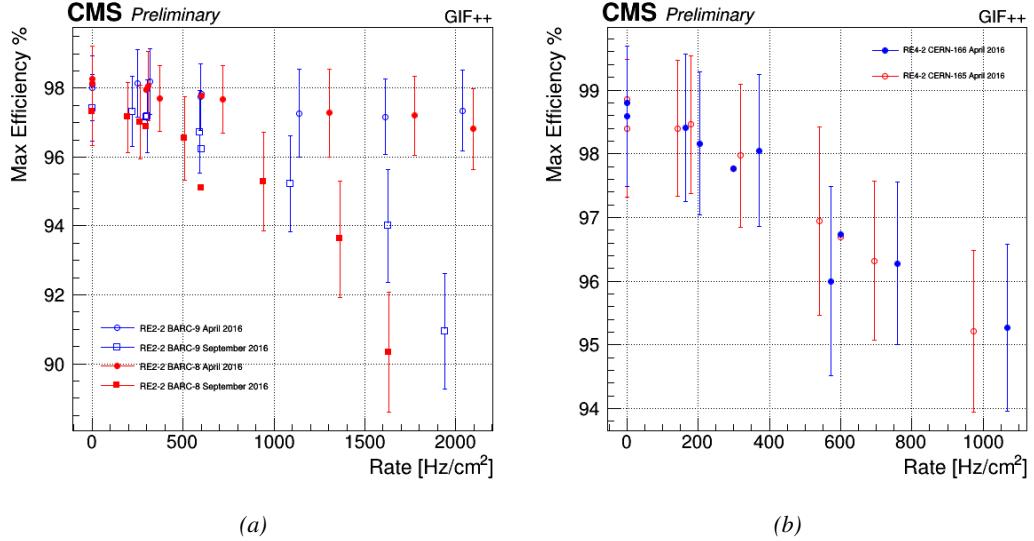


Figure 5.16: Evolution of the maximum efficiency for RE2 (5.16a) and RE4 (5.16b) chambers with increasing extrapolated γ rate per unit area at working point. Both irradiated (blue) and non irradiated (red) chambers are shown.

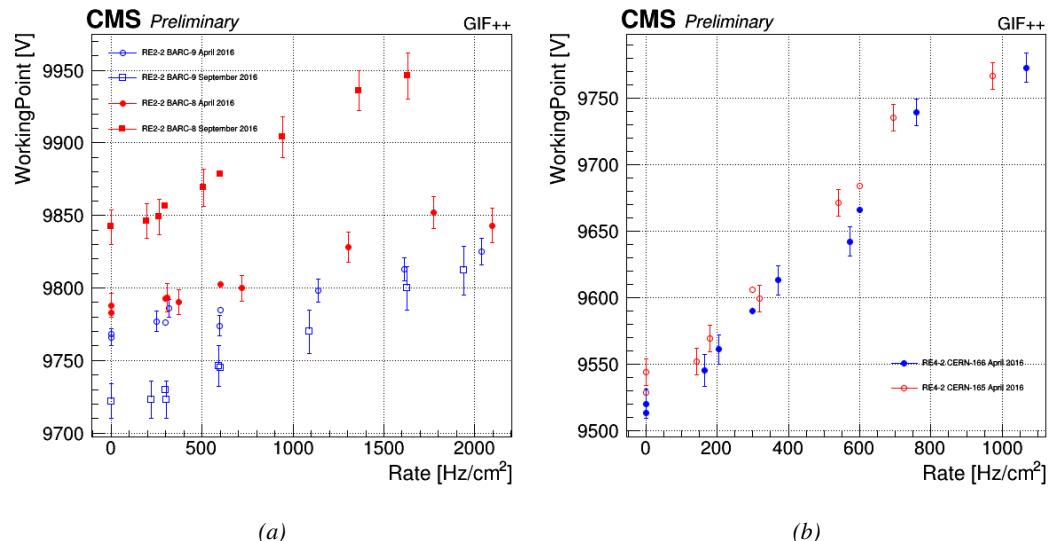


Figure 5.17: Evolution of the working point for RE2 (5.17a) and RE4 (5.17b) with increasing extrapolated γ rate per unit area at working point. Both irradiated (blue) and non irradiated (red) chambers are shown.

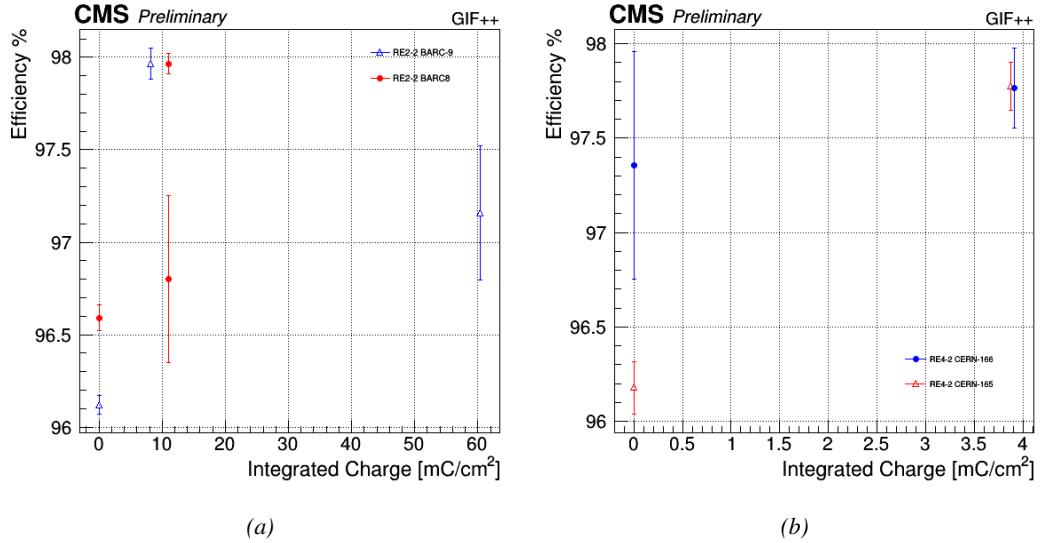


Figure 5.18: Evolution of the maximum efficiency at HL-LHC conditions, i.e. a background hit rate per unit area of $300 \text{ Hz}/\text{cm}^2$, with increasing integrated charge for RE2 (5.18a) and RE4 (5.18b) detectors. Both irradiated (blue) and non irradiated (red) chambers are shown. The integrated charge for non irradiated detectors is recorded during test beam periods and stays small with respect to the charge accumulated in irradiated chambers.

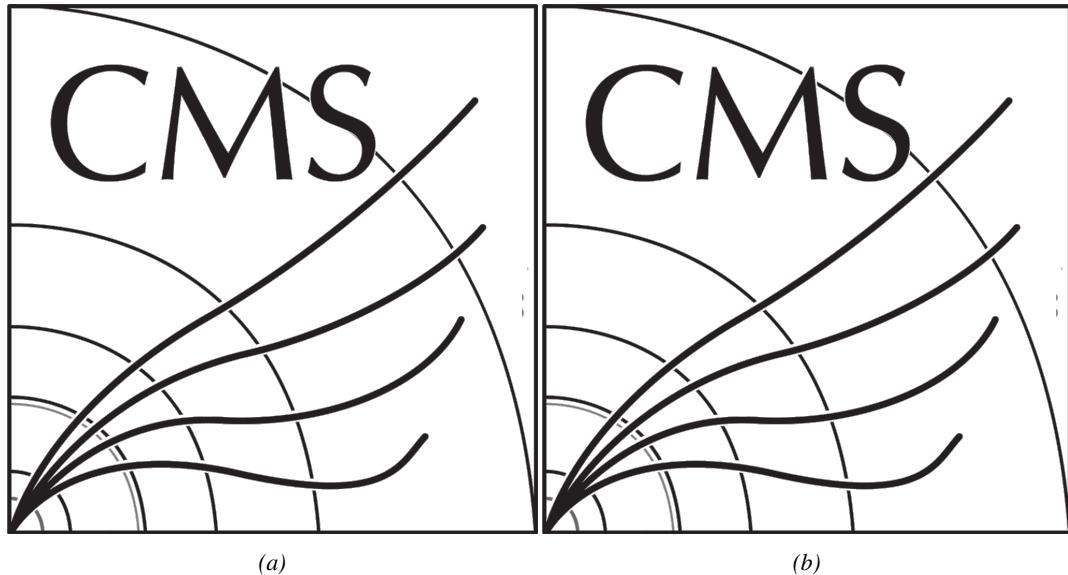


Figure 5.19: Comparison of the efficiency sigmoid before (triangles) and after (circles) irradiation for RE2 (5.19a) and RE4 (5.19b) detectors. Both irradiated (blue) and non irradiated (red) chambers are shown.

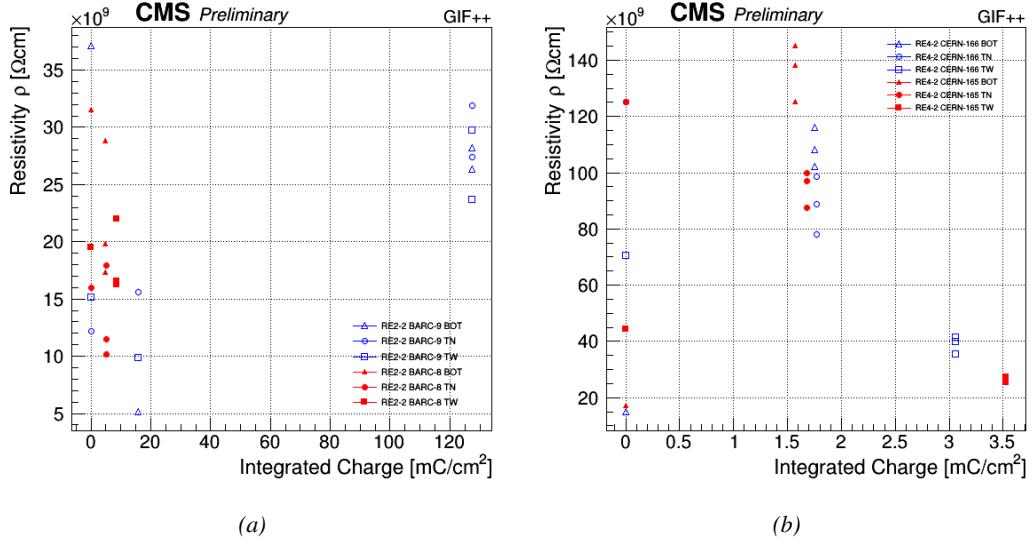


Figure 5.20: Evolution of the Bakelite resistivity for RE2 (5.20a) and RE4 (5.20b) detectors. Both irradiated (blue) and non-irradiated (red) chambers are shown.

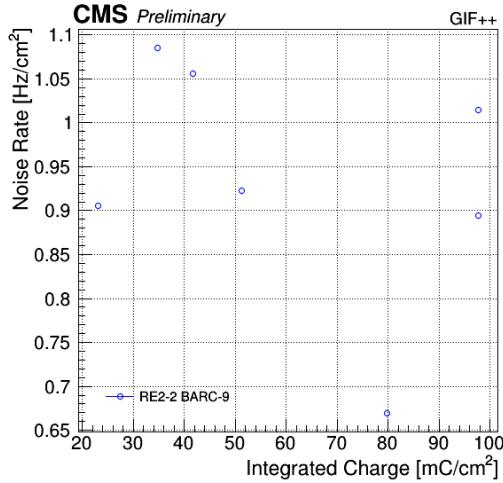


Figure 5.21: Evolution of the noise rate per unit area for the irradiated chamber RE2-2-BARC-9 only.

5.3.1 Description of the Data Acquisition

For the longevity studies, four spare chambers of the present system are used. Two spare RPCs of the RE2,3 stations as well as two spare RPCs from the new RE4 stations have been mounted in a Trolley. Six RE4 gaps are also placed in the trolley. The trolley is placed inside the GIFT++ in the upstream region of the bunker, taking the cesium source as a reference. The trolley is oriented for the detection surface of the chambers to be orthogonal to the beam line. The system can be moved along the orthogonal plane in order to have the beam in all η -partitions. For the aging the trolley is

796 moved outside the beam line and is placed in a distance of 5.2 m to the source, which irradiates the
 797 bunker using an attenuation filter of 2.2 which corresponds to a fluence of 10^7 gamma/cm^2 .

798 During GIF++ operation, the data collected can be divided into different categories as several
 799 parameters are monitored in addition to the usual RPC performance data. On one hand, to know
 800 the performance of a chamber, it is need to measure its efficiency and to know the background
 801 conditions in which it is operated. To do this, the hit signals from the chamber are recorded and
 802 stored in a ROOT file via a Data Acquisition (DAQ) software. On the other hand, it is also very
 803 important to monitor parameters such as environmental pressure and temperature, gas temperature
 804 and humidity, RPC HV, LV, and currents, or even source and beam status. This is done through the
 805 GIF++ web Detector Control Software (DCS) that stores this information in a database.

806 Two different types of tests are conducted on RPCs via the DAQ. Indeed, the performance of the
 807 detectors is measured periodically during dedicated test beam periods using the H4 muon beam. In
 808 between these test beam periods, when the beam is not available, the chambers are irradiated by the
 809 ^{137}Cs in order to accumulate deposited charge and the gamma background is measured.

810 RPCs under test are connected through LVDS cables to V1190A Time-to-Digital Converter
 811 (TDC) modules manufactured by CAEN. These modules, located in the rack area outside of the
 812 bunker, get the logic signals sent by the chambers and save them into their buffers. Due to the
 813 limited size of the buffers, the collected data is regularly erased and replaced. A trigger signal is
 814 needed for the TDC modules to send the useful data to the DAQ computer via a V1718 CAEN USB
 815 communication module.

816 In the case of performance test, the trigger signal used for data acquisition is generated by the
 817 coincidence of three scintillators. A first one is placed upstream outside of the bunker, a second one
 818 is placed downstream outside of the bunker, while a third one is placed in front of the trolley, close by
 819 the chambers. Every time a trigger is sent to the TDCs, i.e. every time a muon is detected, knowing
 820 the time delay in between the trigger and the RPC signals, signals located in the right time window
 821 are extracted from the buffers and saved for later analysis. Signals are taken in a time window of
 822 400 ns centered on the muon peak (here we could show a time spectrum). On the other hand, in the
 823 case of background rate measurement, the trigger signal needs to be "random" not to measure muons
 824 but to look at gamma background. A trigger pulse is continuously generated at a rate of 300 Hz using
 825 a dual timer. To integrate an as great as possible time, all signals contained within a time window of
 826 10us prior to the random trigger signal are extracted form the buffers and saved for further analysis
 827 (here another time spectrum to illustrate could be useful, maybe even place both spectrum together
 828 as a single Figure).

829 The signals sent to the TDCs correspond to hit collections in the RPCs. When a particle hits
 830 a RPC, it induce a signal in the pickup strips of the RPC readout. If this signal is higher than the
 831 detection threshold, a LVDS signal is sent to the TDCs. The data is then organised into 4 branches
 832 keeping track of the event number, the hit multiplicity for the whole setup, and the time and channel
 833 profile of the hits in the TDCs.

834 **5.3.2 RPC current, environmental and operation parameter monitoring**

835 In order to take into account the variation of pressure and temperature between different data taking
 836 periods the applied voltage is corrected following the relationship :

$$837 \quad HV_{eff} = HV_{app} \times \left(0.2 + 0.8 \cdot \frac{P_0}{P} \times \frac{T}{T_0} \right) \quad (5.10)$$

837 where T_0 (=293 K) and P_0 (=990 mbar) are the reference values.

838 **5.3.3 Measurement procedure**

839 Insert a short description of the online tools (DAQ, DCS, DQM).

840 Insert a short description of the offline tools : tracking and efficiency algorithm.

841 Identify long term aging effects we are monitoring the rates per strip.

842 **5.3.4 Longevity studies results**

6

843

844

Investigation on high rate RPCs

845 **6.1 Rate limitations and ageing of RPCs**

846 **6.1.1 Low resistivity electrodes**

847 **6.1.2 Low noise front-end electronics**

848 **6.2 Construction of prototypes**

849 **6.3 Results and discussions**

7

850

851

Conclusions and outlooks

852 **7.1 Conclusions**

853 **7.2 Outlooks**

References

- [1] CERN. Geneva. LHC Experiments Committee. *The CMS muon project : Technical Design Report*. Tech. rep. CERN-LHCC-97-032. CMS Collaboration, 1997.
- [2] CERN. Geneva. LHC Experiments Committee. *Technical Proposal for the Phase-II Upgrade of the CMS Detector*. Tech. rep. CERN-LHCC-2015-010. CMS Collaboration, 2015.
- [3] CERN. Geneva. LHC Experiments Committee. *CMS, the Compact Muon Solenoid : technical proposal*. Tech. rep. CERN-LHCC-94-38. CMS Collaboration, 1994.
- [4] M. Abbrescia et al. “Study of long-term performance of CMS RPC under irradiation at the CERN GIF”. In: *NIMA* 533 (2004), pp. 102–106.
- [5] H.C. Kim et al. “Quantitative aging study with intense irradiation tests for the CMS forward RPCs”. In: *NIMA* 602 (2009), pp. 771–774.
- [6] S. Agosteo et al. “A facility for the test of large-area muon chambers at high rates”. In: *NIMA* 452 (2000), pp. 94–104.
- [7] PoS, ed. *CERN GIF ++ : A new irradiation facility to test large-area particle detectors for the high-luminosity LHC program*. Vol. TIPP2014. 2014, pp. 102–109.
- [8] M. Abbrescia et al. “Cosmic ray tests of double-gap resistive plate chambers for the CMS experiment”. In: *NIMA* 550 (2005), pp. 116–126.
- [9] A. Fagot. *GIF++ DAQ v4.0*. 2017. URL: https://github.com/afagot/GIF_DAQ.
- [10] CAEN. *Mod. V1190-VX1190 A/B, 128/64 Ch Multihit TDC*. 14th ed. 2016.
- [11] CAEN. *Mod. V1718 VME USB Bridge*. 9th ed. 2009.
- [12] W-Ie-Ne-R. *VME 6021-23 VXI*. 5th ed. 2016.
- [13] Wikipedia. *INI file*. 2017. URL: https://en.wikipedia.org/wiki/INI_file.
- [14] S. Carrillo A. Fagot. *GIF++ Offline Analysis v6*. 2017. URL: https://github.com/afagot/GIF_OfflineAnalysis.

A

878

879

880

A data acquisition software for CAEN VME TDCs

881 Certifying detectors in the perspective of HL-LHC required to develop tools for the GIF++ ex-
882 periment. Among them was the C++ Data Acquisition (DAQ) software that allows to make the com-
883 munications in between a computer and TDC modules in order to retrieve the RPC data [9]. In this
884 appendix, details about this software, as of how the software was written, how it functions and how
885 it can be exported to another similar setup, will be given.

886 A.1 GIF++ DAQ file tree

887 GIF++ DAQ source code is fully available on github at [https://github.com/afagot/GIF_](https://github.com/afagot/GIF_DAQ)
888 DAQ. The software requires 3 non-optional dependencies:

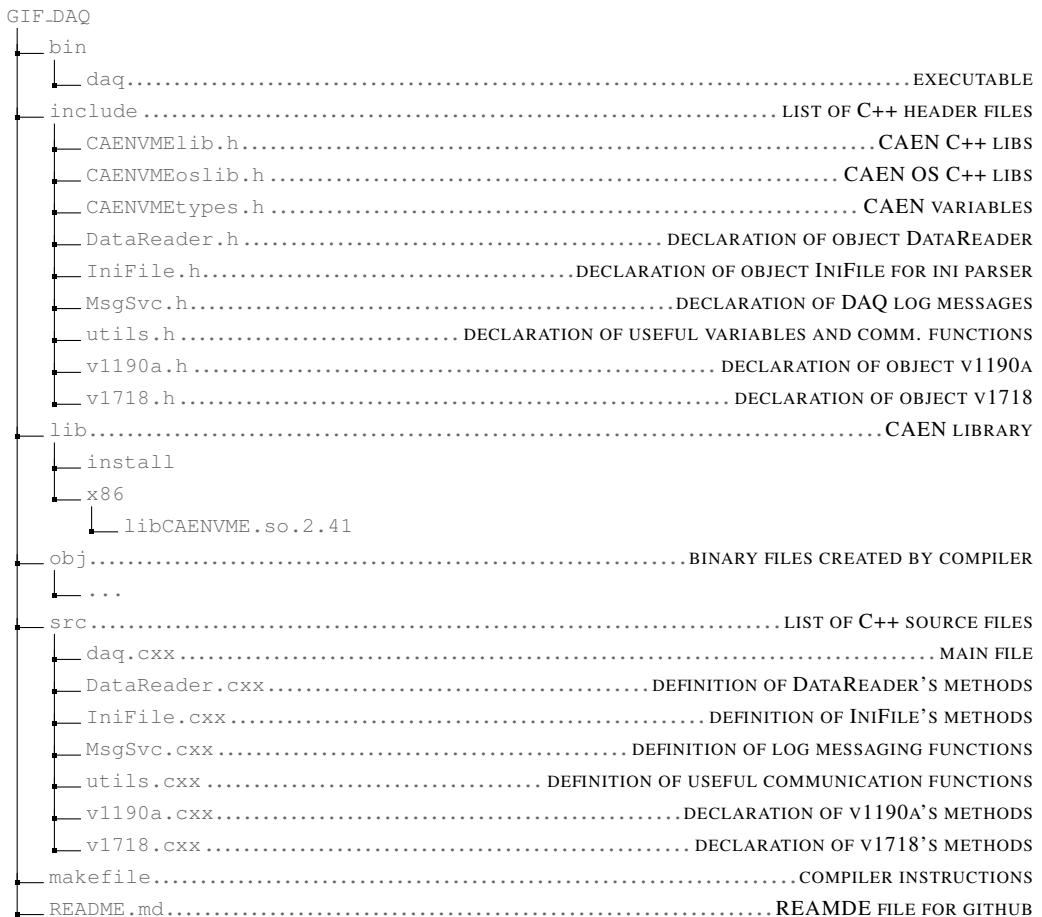
- 889 • CAEN USB Driver, to mount the VME hardware,
890 • CAEN VME Library, to communicate with the VME hardware, and
891 • ROOT, to organize the collected data into a TTree.

892 The CAEN VME library will not be packaged by distributions and will need to be installed man-
893 ually. To compile the GIF++ DAQ project via a terminal, from the DAQ folder use the command:

894
895 `make`

896 The source code tree is provided below along with comments to give an overview of the files' con-
897 tent. The different objects created for this project (`v1718`, `v1190a`, `IniFile` & `DataReader`) will be
898 described in details in the following sections.

899



900 A.2 Usage of the DAQ

901 GIF++ DAQ, as used in GIF++, is not a standalone software. Indeed, the system being more complexe,
 902 the DAQ only is a sub-layer of the software architecture developped to control and monitor
 903 the RPCs that are placed into the bunker for performance study in an irradiated environment. The top
 904 layer of GIF++ is a Web Detector Control System (webDCS) application. The DAQ is only called
 905 by the webDCS when data needs to be acquired. The webDCS operates the DAQ through command
 906 line. To start the DAQ, the webDCS calls:

907
 908 bin/daq /path/to/the/log/file/in/the/output/data/folder

909 where /path/to/the/log/file/in/the/output/data/folder is the only argument required. This
 910 log file is important for the webDCS as this file contains all the content of the communication of the
 911 webDCS and the different systems monitored by the webDCS. Its content is constantly displayed
 912 during data taking for the users to be able to follow the operations. The communication messages
 913 are normally sent to the webDCS log file via the functions declared in file MsgSvc.h, typically
 914 MSG_INFO(string message).

915

916 A.3 Description of the readout setup

917 The CMS RPC setup at GIF++ counts 5 V1190A Time-to-Digital Converter (TDC) manufactured
 918 by CAEN [10]. V1190A are VME units accepting 128 independent Multi-Hit/Multi-Event TDC
 919 channels whose signals are treated by 4 100 ps high performance TDC chips developed by CERN
 920 / ECP-MIC Division. The communication between the computer and the TDCs to transfer data is
 921 done via a V1718 VME master module also manufactured by CAEN and operated from a USB
 922 port [11]. These VME modules are all hosted into a 6U VME 6021 powered crate manufactured by
 923 W-Ie-Ne-R than can accommodate up to 21 VME bus cards [12]. These 3 components of the DAQ
 924 setup are shown in Figure A.1.

925

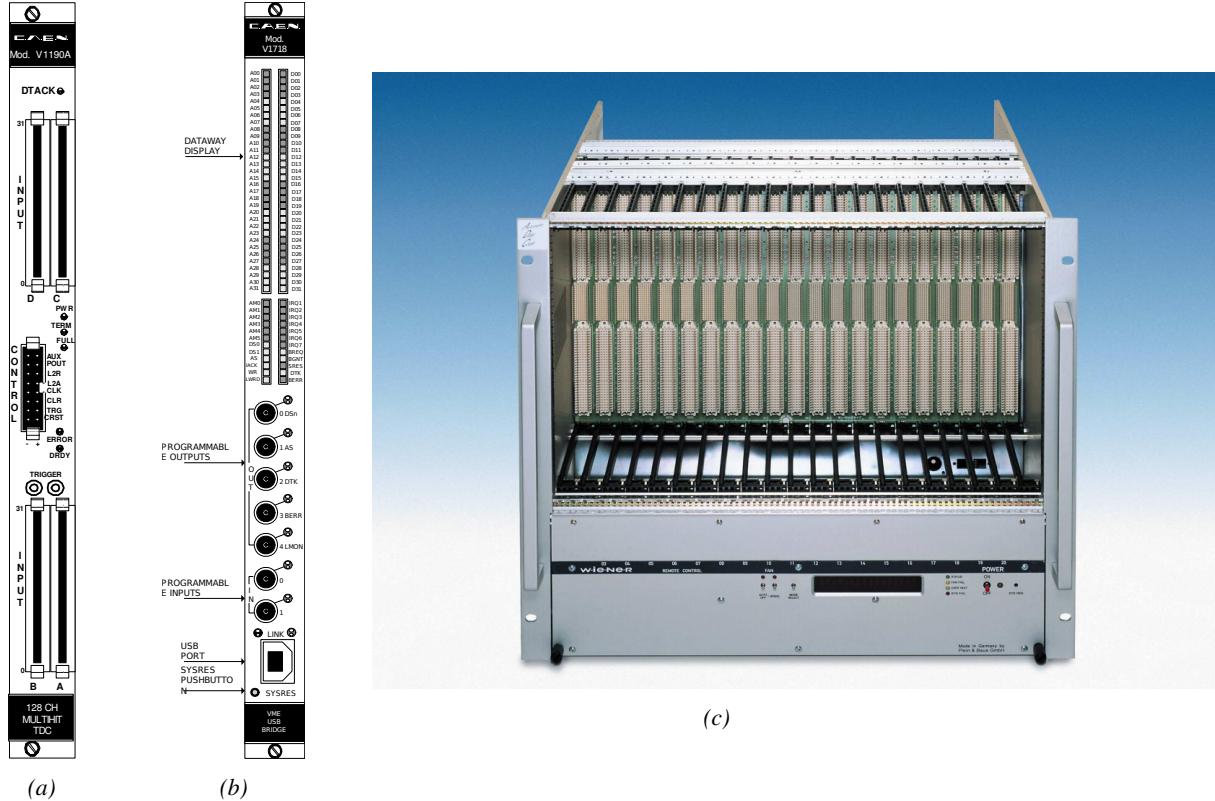


Figure A.1: (A.1a) View of the front panel of a V1190A TDC module [10]. (A.1b) View of the front panel of a V1718 Bridge module [11]. (A.1c) View of the front panel of a 6U 6021 VME crate [12].

926

A.4 Data read-out

927 To efficiently perform a data readout algorithm, C++ objects to handle the VME modules (TDCs
 928 and VME bridge) have been created along with objects to store data and read the configuration file

929 that comes as an input of the DAQ software.

930

931 A.4.1 V1190A TDCs

932 The DAQ used at GIF takes profit of the *Trigger Matching Mode* offered by V1190A modules.
 933 This setting is enabled through the method `v1190a::SetTrigMatching (int ntdcs)` where `ntdcs`
 934 is the total number of TDCs in the setup this setting needs to be enabled for (Source Code A.1). A
 935 trigger matching is performed in between a trigger time tag, a trigger signal sent into the TRIGGER
 936 input of the TDC visible on Figure A.1a, and the channel time measurements, signals recorded from
 937 the detectors under test in our case. Control over this data acquisition mode, explained through
 938 Figure A.2, is offered via 4 programmable parameters:

- 939 • **match window:** the matching between a trigger and a hit is done within a programmable time
 940 window. This is set via the method

```
941     void v1190a::SetTrigWindowWidth(UINT windowWidth, int ntdcs)
```

- 942 • **window offset:** temporal distance between the trigger tag and the start of the trigger matching
 943 window. This is set via the method

```
944     void v1190a::SetTrigWindowWidth(UINT windowWidth, int ntdcs)
```

- 945 • **extra search margin:** an extended time window is used to ensure that all matching hits are
 946 found. This is set via the method

```
947     void v1190a::SetTrigSearchMargin(UINT searchMargin, int ntdcs)
```

- 948 • **reject margin:** older hits are automatically rejected to prevent buffer overflows and to speed
 949 up the search time. This is set via the method

```
950     void v1190a::SetTrigRejectionMargin(UINT rejectMargin, int ntdcs)
```

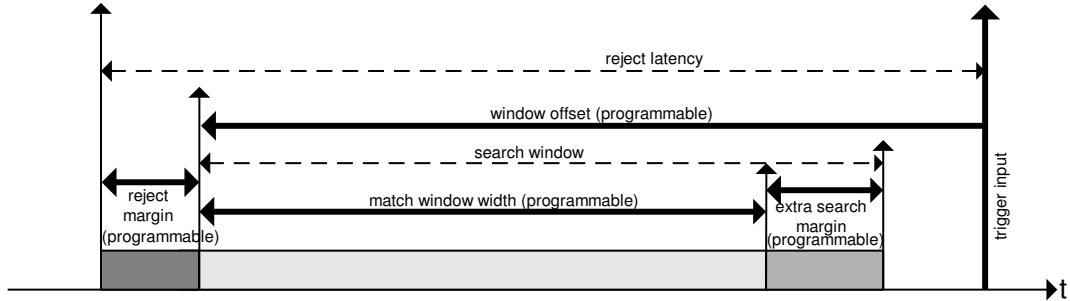


Figure A.2: Module V1190A Trigger Matching Mode timing diagram [10].

951 Each of these 4 parameters are given in number of clocks, 1 clock being 25 ns long. It is easy to
 952 understand at this level that there are 3 possible functioning settings:

- 953 • **1:** the match window is entirely contained after the trigger signal,

- 954 • **2:** the match window overlaps the trigger signal, or

- 955 • **3:** the match window is entirely contained before the trigger signal as displayed on Figure A.2.

In both the first and second cases, the sum of the window width and of the offset can be set to a maximum of 40 clocks, which corresponds to 1 μ s. Evidently, the offset can be negative, allowing for a longer match window, with the constraint of having the window ending at most 1 μ s after the trigger signal. In the third case, the maximum negative offset allowed is of 2048 clocks (12 bit) corresponding to 51.2 μ s, the match window being strictly smaller than the offset. In the case of GIF++, the choice has been made to use this last setting by delaying the trigger signal. During the studies performed in GIF++, both the efficiency of the RPCs, probed using a muon beam, and the noise or gamma background rate are monitored. The extra search and reject margins are left unused.

To probe the efficiency of RPC detectors, the trigger time tag is provided by the coïncidence of scintillators when a bunch of muons passes through GIF++ area is used to trigger the data acquisition. For this measurement, it is useful to reduce the match window width only to contain the muon information. Indeed, the delay in between a trigger signal and the detection of the corresponding muon in the RPC being very contant (typically a few tens of ns due to jitter and cable length), the muon signals are very localised in time. Thus, due to a delay of approximalety 325 ns in between the muons and the trigger, the settings where chosen to have a window width of 24 clocks (600 ns) centered on the muon peak thanks to a negative offset of 29 clocks (725 ns).

On the otherhand, monitoring the rates don't require for the DAQ to look at a specific time window. It is important to integrate enough time to have a robust measurement of the rate as the number of hits per time unit. The triggerring signal is provided by a pulse generator at a frequency of 300 Hz to ensure that the data taking occurs in a random way, with respect to beam physics, to probe only the irradiation spectrum on the detectors. The match window is set to 400 clocks (10 μ s) and the negative offset to 401 clocks as it needs to exceed the value of the match window.

```

978 class v1190a
{
    private :
        long             Handle;
        vector<Data32>   Address;
        CVDataWidth      DataWidth;
        CVAddressModifier AddressModifier;

    public:

        v1190a(long handle, IniFile *inifile, int ntdcs);
        ~v1190a();
        Data16 write_op_reg(Data32 address, int code, string error);
        Data16 read_op_reg(Data32 address, string error);
        void Reset(int ntdcs);
        void Clear(int ntdcs);
        void TestWR(Data16 value,int ntdcs);
        void CheckTDCStatus(int ntdcs);
        void CheckCommunication(int ntdcs);
        void SetTDCTestMode(Data16 mode,int ntdcs);
        void SetTrigMatching(int ntdcs);
        void SetTrigTimeSubtraction(Data16 mode,int ntdcs);
        void SetTrigWindowWidth(Uint windowHeight,int ntdcs);
        void SetTrigWindowOffset(Uint windowOffset,int ntdcs);
        void SetTrigSearchMargin(Uint searchMargin,int ntdcs);
        void SetTrigRejectionMargin(Uint rejectMargin,int ntdcs);
        void GetTrigConfiguration(int ntdcs);
        void SetTrigConfiguration(IniFile *inifile,int ntdcs);
        void SetTDCDetectionMode(Data16 mode,int ntdcs);
        void SetTDCResolution(Data16 lsb,int ntdcs);
        void SetTDCDeadTime(Data16 time,int ntdcs);
        void SetTDCHeadTrailer(Data16 mode,int ntdcs);
        void SetTDCEventSize(Data16 size,int ntdcs);
        void SwitchChannels(IniFile *inifile,int ntdcs);
        void SetIRQ(Data32 level, Data32 count,int ntdcs);
        void SetBlockTransferMode(Data16 mode,int ntdcs);
        void Set(IniFile *inifile,int ntdcs);
        void CheckStatus(CVErrorCodes status) const;
        int ReadBlockD32(Uint tdc, const Data16 address,
                         Data32 *data, const Uint words, bool ignore_berr);
        Uint Read(RAWData *DataList,int ntdcs);
};

980

```

Source Code A.1: Description of C++ object v1190a.

The v1190a object, defined in the DAQ software as in Source Code A.1, offers the possibility to concatenate all TDCs in the readout setup into a single object containing a list of hardware addresses (addresses to access the TDCs' buffer through the VME crate) and each constructor and method acts on the list of TDCs.

985

986 A.4.2 DataReader

987 Enabled thanks to v1190a::SetBlockTransferMode(Data16 mode, int ntdcs), the data transfer
 988 is done via Block Transfer (BLT). Using BLT allows to tranfer a fixed number of events called a
 989 *block*. This is used together with an Almost Full Level (AFL) of the TDCs' output buffers, defined

990 through `v1190a::SetIRQ(Data32 level, Data32 count, int ntdcs)`. This AFL gives the maximum amount of 32735 words (16 bits, corresponding to the depth of a TDC output buffer) that can
 991 written in a buffer before an Interrupt Request (IRQ) is generated and seen by the VME Bridge,
 992 stopping the data acquisition to transfer the content of each TDC buffers before resuming. For each
 993 trigger, 6 words or more are written into the TDC buffer:

- 995 • a **global header** providing information of the event number since the beginning of the data
 acquisition,
- 997 • a **TDC header**,
- 998 • the **TDC data** (*if any*), 1 for each hit recorded during the event, providing the channel and the
 time stamp associated to the hit,
- 1000 • a **TDC error** providing error flags,
- 1001 • a **TDC trailer**,
- 1002 • a **global trigger time tag** that provides the absolute trigger time relatively to the last reset,
 and
- 1004 • a **global trailer** providing the total word count in the event.

1005 As previously described in Section 4.4.3, CMS RPC FEEs provide us with 100 ns long LVDS
 1006 output signals that are injected into the TDCs' input. Any avalanche signal that gives a signal above
 1007 the FEEs threshold is thus recorded by the TDCs as a hit within the match window. Each hit is
 1008 assigned to a specific TDC channel with a time stamp, with a precision of 100 ps. The reference
 1009 time, $t_0 = 0$, is provided by the beginning of the match window. Thus for each trigger, coming from
 1010 a scintillator coïncidence or the pulse generator, a list of hits is stored into the TDCs' buffers and
 1011 will then be transferred into a ROOT Tree.

1012 When the BLT is used, it is easy to understand that the maximum number of words that have
 1013 been set as ALF will not be a finite number of events or, at least, the number of events that would
 1014 be recorded into the TDC buffers will not be a multiple of the block size. In the last BLT cycle to
 1015 transfer data, the number of events to transfer will most probably be lower than the block size. In that
 1016 case, the TDC can add fillers at the end of the block but this option requires to send more data to the
 1017 computer and is thus a little slower. Another solution is to finish the transfer after the last event by
 1018 sending a bus error that states that the BLT reached the last event in the pile. This method has been
 1019 chosen in GIF++.

1021 Due to irradiation, an event in GIF++ can count up to 300 words per TDC. A limit of 4096 words
 1022 (12 bits) has been set to generate IRQ which represent from 14 to almost 700 events depending on
 1023 the average of hits collected per event. Then the block size has been set to 100 events with enabled
 1024 bus errors. When an AFL is reached for one of the TDCs, the VME bridge stops the acquisition by
 1025 sending a BUSY signal.

1027

1028 The data is then transferred one TDC at a time into a structure called `RAWData` (Source Code A.2).

```
1029
1030   struct RAWData{
1031     vector<int>           *EventList;
1032     vector<int>           *NHitsList;
1033     vector<int>           *QFlagList;
1034     vector<vector<int>>  *Channellist;
1035     vector<vector<float>> *TimeStampList;
1036   };
```

1031 *Source Code A.2: Description of data holding C++ structure `RAWData`.*

1032 In order to organize the data transfer and the data storage, an object called `DataReader` was
1033 created (Source Code A.3). On one hand, it has `v1718` and `v1190a` objects as private members for
1034 communication purposes, such as VME modules settings via the configuration file `*iniFile` or data
1035 read-out through `v1190a::Read()` and on the other hand, it contains the structure `RAWData` that allows
1036 to organise the data in vectors reproducing the tree structure of a ROOT file.

```
1037
1038   class DataReader
1039   {
1040     private:
1041       bool      StopFlag;
1042       IniFile *iniFile;
1043       Data32  MaxTriggers;
1044       v1718   *VME;
1045       int      nTDCs;
1046       v1190a  *TDCs;
1047       RAWData TDCData;
1048
1049     public:
1050       DataReader();
1051       virtual ~DataReader();
1052       void      SetIniFile(string inifilename);
1053       void      SetMaxTriggers();
1054       Data32  GetMaxTriggers();
1055       void      SetVME();
1056       void      SetTDC();
1057       int      GetQFlag(Uint it);
1058       void      Init(string inifilename);
1059       void      FlushBuffer();
1060       void      Update();
1061       string  GetFileName();
1062       void      WriteRunRegistry(string filename);
1063       void      Run();
1064   };
```

1039 *Source Code A.3: Description of C++ object `DataReader`.*

1040 Each event is transferred from `TDCData` and saved into branches of a ROOT `TTree` as 3 integers
1041 that represent the event ID (`EventCount`), the number of hits read from the TDCs (`nHits`), and the
1042 quality flag that provides information for any problem in the data transfer (`qflag`), and 2 lists of
1043 `nHits` elements containing the fired TDC channels (`TDCCh`) and their respective time stamps (`TDCTS`),
1044 as presented in Source Code A.4. The ROOT file file is named using information contained into
1045 the configuration file, presented in section A.5.2. The needed information is extracted using method
1046 `DataReader::GetFileName()` and allow to build the output filename format `ScanXXXXXX_HVX_DAQ.root`

1047 where ScanXXXXXX is a 6 digit number representing the scan number into GIFT++ database and HVX
 1048 the HV step within the scan that can be more than a single digit. An example of ROOT data file is
 1049 provided with Figure A.3.

```
1050
RAWData TDCData;
TFile *outputFile = new TFile(outputFileName.c_str(),"recreate");
TTree *RAWDataTree = new TTree("RAWData","RAWData");

int EventCount = -9;
int nHits = -8;
int qflag = -7;
vector<int> TDCCh;
vector<float> TDCTS;

RAWDataTree->Branch("EventNumber",&EventCount, "EventNumber/I");
RAWDataTree->Branch("number_of_hits",&nHits,"number_of_hits/I");
RAWDataTree->Branch("Quality_flag",&qflag,"Quality_flag/I");
RAWDataTree->Branch("TDC_channel",&TDCCh);
RAWDataTree->Branch("TDC_TimeStamp",&TDCTS);

1051
//...
//Here read the TDC data using v1190a::Read() and place it into
//TDCData for as long as you didn't collect the requested amount
//of data.
//...

for(Uint i=0; i<TDCData.EventList->size(); i++){
    EventCount = TDCData.EventList->at(i);
    nHits = TDCData.NHitsList->at(i);
    qflag = TDCData.QFlagList->at(i);
    TDCCh = TDCData.ChannelList->at(i);
    TDCTS = TDCData.TimeStampList->at(i);
    RAWDataTree->Fill();
}
```

1052 *Source Code A.4: Highlight of the data transfer and organisation within DataReader::Run() after the data has been collected into TDCData.*

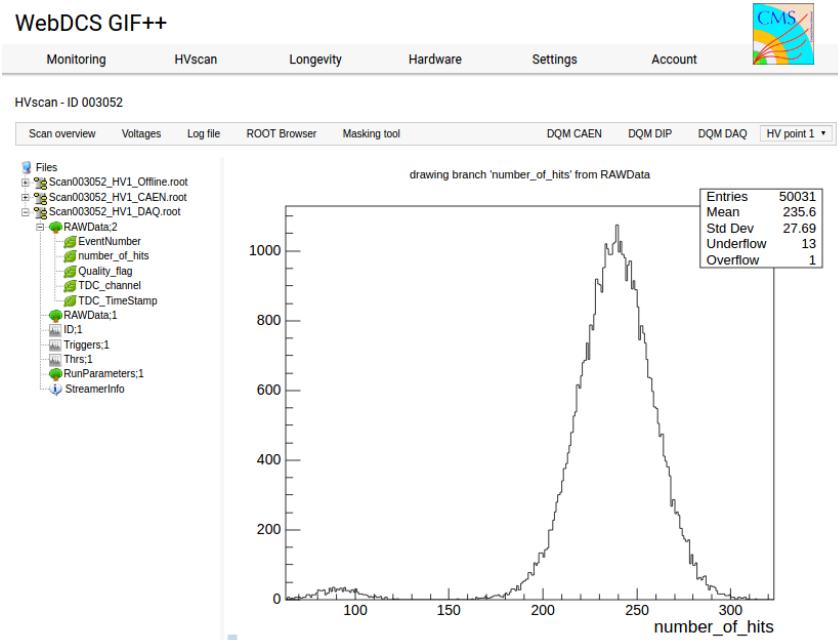


Figure A.3: Structure of the ROOT output file generated by the DAQ. The 5 branches (EventNumber, number_of_hits, Quality_flag, TDC_channel and TDC_TimeStamp) are visible on the left panel of the ROOT browser. On the right panel is visible the histogram corresponding to the variable nHits. In this specific example, there were approximately 50k events recorded to measure the gamma irradiation rate on the detectors. Each event is stored as a single entry in the TTree.

1053 A.4.3 Data quality flag

1054 Among the parameters that are recorded for each event, the quality flag, defined in Source Code A.5,
 1055 is determined on the fly by checking the data recorded by every single TDC. From method `v1190a::Read()`,
 1056 it can be understood that the content of each TDC buffer is readout one TDC at a time. Entries are
 1057 created in the data list for the first TDC and then, when the second buffer is readout, events corre-
 1058 sponding to entries that have already been created to store data for the previous TDC are added to
 1059 the existing list element. On the contrary, when an event entry has not been yet created in the data
 1060 list, a new entry is created.

```
1061
 1062   typedef enum _QualityFlag {
 1063     GOOD      = 1,
 1064     CORRUPTED = 0
 1065   } QualityFlag;
```

1063 *Source Code A.5: Definition of the quality flag `enum`.*

1064 It is possible that each TDC buffer contains a different number of events. In cases where the first
 1065 element in the buffer list is an event for corresponds to a new entry, the difference in between the
 1066 entry from the buffer and the last entry in the data list is recorded and checked. If it is greater than 1,
 1067 what should never be the case, the quality flag is set to CORRUPTED for this TDC and an empty entry
 1068 is created in the place of the missing ones. Missing entries are believe to be the result of a bad hold

1069 on the TDC buffers at the moment of the readout. Indeed, the software hold is effective only on 1
 1070 TDC at a time and no solution as been found yet to completely block the writting in the buffers when
 1071 an IRQ is received.

1072 At the end of each BLT cycle, the ID of the last entry stored for each TDC buffer is not recorded.
 1073 When starting the next cycle, if the first entry in the pile corresponds to an event already existing
 1074 in the list, the readout will start from this list element and will not be able to check the difference
 1075 in between this entry's ID and the one of the last entry that was recorded for this TDC buffer in
 1076 the previous cycle. In the case events were missing, the flag stays at its initial value of 0, which is
 1077 similar to CORRUPTED and it is assumed that then this TDC will not contribute to `number_of_hits`,
 1078 `TDC_channel` or `TDC_TimeStamp`.

1079 Finally, since there will be 1 `RAWData` entry per TDC for each event (meaning `nTDCs` entries,
 1080 referring to `DataReader` private attribute), the individual flags of each TDC will be added together.
 1081 The final format is an integer composed `nTDCs` digits where each digit is the flag of a specific TDC.
 1082 This is constructed using powers of 10 like follows:

```
1083     TDC 0: QFlag = 100 × _QualityFlag
1084     TDC 1: QFlag = 101 × _QualityFlag
1085     ...
1086     TDC N: QFlag = 10N × _QualityFlag
```

1087 and the final flag to be with N digits:

```
1088     QFlag = n....3210
```

1089 each digit being 1 or 0. Below is given an example with a 4 TDCs setup.

1090 If all TDCs were good : `QFlag = 1111`,

1091 but if TDC 2 was corrupted : `QFlag = 1011`.

1092 When data taking is over and the data contained in the dynamical `RAWData` structure is transferred
 1093 to the ROOT file, all the 0s are changed into 2s by calling the method `DataReader::GetQFlag()`.
 1094 This will help translating the flag without knowing the number of TDCs beforehand. Indeed, a flag
 1095 111 could be due to a 3 TDC setup with 3 good individual TDC flags or to a more than 3 TDC setup
 1096 with TDCs those ID is greater than 2 being CORRUPTED, thus giving a 0.

1097 The quality flag has been introduced quite late, in October 2017 only, to the list of GIFT++ DAQ
 1098 parameters to be recorded into the output ROOT file. Before this addition, the missing data, corrupting
 1099 the quality for the offline analysis, was contributing to artificially fill data with lower multiplicity.
 1100 Looking at `TBranch number_of_hits` provides an information about the data of the full GIFT++
 1101 setup. When a TDC is not able to transfer data for a specific event, the effect is a reduction of the
 1102 total number of hits recorded in the full setup, this is what can be seen from Figure A.4. After offline
 1103 reconstruction detector by detector, the effect of missing events can be seen in the artificially filled
 1104 bin at multiplicity 0 shown in Figure A.5. Nonetheless, for data with high irradiation levels, as it is
 1105 the case for Figure A.5a, discarding the fake multiplicity 0 data can be done easily during the offline
 1106 analysis. At lower radiation, the missing events contribution becomes more problematic as the mul-
 1107 tiplicity distribution overlaps the multiplicity 0 and that in the same time the proportion of missing

1108 events decreases. Attempts to fit the distribution with a Poisson or skew distribution function were
 1109 not conclusive and this very problem has been at the origin of the quality flag that allows to give a
 1110 non ambiguous information about each event quality.

1111

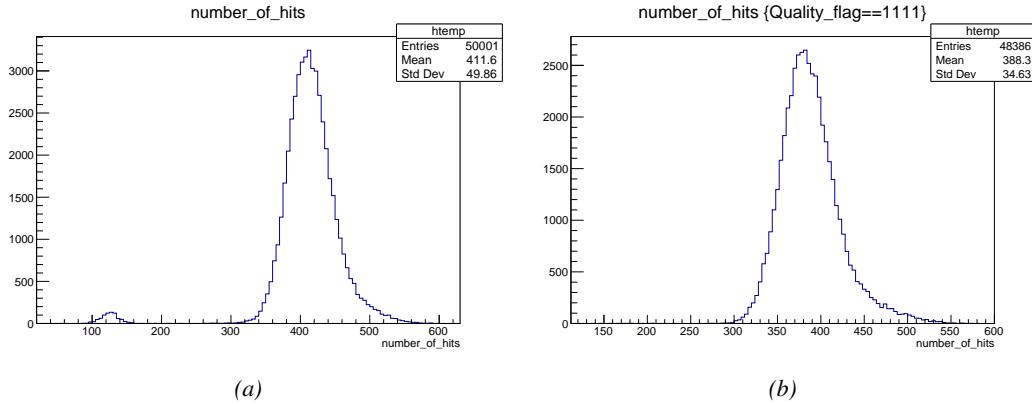


Figure A.4: The effect of the quality flag is explained by presenting the content of TBranch `number_of_hits` of a data file without `Quality_flag` in Figure A.4a and the content of the same TBranch for data corresponding to a `Quality_flag` where all TDCs were labelled as `GOOD` in Figure A.4b taken with similar conditions. It can be noted that the number of entries in Figure A.4b is slightly lower than in Figure A.4a due to the excluded events.

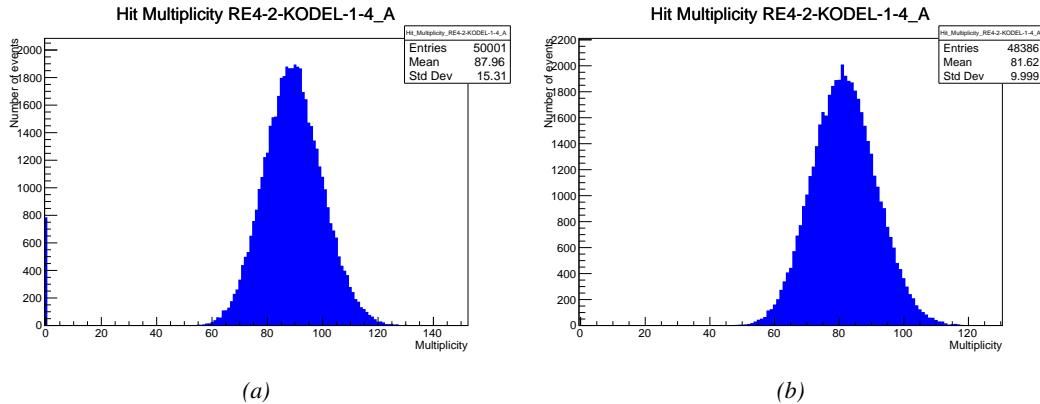


Figure A.5: Using the same data as previously showed in Figure A.4, the effect of the quality flag is explained by presenting the reconstructed hit multiplicity of a data file without `Quality_flag` in Figure A.5a and the reconstructed content of the same RPC partition for data corresponding to a `Quality_flag` where all TDCs were labelled as `GOOD` in Figure A.5b taken with similar conditions. The artificial high content of bin 0 is completely suppressed.

A.5 Communications

1112 To ensure data readout and dialog in between the machine and the TDCs or in between the webDCS
 1113 and the DAQ, different communication solutions were used. First of all, it is important to have a
 1114

1115 module to allow the communication in between the TDCs and the computer from which the DAQ
 1116 operates. When this communication is effective, shifters using the webDCS to control data taking
 1117 can thus send instructions to the DAQ.

1118

1119 A.5.1 V1718 USB Bridge

1120 In the previous section, the data transfer has been discussed. The importance of the `v1718` object
 1121 (Source Code A.6), used as private member of `DataReader`, was not explicated. VME master
 1122 modules are used for communication purposes as they host the USB port that connects the pow-
 1123 ered crate buffer to the computer where the DAQ is installed. From the source code point of view,
 1124 this object is used to control the communication status, by reading the returned error codes with
 1125 `v1718::CheckStatus()`, or to check for IRQs coming from the TDCs through `v1718::CheckIRQ()`.
 1126 Finally, to ensure that triggers are blocked at the hardware level, a NIM pulse is sent out of one of the
 1127 5 programmable outputs (`v1718::SendBUSY()`) to the VETO of the coincidence module where the
 1128 trigger signals originate from. As long as this signal is ON, no trigger can reach the TDCs anymore.
 1129

```
1130 class v1718{
1131     private:
1132         int             Handle;
1133         Data32          Data;           // Data
1134         CVIRQLevels    Level;         // Interrupt level
1135         CVAddressModifier AM;          // Addressing Mode
1136         CVDataWidth     DataSize;       // Data Format
1137         Data32          BaseAddress;   // Base Address
1138
1139     public:
1140         v1718(IniFile *inifile);
1141         ~v1718();
1142         long            GetHandle(void) const;
1143         int             SetData(Data16 data);
1144         Data16          GetData(void);
1145         int             SetLevel(CVIRQLevels level);
1146         CVIRQLevels    GetLevel(void);
1147         int             SetAM(CVAddressModifier am);
1148         CVAddressModifier GetAM(void);
1149         int             SetDatasize(CVDataWidth datasize);
1150         CVDataWidth     GetDataSize(void);
1151         int             SetBaseAddress(Data16 baseaddress);
1152         Data16          GetBaseAddress(void);
1153         void            CheckStatus(CVErrorCodes status) const;
1154         void            CheckIRQ();
1155         void            SetPulsers();
1156         void            SendBUSY(BusyLevel level);
1157     };
1158 }
```

1131 *Source Code A.6: Description of C++ object v1718.*

1132 A.5.2 Configuration file

1133 The DAQ software takes as input a configuration file written using INI standard [13]. This file is
 1134 partly filled with the information provided by the shifters when starting data acquisition using the
 1135 webDCS, as shown by Figure A.6. This information is written in section [`General`] and will later

1136 be stored in the ROOT file that contains the DAQ data as can be seen from Figure A.3. Indeed,
 1137 another `TTree` called `RunParameters` as well as the 2 histograms `ID`, containing the scan number,
 1138 start and stop time stamps, and `Triggers`, containing the number of triggers requested by the shifter,
 1139 are available in the data files. Moreover, `ScanID` and `HV` are then used to construct the file name
 1140 thanks to the method `DataReader::GetFileName()`.

WebDCS GIF++

Chamber	RE2-2-NPD-BARC-8	RE4-2-CERN-106	RE2-3-NPD-BARC-9	RE4-2-CERN-105	RE4-2-KODEL-1-4	Max triggers
HV _{eff} 1	8600	8500	8600	8500	6500	
HV _{eff} 2	8700	8600	8700	8600	6600	
HV _{eff} 3	8800	8700	8800	8700	6700	
HV _{eff} 4	8900	8800	8900	8800	6800	
HV _{eff} 5	9000	8900	9000	8900	6900	
HV _{eff} 6	9100	9000	9100	9000	7000	
HV _{eff} 7	9200	9100	9200	9100	7100	
HV _{eff} 8	9300	9200	9300	9200	7200	
HV _{eff} 9	9400	9300	9400	9300	7300	
HV _{eff} 10	9500	9400	9500	9400	7400	

Start HV scan

Figure A.6: WebDCS DAQ scan page. On this page, shifters need to choose the type of scan (Rate, Efficiency or Noise Reference scan), the gamma source configuration at the moment of data taking, the beam configuration, and the trigger mode. These information will be stored in the DAQ ROOT output. Are also given the minimal measurement time and waiting time after ramping up of the detectors is over before starting the data acquisition. Then, the list of HV points to scan and the number of triggers for each run of the scan are given in the table underneath.

1141 The rest of the information is written beforehand in the configuration file template, as explicated
 1142 in Source Code A.7, and contains the hardware addresses to the different VME modules in the
 1143 setup as well as settings for the TDCs. As the TDC settings available in the configuration file are not
 1144 supposed to be modified, an improvement would be to remove them from the configuration file and
 1145 to hardcode them inside of the DAQ code itself or to place them into a different INI file that would
 1146 host only the TDC settings to lower the probability for a bad manipulation of the configuration file
 1147 that can be modified from one of webDCS' menus.

1148

```

[General]
Tdcs=4
ScanID=$scanid
HV=$HV
RunType=$runtype
MaxTriggers=$maxtriggers
Beam=$beam
[VMEInterface]
Type=V1718
BaseAddress=0xFF0000
Name=VmeInterface
[TDC0]
Type=V1190A
BaseAddress=0x00000000
Name=Tdc0
StatusA00-15=1
StatusA16-31=1
StatusB00-15=1
StatusB16-31=1
StatusC00-15=1
StatusC16-31=1
StatusD00-15=1
StatusD16-31=1
[TDC1]
Type=V1190A
BaseAddress=0x11110000
Name=Tdc1
StatusA00-15=1
StatusA16-31=1
StatusB00-15=1
StatusB16-31=1
StatusC00-15=1
StatusC16-31=1
StatusD00-15=1
StatusD16-31=1
[TDC2]
Type=V1190A
BaseAddress=0x22220000
Name=Tdc2
StatusA00-15=1
StatusA16-31=1
StatusB00-15=1
StatusB16-31=1
StatusC00-15=1
StatusC16-31=1
StatusD00-15=1
StatusD16-31=1
[TDC3]
Type=V1190A
BaseAddress=0x44440000
Name=Tdc3
StatusA00-15=1
StatusA16-31=1
StatusB00-15=1
StatusB16-31=1
StatusC00-15=1
StatusC16-31=1
StatusD00-15=1
StatusD16-31=1
[TDCSettings]
TriggerExtraSearchMargin=0
TriggerRejectMargin=0
TriggerTimeSubtraction=0b1
TdcDetectionMode=0b01
TdcResolution=0b10
TdcDeadTime=0b00
TdcHeadTrailer=0b1
TdcEventSize=0b1001
TdcTestMode=0b0
BLTMode=1

```

Source Code A.7: INI configuration file template for 4 TDCs. In section [General], the number of TDCs is explicitated and information about the ongoing run is given. Then, there are sections for each and every VME modules. There buffer addresses are given and for the TDCs, the list of channels to enable is given. Finally, in section [TDCSettings], a part of the TDC settings are given.

1151 In order to retrieve the information of the configuration file, the object `IniFile` has been developed
 1152 to provide an INI parser, presented in Source Code A.8. It contains private methods returning a
 1153 boolean to check the type of line written in the file, whether a comment, a group header or a key line
 1154 (`IniFile::CheckIfComment()`, `IniFile::CheckIfGroup()` and `IniFile::CheckIfToken()`). The
 1155 key may sometimes be referred to as *token* in the source code. Moreover, the private element
 1156 `FileData` is a map of `const string` to `string` that allows to store the data contained inside the
 1157 configuration file via the public method `IniFile::GetFileData()` following the formatting (see
 1158 method `IniFile::Read()`):

```
1159     string group, token, value;
  // Get the field values for the 3 strings.
  // Then concatenate group and token together as a single string
  // with a dot separation.
  token = group + "." + token;
  FileData[token] = value;
```

1161 More methods have been written to translate the different keys into the right variable format
 1162 when used by the DAQ. For example, to get a `float` value out of the configuration file data, knowing
 1163 the group and the key needed, the method `IniFile::floatType()` can be used. It takes 3 arguments
 1164 being the group name and key name (both `string`), and a default `float` value used as exception in
 1165 the case the expected combination of group and key cannot be found in the configuration file. This
 1166 default value is then used and the DAQ continues on working after sending an alert in the log file for
 1167 further debugging.

```

1168 typedef map< const string, string > IniFileData;
1169
class IniFile{
    private:
        bool          CheckIfComment (string line);
        bool          CheckIfGroup(string line, string& group);
        bool          CheckIfToken(string line, string& key, string& value);
        string         FileName;
        IniFileData   FileData;
        int           Error;

    public:
        IniFile();
        IniFile(string filename);
        virtual      ~IniFile();

        // Basic file operations
        void          SetFileName(string filename);
        int           Read();
        int           Write();
        IniFileData   GetFileData();

        // Data readout methods
        Data32         addressType (string groupname, string keyname, Data32
→     defaultvalue);
        long          intType     (string groupname, string keyname, long
→     defaultvalue);
        long long    longType    (string groupname, string keyname, long long
→     defaultvalue );
        string         stringType  (string groupname, string keyname, string
→     defaultvalue );
        float         floatType   (string groupname, string keyname, float
→     defaultvalue );

        // Error methods
        string         GetErrorMsg();
    };

```

1170 *Source Code A.8: Description of C++ object `IniFile` used as a parser for INI file format.*

1171 A.5.3 WebDCS/DAQ intercommunication

1172 When shifters send instructions to the DAQ via the configuration file, it is the webDCS itself that
 1173 gives the start command to the DAQ and then the 2 softwares use inter-process communication
 1174 through file to synchronise themselves. This communication file is represented by the variable **const**
 1175 string __runstatuspath.

1176 On one side, the webDCS sends commands or status that are readout by the DAQ:

- 1177 • INIT, status sent when launching a scan and read via function `CtrlRunStatus(...)`,
- 1178 • START, command to start data taking and read via function `CheckSTART()`,
- 1179 • STOP, command to stop data taking at the end of the scan and read via function `CheckSTOP()`,
 1180 and
- 1181 • KILL, command to kill data taking sent by user and read via function `CheckKILL()`

and on the other, the DAQ sends status that are controled by the webDCS:

- 1183 ● `DAQ_RDY`, sent with `SendDAQReady()` to signify that the DAQ is ready to receive commands
1184 from the webDCS,
- 1185 ● `RUNNING`, sent with `SendDAQRunning()` to signify that the DAQ is taking data,
- 1186 ● `DAQ_ERR`, sent with `SendDAQError()` to signify that the DAQ didn't receive the expected com-
1187 mand from the webDCS or that the launch command didn't have the right number of argu-
1188 ments,
- 1189 ● `RD_ERR`, sent when the DAQ wasn't able to read the communication file, and
- 1190 ● `WR_ERR`, sent when the DAQ wasn't able to write into the communication file.

1191 **A.5.4 Example of inter-process communication cycle**

1192 Under normal conditions, the webDCS and the DAQ processes exchange commands and status via
1193 the file hosted at the address `__runstatuspath`, as explained in subsection A.5.3. An example of
1194 cycle is given in Table A.1. In this example, the steps 3 to 5 are repeated as long as the webDCS tells
1195 the DAQ to take data. A data taking cycle is the equivalent as what is called a *Scan* in GIFT++ jargon,
1196 referring to a set a runs with several HV steps. Each repetition of steps 3 to 5 is then equivalent to a
1197 single *Run*.

1198 At any moment during the data taking, for any reason, the shifter can decide that the data taking
1199 needs to be stopped before it reached the end of the scheduled cycle. Thus at any moment on the
1200 cycle, the content of the inter-process communication file will be changed to `KILL` and the DAQ will
1201 shut down right away. The DAQ checks for `KILL` signals every 5s after the TDCs configuration is
1202 over. So far, the function `CheckKILL()` has been used only inside of the data taking loop of method
1203 `DataReader::Run()` and thus, if the shifter decides to KILL the data taking during the TDC con-
1204 figuration phase or the HV ramping in between 2 HV steps, the DAQ will not be stopped smoothly
1205 and a *force kill* command will be sent to stop the DAQ process that is still awake on the computer.
1206 Improvements can be brought on this part of the software to make sure that the DAQ can safely
1207 shutdown at any moment.

1210 **A.6 Software export**

1211 In section A.2 was discussed the fact that the DAQ as written in its last version is not a standalone
1212 software. It is possible to make it a standalone program that could be adapted to any VME setup
1213 using V1190A and V1718 modules by creating a GUI for the software or by printing the log mes-
1214 sages that are normally printed in the webDCS through the log file, directly into the terminal. This
1215 method was used by the DAQ up to version 3.0 moment where the webDCS was completed. Also, it
1216 is possible to check branches of DAQ v2.X to have example of communication through a terminal.

1217 DAQ v2.X is nonetheless limited in it's possibilities and requires a lot of offline manual interven-
1218 tions from the users. Indeed, there is no communication of the software with the detectors' power
1219 supply system that would allow for a user a predefine a list of voltages to operate the detectors at

step	actions of webDCS	status of DAQ	__runstatuspath
1	launch DAQ ramp voltages ramping over wait for currents stabilization	readout of IniFile configuration of TDCs	INIT
2		configuration done send DAQ ready wait for START signal	DAQ_RDY
3	waiting time over send START		START
4	wait for run to end monitor DAQ run status	data taking ongoing check for KILL signal	RUNNING
5		run over send DAQ_RDY wait for next DCS signal	DAQ_RDY
6	ramp voltages ramping over wait for currents stabilization		DAQ_RDY
3	waiting time over send START		START
4	wait for run to end monitor DAQ run status	update IniFile information data taking ongoing check for KILL signal	RUNNING
5		run over send DAQ_RDY wait for next DCS signal	DAQ_RDY
7	send command STOP	DAQ shuts down	STOP

Table A.1: Inter-process communication cycles in between the webDCS and the DAQ through file string signals.

1221 and loop over to take data without any further manual intervention. In v2.X, the data is taken for a
1222 single detector setting and at the end of each run, the softwares asks the user if he intends on taking
1223 more runs. If so, the software invites the user to set the operating voltages accordingly to what is
1224 necessary and to manual update the configuration file in consequence. This working mode can be a
1225 very first approach before an evolution and has been successfully used by colleagues from different
1226 collaborations.

1227

1228 For a more robust operation, it is recommended to develop a GUI or a web application to inter-
1229 face the DAQ. Moreover, to limit the amount of manual interventions, and thus the probability to
1230 make mistakes, it is also recommended to add an extra feature into the DAQ by installing the HV
1231 Wrapper library provided by CAEN of which an example of use in a similar DAQ software devel-
1232 opped by a master student of UGent, and called TinyDAQ, is provided on UGent's github. Then, this
1233 HV Wrapper will help you communicating with and give instructions to a CAEN HV powered crate
1234 and can be added into the DAQ at the same level where the communication with the user was made
1235 in DAQ v2.X. In case you are using another kind of power system for your detectors, it is stringly
1236 adviced to use HV modules or crates that can be remotely controled via a using C++ libraries.

1237

B

1238

1239

Details on the offline analysis package

1240 The data collected in GIF++ thanks to the DAQ described in Appendix A is difficult to interpret by
1241 a human user that doesn't have a clear idea of the raw data architecture of the ROOT data files. In
1242 order to render the data human readable, a C++ offline analysis tool was designed to provide users
1243 with detector by detector histograms that give a clear overview of the parameters monitored during
1244 the data acquisition [14]. In this appendix, details about this software in the context of GIF++, as of
1245 how the software was written and how it functions will be given.

1246 B.1 GIF++ Offline Analysis file tree

1247 GIF++ Offline Analysis source code is fully available on github at https://github.com/afagot/GIF_OfflineAnalysis. The software requires ROOT as non-optionnal dependency
1248 as it takes ROOT files in input and write an output ROOT file containing histograms. To compile the
1249 GIF++ Offline Analysis project is compiled with cmake. To compile, first a build/ directory must
1250 be created to compile from there:

```
1252 mkdir build  
1253 cd build  
1254 cmake ..  
1255 make  
1256 make install
```

1254 To clean the directory and create a new build directory, the bash script cleandir.sh can be used:

```
1255  
1256 ./cleandir.sh
```

1257 The source code tree is provided below along with comments to give an overview of the files' con-
1258 tent. The different objects created for this project (`Infrastructure`, `Trolley`, `RPC`, `Mapping`, `RPCHit`,
1259 `RPCCluster` and `Inifile`) will be described in details in the following sections.

1260

```

GIF_OfflineAnalysis
├── bin
│   └── offlineanalysis ..... EXECUTABLE
├── build..... CMAKE COMPILATION DIRECTORY
└── ...
    ├── include..... LIST OF C++ HEADER FILES
    │   ├── Cluster.h..... DECLARATION OF OBJECT RPCCLUSTER
    │   ├── Current.h..... DECLARATION OF GETCURRENT ANALYSIS MACRO
    │   ├── GIFTrolley.h..... DECLARATION OF OBJECT TROLLEY
    │   ├── Infrastructure.h..... DECLARATION OF OBJECT INFRASTRUCTURE
    │   ├── IniFile.h..... DECLARATION OF OBJECT INI FILE FORINI PARSER
    │   ├── Mapping.h..... DECLARATION OF OBJECT MAPPING
    │   ├── MsgSvc.h..... DECLARATION OF OFFLINE LOG MESSAGES
    │   ├── OfflineAnalysis.h..... DECLARATION OF DATA ANALYSIS MACRO
    │   ├── RPCDetector.h..... DECLARATION OF OBJECT RPC
    │   ├── RPCHit.h..... DECLARARION OF OBJECT RPCHIT
    │   ├── types.h..... DEFINITION OF USEFUL VARIABLE TYPES
    │   └── utils.h..... DECLARATION OF USEFUL FUNCTIONS
    ├── obj..... BINARY FILES CREATED BY COMPILER
    └── ...
        ├── src..... LIST OF C++ SOURCE FILES
        │   ├── Cluster.cc..... DEFINITION OF OBJECT RPCCLUSTER
        │   ├── Current.cc ..... DEFINITION OF GETCURRENT ANALYSIS MACRO
        │   ├── GIFTrolley.cc..... DEFINITION OF OBJECT TROLLEY
        │   ├── Infrastructure.cc..... DEFINITION OF OBJECT INFRASTRUCTURE
        │   ├── IniFile.cc..... DEFINITION OF OBJECT INI FILE FORINI PARSER
        │   ├── main.cc..... MAIN FILE
        │   ├── Mapping.cc..... DEFINITION OF OBJECT MAPPING
        │   ├── MsgSvc.cc..... DEFINITION OF OFFLINE LOG MESSAGES
        │   ├── OfflineAnalysis.cc..... DEFINITION OF DATA ANALYSIS MACRO
        │   ├── RPCDetector.cc..... DEFINITION OF OBJECT RPC
        │   ├── RPCHit.cc..... DEFINITION OF OBJECT RPCHIT
        │   └── utils.cc..... DEFINITION OF USEFUL FUNCTIONS
        ├── cleandir.sh..... BASH SCRIPT TO CLEAN BUILD DIRECTORY
        ├── CMakeLists.txt..... SET OF INSTRUCTIONS FOR CMAKE
        ├── config.h.in..... DEFINITION OF VERSION NUMBER
        └── README.md..... REAMDE FILE FOR GITHUB

```

1261

B.2 Usage of the Offline Analysis

1262

In order to use the Offline Analysis tool, it is necessary to know the Scan number and the HV Step of the run that needs to be analysed. This information needs to be written in the following format:

1264

1265

```
Scan00XXXX_HVY
```

1266

where XXXX is the scan ID and Y is the high voltage step (in case of a high voltage scan, data will be taken for several HV steps). This format corresponds to the base name of data files in the database

1267

1268 of the GIF++ webDCS. Usually, the offline analysis tool is automatically called by the webDCS at
 1269 the end of data taking or by a user from the webDCS panel if an update of the tool was brought.
 1270 Nonetheless, an expert can locally launch the analysis for tests on the GIF++ computer, or a user can
 1271 get the code on its local machine from github and download data from the webDCS for its own anal-
 1272 ysis. To launch the code, the following command can be used from the `GIF_OfflineAnalysis` folder:

1273
 1274 `bin/offlineanalysis /path/to/Scan00XXXX_HVY`

1275 where, `/path/to/Scan00XXXX_HVY` refers to the local data files. Then, the offline tool will by itself
 1276 take care of finding all available ROOT data files present in the folder, as listed below:

- 1277 • `Scan00XXXX_HVY_DAQ.root` containing the TDC data as described in Appendix ?? (events, hit
 1278 and timestamp lists), and
- 1279 • `Scan00XXXX_HVY_CAEN.root` containing the CAEN mainframe data recorded by the monitor-
 1280 ing tool webDCS during data taking (HVs and currents of every HV channels). This file is
 1281 created independently of the DAQ.

1282 **B.2.1 Output of the offline tool**

1283 **B.2.1.1 ROOT file**

1284 The analysis gives in output ROOT datafiles that are saved into the data folder and called using the
 1285 naming convention `Scan00XXXX_HVY_Offline.root`. Inside those, a list of `TH1` histograms can be
 1286 found. Its size will vary as a function of the number of detectors in the setup as each set of histograms
 1287 is produced detector by detector. For each partition of each chamber, can be found:

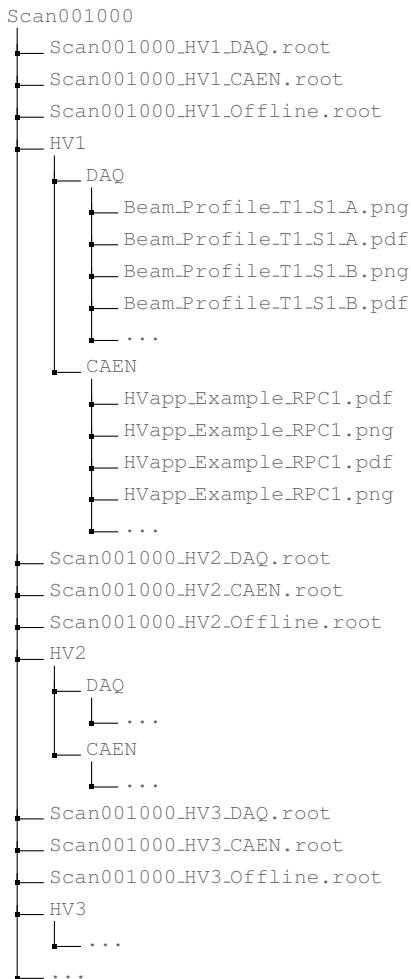
- 1288 • `Time_Profile_Tt_Sc_p` shows the time profile of all recorded events (number of events per
 1289 time bin),
- 1290 • `Hit_Profile_Tt_Sc_p` shows the hit profile of all recorded events (number of events per chan-
 1291 nel),
- 1292 • `Hit_Multiplicity_Tt_Sc_p` shows the hit multiplicity (number of hits per event) of all recorded
 1293 events (number of occurrences per multiplicity bin),
- 1294 • `Strip_Mean_Noise_Tt_Sc_p` shows noise/gamma rate per unit area for each strip in a se-
 1295 lected time range. After filters are applied on `Time_Profile_Tt_Sc_p`, the filtered version
 1296 of `Hit_Profile_Tt_Sc_p` is normalised to the total integrated time and active detection area
 1297 of a single channel,
- 1298 • `Strip_Activity_Tt_Sc_p` shows noise/gamma activity for each strip (normalised version of
 1299 previous histogram - strip activity = strip rate / average partition rate),
- 1300 • `Strip_Homogeneity_Tt_Sc_p` shows the *homogeneity* of a given partition ($\text{homogeneity} = \exp(-\text{strip rates standard deviation(strip rates in partition/average partition rate)})$),
- 1302 • `mask_Strip_Mean_Noise_Tt_Sc_p` shows noise/gamma rate per unit area for each masked
 1303 strip in a selected time range. Offline, the user can control the noise/gamma rate and decide to
 1304 mask the strips that are judged to be noisy or dead. This is done via the *Masking Tool* provided
 1305 by the webDCS,

- 1306 ● `mask_Strip_Activity_Tt_Sc_p` shows noise/gamma activity per unit area for each masked
1307 strip with respect to the average rate of active strips,
- 1308 ● `NoiseCSize_H_Tt_Sc_p` shows noise/gamma cluster size, a cluster being constructed out of
1309 adjacent strips giving a signal at the *same time* (hits within a time window of 25 ns),
- 1310 ● `NoiseCMult_H_Tt_Sc_p` shows noise/gamma cluster multiplicity (number of reconstructed
1311 clusters per event),
- 1312 ● `Chip_Mean_Noise_Tt_Sc_p` shows the same information than `Strip_Mean_Noise_Tt_Scp` us-
1313 ing a different binning (1 chip corresponds to 8 strips),
- 1314 ● `Chip_Activity_Tt_Sc_p` shows the same information than `Strip_Activity_Tt_Scp` using
1315 chip binning,
- 1316 ● `Chip_Homogeneity_Tt_Sc_p` shows the homogeneity of a given partition using chip binning,
- 1317 ● `Beam_Profile_Tt_Sc_p` shows the estimated beam profile when taking efficiency scan. This
1318 is obtained by filtering `Time_Profile_Tt_Sc_p` to only consider the muon peak where the
1319 noise/gamma background has been subtracted. The resulting hit profile corresponds to the
1320 beam profile on the detector channels,
- 1321 ● `L0_Efficiency_Tt_Sc_p` shows the level 0 efficiency that was estimated **without** muon track-
1322 ing,
- 1323 ● `MuonCSize_H_Tt_Sc_p` shows the level 0 muon cluster size that was estimated **without** muon
1324 tracking, and
- 1325 ● `MuonCMult_H_Tt_Sc_p` shows the level 0 muon cluster multiplicity that was estimated **without**
1326 muon tracking.

1327 In the histogram labels, t stands for the trolley number (1 or 3), c for the chamber slot label in
1328 trolley t and p for the partition label (A, B, C or D depending on the chamber layout) as explained
1329 in Chapter 5.3.

1330
1331 In the context of GIF++, an extra script called by the webDCS is called to extract the histograms
1332 from the ROOT files. The histograms are then stored in PNG and PDF formats into the correspond-
1333 ing folder (a single folder per HV step, so per ROOT file). the goal is to then display the histograms
1334 on the Data Quality Monitoring (DQM) page of the webDCS in order for the users to control the
1335 quality of the data taking at the end of data taking. An example of histogram organisation is given
1336 below:

1337



1338 *Here can put some screens from the webDCS to show the DQM and the plots available to users.*
 1339

1340 **B.2.1.2 CSV files**

1341 Moreover, up to 3 CSV files can be created depending on which ones of the 3 input files were in the
 1342 data folder:

- 1343 • `Offline-Corrupted.csv`, is used to keep track of the amount of data that was corrupted and
 1344 removed from old data format files that don't contain any data quality flag.
- 1345 • `Offline-Current.csv`, contains the summary of the currents and voltages applied on each
 1346 RPC HV channel.
- 1347 • `Offline-L0-EffC1.csv`, is used to write the efficiencies, cluster size and cluster multiplicity
 1348 of efficiency runs. Note that `L0` refers here to *Level 0* and means that the results of efficiency and
 1349 clusterization are a first approximation calculated without performing any muon tracking in

1350 between the different detectors. This offline tool provides the user with a preliminar calculation
 1351 of the efficiency and of the muon event parameters. Another analysis software especially
 1352 dedicated to muon tracking is called on selected data to retrieve the results of efficiency and
 1353 muon clusterization using a tracking algorithm to discriminate noise or gamma from muons
 1354 as muons are the only particles that pass through the full setup, leaving hits than can be used
 1355 to reconstruct their tracks.

- 1356 • `Offline-Rate.csv`, is used to write the noise or gamma rates measured in the detector readout
 1357 partitions.

1358 Note that these 4 CSV files are created along with their *headers* (`Offline-[...]-Header.csv`
 1359 containing the names of each data columns) and are automatically merged together when the offline
 1360 analysis tool is called from the webDCS, contrary to the case where the tool is runned locally from
 1361 the terminal as the merging bash script is then not called. Thus, the resulting files, used to make
 1362 official plots, are:

- 1363 • `Corrupted.csv`,
 1364 • `Current.csv`,
 1365 • `L0-EffCl.csv`.
 1366 • `Rate.csv`.

1367 B.3 Analysis inputs and information handling

1368 The usage of the Offline Analysis tool as well as its output have been presented in the previous section.
 1369 It is now important to dig further and start looking at the source code and the inputs necessary
 1370 for the tool to work. Indeed, other than the raw ROOT data files that are analysed, more information
 1371 needs to be imported inside of the program to perform the analysis such as the description of the
 1372 setup inside of GIFT++ at the time of data taking (number of trolleys, of RPCs, dimensions of the
 1373 detectors, etc...) or the mapping that links the TDC channels to the coresponding RPC channels in
 1374 order to translate the TDC information into human readable data. 2 files are used to transmit all this
 1375 information:
 1376

- 1377 • `Dimensions.ini`, that provides the necessary setup and RPC information, and
 1378 • `ChannelsMapping.csv`, that gives the link between the TDC and RPC channels as well as the
 1379 *mask* for each channel (masked or not?).

1380 B.3.1 Dimensions file and InFile parser

1381 This input file, present in every data folder, allows the analysis tool to know of the number of active
 1382 trolleys, the number of active RPCs in those trolleys, and the details about each RPCs such as
 1383 the number of RPC gaps, the number of pseudo-rapidity partitions (for CMS-like prototypes), the
 1384 number of strips per partion or the dimensions. To do so, there are 3 types of groups in the INI file
 1385 architecture. A first general group, appearing only once at the head of the document, gives information
 1386 about the number of active trolleys as well as their IDs, as presented in Source Code B.1. For

1387 each active trolley, a group similar to Source Code B.2 can be found containing information about
 1388 the number of active detectors in the trolley and their IDs. Each trolley group as a `Tt` name format,
 1389 where `t` is the trolley ID. Finally, for each detector stored in slots of an active trolley, there is a group
 1390 providing information about their names and dimensions, as showed in Source Code B.3. Each slot
 1391 group as a `TtSs` name format, where `s` is the slot ID of trolley `t` where the active RPC is hosted.

```
1392 [General]
1393 nTrolleys=2
  TrolleysID=13
```

1394 *Source Code B.1: Example of `[General]` group as might be found in `Dimensions.ini`. In Gif++, only 2 trolleys are available to hold RPCs and place them inside of the bunker for irradiation. The IDs of the trolleys are written in a single string as "13" and then read character by character by the program.*

```
1395 [T1]
  nSlots=4
  SlotsID=1234
```

1396 *Source Code B.2: Example of trolley group as might be found in `Dimensions.ini`. In this example, the file tells that there are 4 detectors placed in the holding slots of the trolley `T1` and that their IDs, written as a single string variable, are 1, 2, 3 and 4.*

```
1397 [T1S1]
  Name=RE2-2-NPD-BARC-8
  Partitions=3
  Gaps=3
  Gap1=BOT
  Gap2=TN
  Gap3=TW
  AreaGap1=11694.25
  AreaGap2=6432
  AreaGap3=4582.82
  Strips=32
  ActiveArea-A=157.8
  ActiveArea-B=121.69
  ActiveArea-C=93.03
```

1398 *Source Code B.3: Example of slot group as might be found in `Dimensions.ini`. In this example, the file provides information about a detector named `RE2-2-NPD-BARC-8`, having 3 pseudo-rapidity readout partitions and stored in slot `S1` of trolley `T1`. This is a CMS RE2-2 type of detector. This information will then be used for example to compute the rate per unit area calculation.*

1399 This information is readout and stored in a C++ object called `IniFile`, that parses the information
 1400 in the INI input file and stores it into a local buffer for later use. This INI parser is the exact same
 1401 one that was previously developed for the Gif++ DAQ and described in Appendix A.5.2.

1402 B.3.2 TDC to RPC link file and Mapping

1403 The same way the INI dimension file information is stored using `map`, the channel mapping and mask
 1404 information is stored and accessed through `map`. First of all, the mapping CSV file is organised into
 1405 3 columns separated by tabulations (and not by commas, as expected for CSV files as it is easier using
 1406 streams to read tab or space separated data using C++):

1407

1408 `RPC_channel` `TDC_channel` `mask`

1409 using as formatting for each field:

1410

<code>TSCCC</code>	<code>TCCC</code>	<code>M</code>
--------------------	-------------------	----------------

1411

1412 `TSCCC` is a 5-digit integer where `T` is the trolley ID, `s` the slot ID in which the RPC is held insite
 1413 the trolley `T` and `ccc` is the RPC channel number, or *strip* number, that can take values up to
 1414 3-digits depending on the detector,

1415 `TCCC` is a 4 digit integer where `T` is the TDC ID, `ccc` is the TDC channel number that can take values
 1416 in between 0 and 127, and

1417 `M` is a 1-digit integer indicating if the channel should be considered (`M = 1`) or discarded (`M = 0`)
 1418 during analysis.

1419 This mapping and masking information is readout and stored thanks to the object `Mapping`, pre-
 1420 sented in Source Code B.4. Similarly to `IniFile` objects, this class has private methods. The first
 1421 one, `Mapping::CheckIfNewLine()` is used to find the newline character '`\n`' or return character
 1422 '`\r`' (depending on which kind of operating system interacted with the file). This is used for the
 1423 simple reason that the masking information has been introduced only during the year 2017 but the
 1424 channel mapping files exist since 2015 and the very beginning of data taking at GIF++. This means
 1425 that in the older data folders, before the upgrade, the channel mapping file only had 2 columns, the
 1426 RPC channel and the TDC channel. For compatibility reasons, this method helps controling the
 1427 character following the readout of the 2 first fields of a line. In case any end of line character is
 1428 found, no mask information is present in the file and the default `M = 1` is used. On the contrary, if
 1429 the next character was a tabulation or a space, the mask information is present.

1430 Once the 3 fields have been readout, the second private method `Mapping::CheckIfTDCCh()` is
 1431 used to control that the TDC channel is an existing TDC channel. Finally, the information is stored
 1432 into 3 different maps (`Link`, `ReverseLink` and `Mask`) thanks to the public method `Mapping::Read()`.
 1433 `Link` allows to get the RPC channel by knowing the TDC channel while `ReverseLink` does the op-
 1434 posite by returning the TDC channel by knowing the RPC channel. Finally, `Mask` returns the mask
 1435 associated to a given RPC channel.

```

1436 typedef map<UInt, UInt> MappingData;

1437 class Mapping {
1438     private:
1439         bool          CheckIfNewLine(char next);
1440         bool          CheckIfTDCCh(UInt channel);
1441         string        FileName;
1442         MappingData  Link;
1443         MappingData  ReverseLink;
1444         MappingData  Mask;
1445         int           Error;
1446
1447     public:
1448         Mapping();
1449         Mapping(string baseName);
1450         ~Mapping();
1451
1452         void SetFileName(const string filename);
1453         int Read();
1454         UInt GetLink(UInt tdcchannel);
1455         UInt GetReverse(UInt rpcchannel);
1456         UInt GetMask(UInt rpcchannel);
1457     };

```

1438 *Source Code B.4: Description of C++ object Mapping used as a parser for the channel mapping and mask file.*

1439 B.4 Description of GIF++ setup within the Offline Analysis tool

1440 In the previous section, the tool input files have been discussed. The dimension file information is
 1441 stored in a map hosted by the `IniFile` object. But this information is then used to create a series of
 1442 new objects that helps defining the GIF++ infrastructure directly into the Offline Analysis. Indeed,
 1443 from the `RPC`, to the more general `Infrastructure`, every element of the GIF++ infrastrucutre is
 1444 recreated for each data analysis based on the information provided in input. All this information
 1445 about the infrastructure will be used to assign each hit signal to a specific strip channel of a specific
 1446 detector, and having a specific active area. This way, rate per unit area calculation is possible.
 1447

1448 B.4.1 RPC objects

1449 `RPC` objects have been developped to represent physical active detectors in GIF++ at the moment
 1450 of data taking. Thus, there are as many `RPC` objects created during the analysis than there were
 1451 active `RPCs` tested during a run. Each `RPC` hosts the information present in the corresponding INI
 1452 slot group, as shown in B.3, and organises it using a similar architecture. This can be seen from
 1453 Source B.5.

1454 To make the object more compact, the lists of gap labels, of gap active areas and strip active
 1455 areas are stored into `vector` dynamical containers. `RPC` objects are always contructed thanks to the
 1456 dimension file information stored into the `IniFILE` and their ID, using the format `TtSs`. Using the
 1457 `RPC` ID, the constructor calls the methods of `IniFILE` to initialise the `RPC`. The other constructors
 1458 are not used but exist in case of need. Finally, some getters have been written to access the different
 1459 private parameters storing the detector information.

```

1460
class RPC{
    private:
        string      name;           //RPC name as in webDCS database
        Uint        nGaps;          //Number of gaps in the RPC
        Uint        nPartitions;    //Number of partitions in the RPC
        Uint        nStrips;         //Number of strips per partition
        vector<string> gaps;       //List of gap labels (BOT, TOP, etc...)
        vector<float>  gapGeo;        //List of gap active areas
        vector<float>  stripGeo;      //List of strip active areas

    public:
        RPC();
        RPC(string ID, IniFile* geofile);
        RPC(const RPC& other);
        ~RPC();
        RPC& operator=(const RPC& other);

        string GetName();
        Uint GetNGaps();
        Uint GetNPartitions();
        Uint GetNStrips();
        string GetGap(Uint g);
        float GetGapGeo(Uint g);
        float GetStripGeo(Uint p);
};

1461
1462  Source Code B.5: Description of C++ objects RPC that describe each active detectors used during data taking.

```

1463 B.4.2 Trolley objects

1464 Trolley objects have been developped to represent physical active trolleys in GIFT++ at the moment
 1465 of data taking. Thus, there are as many trolley objects created during the analysis than there were
 1466 active trolleys hosting tested RPCs during a run. Each Trolley hosts the information present in the
 1467 corresponding INI trolley group, as shown in B.2, and organises it using a similar architecture. In
 1468 addition to the information hosted in the INI file, these object have a dynamical container of RPC
 1469 objects, representing the active detectors the active trolley was hosting at the time of data taking.
 1470 This can been seen from Source Code B.6.

1471 Trolley objects are always contructed thanks to the dimension file information stored into the
 1472 IniFILE and their ID, using the format Tt. Using the Trolley ID, the constructor calls the methods
 1473 of IniFILE to initialise the Trolley. Retrieving the information of the RPC IDs via SlotsID, a new
 1474 RPC is constructed and added to the container RPCs for each character in the ID string. The other
 1475 constructors are not used but exist in case of need. Finally, some getters have been written to access
 1476 the different private parameters storing the trolley and detectors information.

```

1477
class Trolley{
    private:
        Uint          nSlots; //Number of active RPCs in the considered trolley
        string        SlotsID; //Active RPC IDs written into a string
        vector<RPC*> RPCs;   //List of active RPCs

    public:
        //Constructors, destructor and operator =
        Trolley();
        Trolley(string ID, IniFile* geofile);
        Trolley(const Trolley& other);
        ~Trolley();
        Trolley& operator=(const Trolley& other);

        //Get GIFTrolley members
        Uint  GetNSlots();
        string GetSlotsID();
        Uint   GetSlotID(Uint s);

        //Manage RPC list
        RPC*  GetRPC(Uint r);
        void  DeleteRPC(Uint r);

        //Methods to get members of RPC objects stored in RPCs
        string GetName(Uint r);
        Uint   GetNGaps(Uint r);
        Uint   GetNPartitions(Uint r);
        Uint   GetNStrips(Uint r);
        string GetGap(Uint r, Uint g);
        float  GetGapGeo(Uint r, Uint g);
        float  GetStripGeo(Uint r, Uint p);
    };

```

Source Code B.6: Description of C++ objects `Trolley` that describe each active trolley used during data taking.

1480 B.4.3 Infrastructure object

1481 The `Infrastructure` object has been developped to represent the GIFT++ bunker area dedicated to
 1482 CMS RPC experiments. With this very specific object, all the information about the CMS RPC
 1483 setup within GIFT++ at the moment of data taking is stored. It hosts the information present in the
 1484 corresponding INI general group, as shown in B.1, and organises it using a similar architecture. In
 1485 addition to the information hosted in the INI file, this object have a dynamical container of `Trolley`
 1486 objects, representing the active tolleys in GIFT++ area. This can be seen from Source Code B.7.

1487 The `Infrastructure` object is always contructed thanks to the dimension file information stored
 1488 into the `IniFILE`. Retrieving the information of the trolley IDs via `TrolleysID`, a new `Trolley` is
 1489 constructed and added to the container `Trolleys` for each character in the ID `string`. By extension,
 1490 it is easy to understand that the process described in Section B.4.2 for the construction of RPCs
 1491 takes place when a trolley is constructed. The other constructors are not used but exist in case of
 1492 need. Finally, some getters have been written to access the different private parameters storing the
 1493 infrastructure, tolleys and detectors information.

```

1494 class Infrastructure {
1495     private:
1496         Uint             nTrolleys; //Number of active Trolleys in the run
1497         string          TrolleysID; //Active trolley IDs written into a string
1498         vector<Trolley*> Trolleys; //List of active Trolleys (struct)
1499
1500     public:
1501         //Constructors and destructor
1502         Infrastructure();
1503         Infrastructure(IniFile* geofile);
1504         Infrastructure(const Infrastructure& other);
1505         ~Infrastructure();
1506         Infrastructure& operator=(const Infrastructure& other);
1507
1508         //Get Infrastructure members
1509         Uint GetNTrolleys();
1510         string GetTrolleysID();
1511         Uint GetTrolleyID(Uint t);
1512
1513         //Manage Trolleys
1514         Trolley* GetTrolley(Uint t);
1515         void DeleteTrolley(Uint t);
1516
1517         //Methods to get members of GIFTrolley objects stored in Trolleys
1518         Uint GetNSlots(Uint t);
1519         string GetSlotsID(Uint t);
1520         Uint GetSlotID(Uint t, Uint s);
1521         RPC* GetRPC(Uint t, Uint r);
1522
1523         //Methods to get members of RPC objects stored in RPCs
1524         string GetName(Uint t, Uint r);
1525         Uint GetNGaps(Uint t, Uint r);
1526         Uint GetNPartitions(Uint t, Uint r);
1527         Uint GetNStrips(Uint t, Uint r);
1528         string GetGap(Uint t, Uint r, Uint g);
1529         float GetGapGeo(Uint t, Uint r, Uint g);
1530         float GetStripGeo(Uint t, Uint r, Uint p);
1531     };

```

Source Code B.7: Description of C++ object Infrastructure that contains the full information about CMS RPC experiment in GIF++.

1497 B.5 Handeling of data

1498 As discussed in Appendix A.4.2, the raw data as a `TTree` architecture where every entry is related to
1499 a trigger signal provided by a muon or a random pulse, whether the goal of the data taking was to
1500 measure the performance of the detector or the noise/gamma background respectively. Each of these
1501 entries, referred also as events, contain a more or less full list of hits in the TDC channels to which
1502 the detectors are connected. To this list of hits corresponds a list of time stamps, marking the arrival
1503 of the hits within the TDC channel.

1504 The infrastructure of the CMS RPC experiment within `GIF++` being defined, combining the
1505 information about the raw data with the information provided by both the mapping/mask file and the
1506 dimension file allows to build new physical objects that will help in computing efficiency or rates.

1507 B.5.1 RPC hits

1508 The raw data stored in the ROOT file as output of the GIFT++ DAQ, is readout by the analysis tool
 1509 using the structure `RAWData` presented in Source Code B.9 that differs from the structure presented
 1510 in Appendix A.4.2 as it is not meant to hold all of the data contained in the ROOT file. In this sense,
 1511 this structure is in the case of the offline analysis tool not a dynamical object and will only be storing
 1512 a single event contained in a single entry of the `TTree`.

```
1513
1514 class RPCHit {
1515     private:
1516         Uint Channel;      //RPC channel according to mapping (5 digits)
1517         Uint Trolley;      //0, 1 or 3 (1st digit of the RPC channel)
1518         Uint Station;      //Slot where is held the RPC in Trolley (2nd digit)
1519         Uint Strip;        //Physical RPC strip where the hit occurred (last 3
1520         →   digits)
1521         Uint Partition;    //Readout partition along eta segmentation
1522         float TimeStamp;   //Time stamp of the arrival in TDC
1523
1524     public:
1525         //Constructors, destructor & operator =
1526         RPCHit();
1527         RPCHit(Uint channel, float time, Infrastructure* Infra);
1528         RPCHit(const RPCHit& other);
1529         ~RPCHit();
1530         RPCHit& operator=(const RPCHit& other);
1531
1532         //Get RPCHit members
1533         Uint GetChannel();
1534         Uint GetTrolley();
1535         Uint GetStation();
1536         Uint GetStrip();
1537         Uint GetPartition();
1538         float GetTime();
1539     };
1540
1541     typedef vector<RPCHit> HitList;
1542     typedef struct GIFHitList { HitList rpc[NTROLLEYS][NSLOTS][NPARTITIONS]; } →
1543     →   GIFHitList;
1544
1545     bool SortHitbyStrip(RPCHit h1, RPCHit h2);
1546     bool SortHitbyTime(RPCHit h1, RPCHit h2);

```

1515 *Source Code B.8: Description of C++ object RPCHit.*

```
1516
1517     struct RAWData{
1518         int iEvent;          //Event i
1519         int TDCNHits;       //Number of hits in event i
1520         int QFlag;           //Quality flag list (1 flag digit per TDC)
1521         vector<Uint> *TDCCh; //List of channels giving hits per event
1522         vector<float> *TDCTS; //List of the corresponding time stamps
1523     };

```

1517 *Source Code B.9: Description of C++ structure RAWData.*

1518 Each member of the structure is then linked to the corresponding branch of the ROOT data tree,
 1519 as shown in the example of Source Code B.10, and using the method `GetEntry(int i)` of the ROOT
 1520 class `TTree` will update the state of the members of `RAWData`.

```

1521   TTree* dataTree = (TTree*)dataFile.Get("RAWData");
1522   RAWData data;
1523
1524   dataTree->SetBranchAddress("EventNumber", &data.iEvent);
1525   dataTree->SetBranchAddress("number_of_hits", &data.TDCNHits);
1526   dataTree->SetBranchAddress("Quality_flag", &data.QFlag);
1527   dataTree->SetBranchAddress("TDC_channel", &data.TDCCh);
1528   dataTree->SetBranchAddress("TDC_TimeStamp", &data.TDCTS);

```

1523 *Source Code B.10: Example of link in between RAWData and TTree.*

1524 The data is then analysed entry by entry and to each element of the TDC channel list, a `RPCHit` is
 1525 constructed by linking each TDC channel to the corresponding RPC channel thanks to the `Mapping`
 1526 object. The information carried by the RPC channel format allows to easily retrieve the trolley and
 1527 slot from which the hit was recorded (see section B.3.2). Using these 2 values, the readout partition
 1528 can be found by knowing the strip channel and comparing it with the number of partitions and strips
 1529 per partition stored into the `Infrastructure` object.

1530 Thus `RPCHit` objects are then stored into 3D dynamical list called `GIFHitList` (Source Code B.9)
 1531 where the 3 dimensions refer to the 3 layers of the readout in `GIF++` : in the bunker there are *trolleys*
 1532 (τ) holding detectors in *slots* (s) and each detector readout is divided into 1 or more pseudo-rapidity
 1533 *partitions* (p). Using these 3 information allows to assign an address to each readout partition and
 1534 this address will point to a specific hit list.

1535

1536 B.5.2 Clusters of hits

1537 All the hits contained in the ROOT file have been sorted into the different hit lists through the
 1538 `GIFHitList`. At this point, it is possible to start looking for clusters. A cluster is a group of adjacent
 1539 strips getting hits within a time window of 25 ns. These strips are then assumed to be part of the same
 1540 physical avalanche signal generated by a muon passing through the chamber or by the interaction of
 1541 a gamma stopping into the electrodes of the RPCs.

1542 To keep the cluster information, `RPCCluster` objects have been defined as shown in Source
 1543 Code B.11. Using the information of each individual `RPCHit` taken out of the hit list, it stores
 1544 the cluster size (number of adjacent strips composing the cluster), the first and last hit, the center for
 1545 spatial reconstruction and finally the start and stop time stamps as well as the time spread in between
 1546 the first and last hit.

```

1547
class RPCCluster{
    private:
        Uint ClusterSize; //Size of cluster #ID
        Uint FirstStrip; //First strip of cluster #ID
        Uint LastStrip; //Last strip of cluster #ID
        float Center; //Center of cluster #ID ((first+last)/2)
        float StartStamp; //Time stamp of the earliest hit of cluster #ID
        float StopStamp; //Time stamp of the latest hit of cluster #ID
        float TimeSpread; //Time difference between earliest and latest hits
                           //of cluster #ID

    public:
        //Constructors, destructor & operator =
        RPCCluster();
        RPCCluster(HitList List, Uint cID, Uint cSize, Uint first, Uint firstID);
        RPCCluster(const RPCCluster& other);
        ~RPCCluster();
        RPCCluster& operator=(const RPCCluster& other);

        //Get Cluster members
        Uint GetID();
        Uint GetSize();
        Uint GetFirstStrip();
        Uint GetLastStrip();
        float GetCenter();
        float GetStart();
        float GetStop();
        float GetSpread();
};

typedef vector<RPCCluster> ClusterList;

//Other functions to build cluster lists out of hit lists
void BuildClusters(HitList &cluster, ClusterList &clusterList);
void Clusterization(HitList &hits, TH1 *hcSize, TH1 *hcMult);

```

Source Code B.11: Description of C++ object cluster.

To investigate the hit list of a given detector partition, the function `Clusterization()` defined in `include/Cluster.h` needs the hits in the list to be time sorted. This is achieved by calling function `sort()` of library `<algorithm>` using the comparator `SortHitbyTime(RPCHit h1, RPCHit h2)` defined in `include/RPCHit.h` that returns `true` if the time stamp of hit `h1` is lower than that of `h2`. A first isolation of strips is made only based on time information. All the hits within the 25 ns window are taken separately from the rest. Then, this sub-list of hits is sorted this time by ascending strip number, using this time the comparator `SortHitbyStrip(RPCHit h1, RPCHit h2)`. Finally, the groups of adjacent strips are used to construct `RPCCluster` objects that are then stored in a temporary list of clusters that is at the end of the process used to know how many clusters were reconstructed and to fill their sizes into an histogram that will allows to know the mean size of muon or gamma clusters.

1562 B.6 DAQ data Analysis

All the ingredients to analyse GIF++ data have been defined. This section will focus on the different part of the analysis performed on the data, from determining the type of data the tool is dealing with

1565 to calculating the rate in each detector or reconstructing muon or gamma clusters.

1566 B.6.1 Determination of the run type

1567 In GIF++, both the performance of the detectors in detecting muons in an irradiated environment and
 1568 the gamma background can be independantly measured. These corresponds to different run types
 1569 and thus, to different TDC settings giving different data to look at.

1570

1571 In the case of performance measurements, the trigger for data taking is provided by the coïncidence
 1572 of several scintillators when muons from the beam passing through the area are detected. Data
 1573 is collected in a 600 ns wide window around the arrival of muons in the RPCs. The expected time
 1574 distribution of hits is shown in Figure B.1a. The muon peak is clearly visible in the center of the
 1575 distribution and is to be extracted from the gamma background that composes the flat part of the
 1576 distribution.

1577 On the other hand, gamma background or noise measurements are focussed on the non muon
 1578 related physics and the trigger needs to be independant from the muons to give a good measurement
 1579 of the gamma/noise distribution as seen by the detectors. The trigger is then provided by a pulse
 1580 generator at a frequency of 300 Hz whose pulse is not likely to be on time with a muon. In order
 1581 to increase the integrated time without increasing the acquisition time too much, the width of the
 1582 acquisition windows are increased to 10 μ s. The time distribution of the hits is expected to be flat, as
 1583 shown by Figure B.1b.

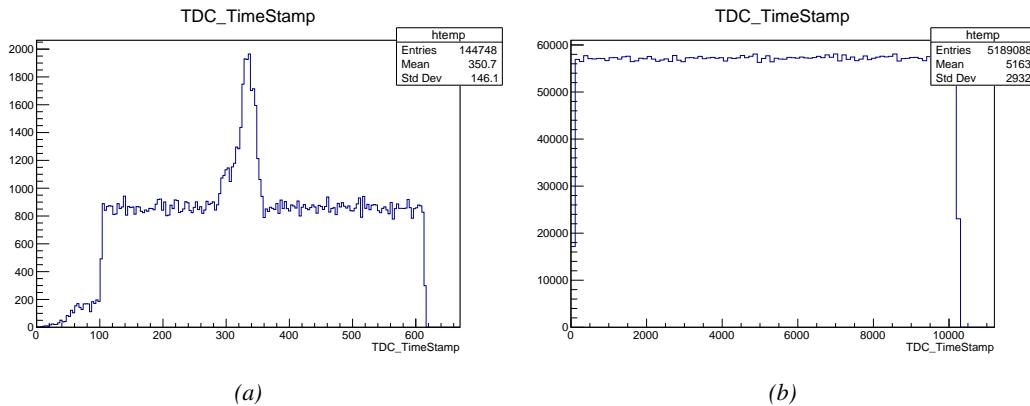


Figure B.1: Example of expected hit time distributions in the cases of efficiency (Figure B.1a) and noise/gamma rate per unit area (Figure B.1b) measurements as extracted from the raw ROOT files. The unit along the x-axis corresponds to ns. The fact that "the" muon peak is not well defined in Figure B.1a is due to the contribution of all the RPCs being tested at the same time that don't necessarily have the same signal arrival time. Each individual peak can have an offset with the ones of other detectors. The inconsistency in the first 100 ns of both time distributions is an artefact of the TDCs and are systematically rejected during the analysis.

1584 The ROOT files include a TTree called RunParameters containing, among other things, the in-
 1585 formation related to the type of run. The run type can then be accessed as described by Source
 1586 Code B.12 and the function IsEfficiencyRun() is then used to determine if the run file is an effi-
 1587 ciency run or, on the contrary, another type of run (noise or gamma measurement).

```

1588
1589     TTree* RunParameters = (TTree*)dataFile.Get("RunParameters");
1590     TString* RunType = new TString();
1591     RunParameters->SetBranchAddress("RunType", &RunType);
1592     RunParameters->GetEntry(0);

```

1590 *Source Code B.12: Access to the run type contained in TTree* RunParameters.*

1591 Finally, the data files will have a slightly different content whether it was collected before or after
1592 October 2017 and the upgrade of the DAQ software that brought a new information into the ROOT
1593 output. This is discussed in Appendix A.4.3 and implies that the analysis will differ a little depending
1594 on the data format. Indeed, as no information on the data quality is stored, in older data files, the cor-
1595 rections for missing events has to be done at the end of the analysis. The information about the type
1596 of data format is stored in the variable **bool** `isNewFormat` by checking the list of branches contained
1597 in the data tree via the methods `TTree::GetListOfBranches()` and `TCollection::Contains()`.

1598 **B.6.2 Beam time window calculation for efficiency runs**

1599 Knowing the run type is important first of all to know the width of the acquisition window to be used
1600 for the rate calculation and finally to be able to seek for muons. Indeed, the peak that appears in the
1601 time distribution for each detectors is then fitted to extract the most probable time window in which
1602 the tool should look for muon hits. The data outside of this time window is then used to evaluate the
1603 noise or gamma background the detector was subjected to during the data taking. Computing the
1604 position of the peak is done calling the function `SetBeamWindow()` defined in file `src/RPCHit.cc` that
1605 loops a first time on the data. The data is first sorted in a 3D array of 1D histograms (`GIFH1Array`, see
1606 `include/types.h`). Then the location of the highest bin is determined using `TH1::GetMaximumBin()`
1607 and is used to define a window in which a gaussian fit will be applied to compute the peak width.
1608 This window is a 80 ns defined by Formula B.1 around the central bin.

$$t_{center}(ns) = bin \times width_{bin}(ns) \quad (\text{B.1a})$$

$$[t_{low}; t_{high}] = [t_{center} - 40; t_{center} + 40] \quad (\text{B.1b})$$

1609 Before the fit is performed, the average number of noise/gamma hits per bin is evaluated using
1610 the data outside of the fit window. Excluding the first 100 ns, the average number of hits per bin
1611 due to the noise or gamma is defined by Formula B.2 after extracting the amount of hits in the time
1612 windows $[100; t_{low}]$ and $[t_{high}; 600]$ thanks to the method `TH1::Integral()`. This average number
1613 of hits is then subtracted to every bin of the 1D histogram, in order to *clean* it from the noise or
1614 gamma contribution as much as possible to improve the fit quality. Bins where $\langle n_{hits} \rangle$ is greater
1615 than the actual bin content are set to 0.

$$\Delta t_{noise}(ns) = 600 \overbrace{-t_{high} + t_{low}}^{-80ns} - 100 = 420ns \quad (\text{B.2a})$$

$$\langle n_{hits} \rangle = width_{bin}(ns) \times \frac{\sum_{t=100}^{t_{low}} + \sum_{t=t_{high}}^{600}}{\Delta t_{noise}(ns)} \quad (\text{B.2b})$$

1616 Finally, the fit parameters are extracted and saved for each detector in 3D arrays of **float**
1617 (`muonPeak`, see `include/types.h`), a first one for the mean arrival time of the muons, `PeakTime`,

1618 and a second one for the width of the peak, `PeakWidth`. The width is defined as 6σ of the gaussian
 1619 fit. The same settings are applied to every partitions of the same detector. To determine which one
 1620 of the detector's partitions is directly illuminated by the beam, the peak height of each partition is
 1621 compared and the highest one is then used to define the peak settings.

1622 B.6.3 Data loop and histogram filling

1623 3D arrays of histogram are created to store the data and display it on the DQM of G4F++ webDCS
 1624 for the use of shifters. These histograms, presented in section B.2.1.1, are filled while looping on
 1625 the data. Before starting the analysis loop, it is necessary to control the entry quality for the new
 1626 file formats featuring `QFlag`. If the `QFlag` value for this entry shows that 1 TDC or more have a
 1627 CORRUPTED flag, then this event is discarded. The loss of statistics is low enough to be neglected.
 1628 `QFlag` is controlled using the function `IsCorruptedEvent()` defined in `src/utils.cc`. As explained
 1629 in Appendix A.4.3, each digit of this integer represent a TDC flag that can be 1 or 2. Each 2 is
 1630 the sign of a CORRUPTED state. Then, the data is accessed entry by entry in the ROOT `TTree` using
 1631 `RAWData` and each hit in the hit list is assigned to a detector channel and saved in the corresponding
 1632 histograms. In the first part of the analysis, in which the loop over the ROOT file's content is
 1633 performed, the different steps are:

1634 **1- RPC channel assignment and control:** a check is done on the RPC channel extracted thanks
 1635 to the mapping via the method `Mapping::GetLink()`. If the channel is not initialised and is 0, or if
 1636 the TDC channel was greater than 5127, the hit is discarded. This means there was a problem in the
 1637 mapping. Often a mapping problem leads to the crash of the offline tool.

1638 **2- Creation of a `RPCHit` object:** to easily get the trolley, slot and partition in which the hit has
 1639 been assigned, this object is particularly helpful.

1640 **3- General histograms are filled:** the hit is filled into the time distribution and the general hit
 1641 distribution histograms, and if the arrival time is within the first 100 ns, it is discarded and nothing
 1642 else happens and the loop proceeds with the next hit in the list.

1643 **4- Multiplicity counter:** the hit multiplicity counter of the corresponding detectors incremented.

1644 **5-a- Efficiency runs - Is the hit within the peak window? :** if the peak is contained in the peak
 1645 window previously defined in section B.6.2, the hit is filled into the beam hit profile histogram of
 1646 the corresponding chamber, added into the list of muon hits and increments the counter of *in time*
 1647 hits. The term *in time* here refers to the hits that are likely to be muons by arriving in the expected
 1648 time window. If the hit is outside of the peak window, it is filled into the noise profile histogram
 1649 of the corresponding detector, added into the list of noise/gamma hits and increments the counter of
 1650 noise/gamma hits.

1651 **5-b- Noise/gamma rate runs - Noise histograms are filled:** the hit is filled into the noise profile
 1652 histogram of the corresponding detector, added into the list of noise/gamma hits and increments the
 1653 counter of noise/gamma hits.

1655 After the loop on the hit list of the entry is over, the next step is to clusterize the 3D lists filled
 1656 in the previous steps. A 3D loop is then started over the active trolley, slot and RPC partitions to
 1657 access these objects. Each `NoiseHitList` and `MuonHitList`, in case of efficiency run, are clusterized
 1658 as described in section B.5.2. There corresponding cluster size and multiplicity histograms are filled
 1659 at the end of the clustering process. Then, the efficiency histogram is filled in case of efficiency run.
 1660 The selection is simply made by checking whether the RPC detected signals in the peak window
 1661 during this event. Nevertheless, it is useful to highlight that at this level, it is not possible yet to
 1662 discriminate in between a muon hit and noise or gamma hit. Thus, `MuonCSize_H`, `MuonCMult_H`
 1663 and `Efficiency0_H` are subjected to noise and gamma contamination. This contamination will be
 1664 estimated and corrected at the moment the results will be written into output CSV files. Finally, the
 1665 loop ends on the filling of the general hit multiplicity histogram.

1666 **B.6.4 Results calculation**

1667 As mentioned in section B.2.1, the analysis of DAQ data provides the user with 3 CSV files and
 1668 a ROOT file associated to each and every ROOT data file. The fourth CSV file is provided by the
 1669 extraction of the CEAN main frame data monitored during data taking and will be discussed later.
 1670 After looping on the data in the previous part of the analysis macro, the output files are created and a
 1671 3D loop on each RPC readout partitions is started to extract the histograms parameters and compute
 1672 the final results.

1673

1674 **B.6.4.1 Rate normalisation**

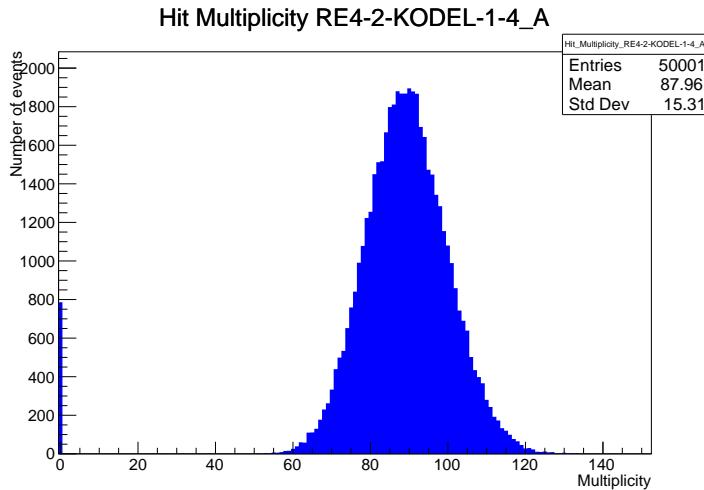


Figure B.2: The effect of the quality flag is explained by presenting the reconstructed hit multiplicity of a data file without `Quality_flag`. The artificial high content of bin 0 is the effect of corrupted data.

1675 To analyse old data format files, not containing any quality flag, it is needed to estimate the amount
 1676 of corrupted data via a fit as the corrupted data will always fill events with a fake "0 multiplicity".
 1677 Indeed, as no hits were stored in the DAQ ROOT files, these events artificially contribute to fill
 1678 the bin corresponding to a null multiplicity, as shown in Figure B.2. In the case the mean of the

hit multiplicity distribution is high, the contribution of the corrupted data can easily be evaluated for later correction by comparing the level of the bin at multiplicity 0 and of a skew fit curve that should indicate a value consistent with 0. A skew fit has been chosen over a Poisson fit as it was giving better results for lower mean multiplicity values. Nevertheless, for low irradiation cases, as explained in Appendix A.4.3, the hit multiplicity distribution mean is, on the contrary, rather small and the probability to record events without hits can't be considered small anymore, leading to a difficult and non-reliable estimation of the corruption. As can be seen in Source Code B.13, conditions have been applied to prevent bad fits and wrong corruption estimation in cases where :

- The difference in between the data for multiplicity 1 and the corresponding fit value should be lower than 1% of the total amount of data : $\frac{|n_{m=1} - sk(1)|}{N_{tot}} < 0.01$ where $n_{m=1}$ is the number of entries with multiplicity 1, $sk(1)$ the value of the skew fit, as defined by Formula 5.3, for multiplicity 1 and N_{tot} the total number of entries.

- The amount of data contained in the multiplicity 0 bin should not exceed 40% : $\frac{n_{m=0}}{N_{tot}} \leq 0.4$ where $n_{m=0}$ is the number of entries with multiplicity 0. This number has been determined to be the maximum to be able to separate the excess of data due to corruption from the hit multiplicity distribution.

Those 2 conditions need to be fulfilled to estimate the corruption of old data format files. If the fit was successful, the level of corruption is written in `Offline-Corrupted.csv` and the number of corrupted entries, refered as the integer `nEmptyEvent`, is subtracted from the total number of entries when the rate normalisation factor is computed as explicit in Source Code B.13. Note that for new data format files, the number of corrupted entries being set to 0, the definition of `rate_norm` stays valid.

```

1701
if(!isNewFormat){
    TF1* GaussFit = new TF1("gaussfit","[0]*exp(-0.5*((x-[1])/[2])**2)",0,Xmax);
    GaussFit->SetParameter(0,100);
    GaussFit->SetParameter(1,10);
    GaussFit->SetParameter(2,1);
    HitMultiplicity_H.rpc[T][S][p]->Fit(GaussFit,"LIQR","");
    HitMultiplicity_H.rpc[T][S][p]->Fit(GaussFit,"LIQR","");
}

TF1* SkewFit = new TF1("skewfit","[0]*exp(-0.5*((x-[1])/[2])**2) / (1 +
→ exp(-[3]*(x-[4])))",0,Xmax);
SkewFit->SetParameter(0,GaussFit->GetParameter(0));
SkewFit->SetParameter(1,GaussFit->GetParameter(1));
SkewFit->SetParameter(2,GaussFit->GetParameter(2));
SkewFit->SetParameter(3,1);
SkewFit->SetParameter(4,1);
HitMultiplicity_H.rpc[T][S][p]->Fit(SkewFit,"LIQR","");
HitMultiplicity_H.rpc[T][S][p]->Fit(SkewFit,"LIQR","");
double fitValue = SkewFit->Eval(1,0,0,0);
double dataValue = (double)HitMultiplicity_H.rpc[T][S][p]->GetBinContent(2);
double difference = TMath::Abs(dataValue - fitValue);
double fitTOdataVSentries_ratio = difference / (double)nEntries;
bool isFitGOOD = fitTOdataVSentries_ratio < 0.01;

1702
double nSinglehit = (double)HitMultiplicity_H.rpc[T][S][p]->GetBinContent(1);
double lowMultRatio = nSinglehit / (double)nEntries;
bool isMultLOW = lowMultRatio > 0.4;

if(isFitGOOD && !isMultLOW){
    nEmptyEvent = HitMultiplicity_H.rpc[T][S][p]->GetBinContent(1);
    nPhysics = (int)SkewFit->Eval(0,0,0,0);
    if(nPhysics < nEmptyEvent)
        nEmptyEvent = nEmptyEvent-nPhysics;
}
}

double corrupt_ratio = 100.*(double)nEmptyEvent / (double)nEntries;
outputCorrCSV << corrupt_ratio << '\t';

float rate_norm = 0.;
float stripArea = GIFIInfra->GetStripGeo(tr,sl,p);

if(IsEfficiencyRun(RunType)){
    float noiseWindow = BMTDCWINDOW - TIMEREJECT - 2*PeakWidth.rpc[T][S][p];
    rate_norm = (nEntries-nEmptyEvent)*noiseWindow*1e-9*stripArea;
} else
    rate_norm = (nEntries-nEmptyEvent)*RDMNOISEWDW*1e-9*stripArea;

```

Source Code B.13: Definition of the rate normalisation variable. It takes into account the number of non corrupted entries and the time window used for noise calculation, to estimate the total integrated time, and the strip active area to express the result as rate per unit area.

1704 B.6.4.2 Rate and activity

1705 At this point, the strip rate histograms, StripNoiseProfile_H.rpc[T][S][p], only contain an in-
 1706 formation about the total number of noise or rate hits each channel received during the data taking.
 1707 As described in Source Code B.14, a loop on the strip channels will be used to normalise the content
 1708 of the rate distribution histogram for each detector partitions. The initial number of hits recorded for
 1709 a given bin will be extracted and 2 values will be computed:

- 1710 ● the strip rate, defined as the number of hits recorded in the bin normalised like described in
 1711 the previous section, using the variable `rate_norm`, and

- 1712 ● the strip activity, defined as the number of hits recorded in the bin normalised to the average
 1713 number of hits per bin contained in the partition histogram, using the variable `averageNhit`.
 1714 This value provides an information on the homogeneity of the detector response to the gamma
 1715 background or of the detector noise. An activity of 1 corresponds to an average response.
 1716 Above 1, the channel is more active than the average and bellow 1, the channel is less active.

```

int nNoise = StripNoiseProfile_H.rpc[T][S][p]->GetEntries();
float averageNhit = (nNoise>0) ? (float)(nNoise/nStripsPart) : 1.;

for(Uint st = 1; st <= nStripsPart; st++) {
    float stripRate =
        StripNoiseProfile_H.rpc[T][S][p]->GetBinContent(st)/rate_norm;
    float stripAct =
        StripNoiseProfile_H.rpc[T][S][p]->GetBinContent(st)/averageNhit;

    StripNoiseProfile_H.rpc[T][S][p]->SetBinContent(st,stripRate);
    StripActivity_H.rpc[T][S][p]->SetBinContent(st,stripAct);
}

```

1718 *Source Code B.14: Description of the loop that allows to set the content of each strip rate and strip activity
 channel for each detector partition.*

1719 On each detector partitions, which are readout by a single FEE, all the channels are not processed
 1720 by the same chip. Each chip can give a different noise response and thus, histograms using a chip
 1721 binning are used to investigate chip related noise behaviours. The average values of the strip rate
 1722 or activity grouped into a given chip are extracted using the using the function `GetChipBin()` and
 1723 stored in dedicated histograms as described in Source Codes B.15 and B.16 respectively.

```

float GetChipBin(TH1* H, Uint chip){
    Uint start = 1 + chip*NSTRIPSCHIP;
    int nActive = NSTRIPSCHIP;
    float mean = 0.;

    for(Uint b = start; b <= (chip+1)*NSTRIPSCHIP; b++) {
        float value = H->GetBinContent(b);
        mean += value;
        if(value == 0.) nActive--;
    }

    if(nActive != 0) mean /= (float)nActive;
    else mean = 0.;

    return mean;
}

```

1726 *Source Code B.15: Function used to compute the content of a bin for an histogram using chip binning.*

```

1727   for(UInt ch = 0; ch < (nStripsPart/NSTRIPSCHIP); ch++) {
1728     ChipMeanNoiseProf_H.rpc[T][S][p]->
1729       SetBinContent(ch+1,GetChipBin(StripNoiseProfile_H.rpc[T][S][p],ch));
1730     ChipActivity_H.rpc[T][S][p]->
1731       SetBinContent(ch+1,GetChipBin(StripActivity_H.rpc[T][S][p],ch));
1732   }

```

Source Code B.16: Description of the loop that allows to set the content of each chip rate and chip activity bins for each detector partition knowing the information contained in the corresponding strip distribution histograms.

The activity variable is used to evaluate the homogeneity of the detector response to background or of the detector noise. The homogeneity h_p of each detector partition can be evaluated using the formula $h_p = \exp(-\sigma_p^R / \langle R \rangle_p)$, where $\langle R \rangle_p$ is the partition mean rate and σ_p^R is the rate standard deviation calculated over the partition channels. The more homogeneously the rates are distributed and the smaller will σ_p^R be, and the closer to 1 will h_p get. On the contrary, if the standard deviation of the channel's rates is large, h_p will rapidly get to 0. This value is saved into histograms as showed in Source Code B.17 and could in the future be used to monitor through time, once extracted, the evolution of every partition homogeneity. This could be of great help to understand the apparition of eventual hot spots due to ageing of the chambers subjected to high radiation levels. The monitored homogeneity information could then be combined with a monitoring of the activity of each individual channel in order to have a finer information. Monitoring tools have been suggested and need to be developped for this purpose.

```

1741   float MeanPartSDev = GetTH1StdDev(StripNoiseProfile_H.rpc[T][S][p]);
1742   float strip_homog = (MeanPartRate==0)
1743     ? 0.
1744     : exp(-MeanPartSDev/MeanPartRate);
1745   StripHomogeneity_H.rpc[T][S][p]->Fill("exp -#left(#frac{\#sigma_{Strip}
1746     \rightarrow Rate}{\#mu_{Strip Rate}}\#right)",strip_homog);
1747   StripHomogeneity_H.rpc[T][S][p]->GetYaxis()->SetRangeUser(0.,1.);
1748
1749   float ChipStDevMean = GetTH1StdDev(ChipMeanNoiseProf_H.rpc[T][S][p]);
1750   float chip_homog = (MeanPartRate==0)
1751     ? 0.
1752     : exp(-ChipStDevMean/MeanPartRate);
1753   ChipHomogeneity_H.rpc[T][S][p]->Fill("exp -#left(#frac{\#sigma_{Chip}
1754     \rightarrow Rate}{\#mu_{Chip Rate}}\#right)",chip_homog);
1755   ChipHomogeneity_H.rpc[T][S][p]->GetYaxis()->SetRangeUser(0.,1.);

```

Source Code B.17: Storage of the homogeneity into dedicated histograms.

1744 B.6.4.3 Strip masking tool

The offline tool is automatically called at the end of each data taking to analyse the data and offer the shifter DQM histograms to control the data quality. After the histograms have been published online in the DQM page, the shifter can decide to mask noisy or dead channels that will contribute to bias the final rate calculation by editing the mask column of `ChannelsMapping.csv` as can be seen in Figure B.3.

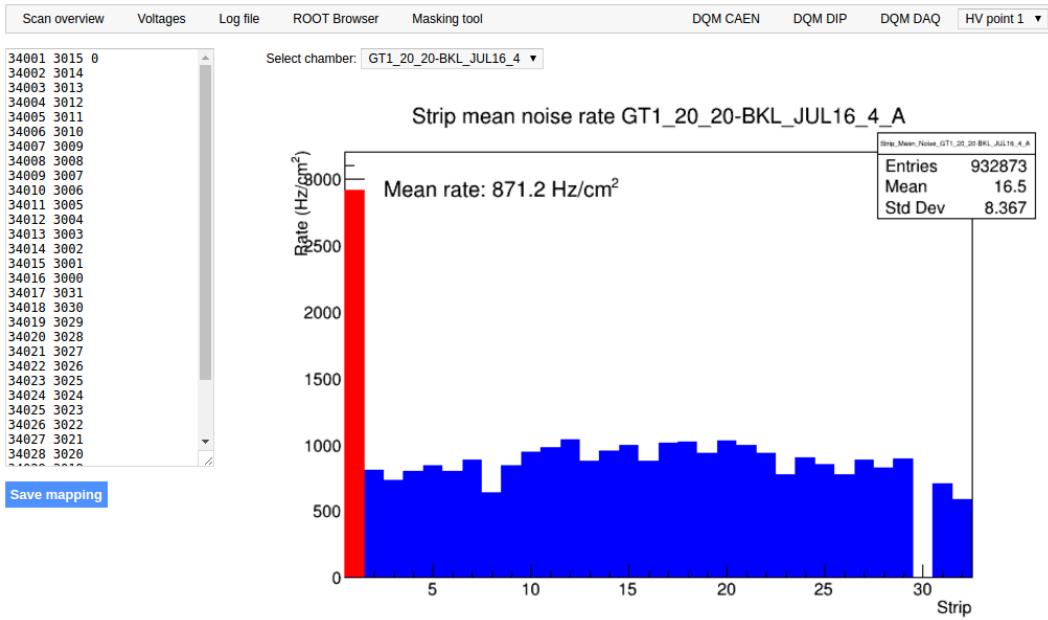


Figure B.3: Display of the masking tool page on the webDCS. The window on the left allows the shifter to edit ChannelsMapping.csv. To mask a channel, it only is needed to set the 3rd field corresponding to the strip to mask to 0. It is not necessary for older mapping file formats to add a 1 for each strip that is not masked as the code is versatile and the default behaviour is to consider missing mask fields as active strips. The effect of the mask is directly visible for noisy channels as the corresponding bin turns red. The global effect of masking strips will be an update of the rate value showed on the histogram that will take into consideration the rejected channels.

1750 From the code point of view, the function `GetTH1Mean()` is used to retrieve the mean rate par-
 1751 tition by partition after the rates have been calculated strip by strip and filled into the histograms
 1752 `StripNoiseProfile_H.rpc[T][S][p]`, as described through Source Code B.18.

1753 Once the mask for each rejected channel has been updated, the shifter can manually run the of-
 1754 fline tool again to update the DQM plots, now including the masked strips, as well the rate results
 1755 written in the output CSV file `Offline-Rate.csv`. If not done during the shifts, the strip masking
 1756 procedure needs to be carefully done by the person in charge of data analysis on the scans that were
 1757 selected to produce the final results.

```
1758 float GetTH1Mean(TH1* H){  
1759     int nBins = H->GetNbinsX();  
     int nActive = nBins;  
     float mean = 0.;  
  
     for(int b = 1; b <= nBins; b++){  
         float value = H->GetBinContent(b);  
         mean += value;  
         if(value == 0.) nActive--;  
     }  
  
     if(nActive != 0) mean /= (float)nActive;  
     else mean = 0.;  
  
     return mean;  
}
```

Source Code B.18: The function `GetTH1Mean()` is used to return the mean along the y-axis of `TH1` histograms containing rate information. In order to take into account masked strips whose rate is set to 0, the function looks for masked channels and decrement the number of active channels for each null value found.

¹⁷⁶¹ **B.6.4.4 Output CSV files filling**

```

for (UInt tr = 0; tr < GIFInfra->GetNTrolleys(); tr++){
    UInt T = GIFInfra->GetTrolleyID(tr);

    for (UInt sl = 0; sl < GIFInfra->GetNSlots(tr); sl++){
        UInt S = GIFInfra->GetSlotID(tr,sl) - 1;

        float MeanNoiseRate = 0.;
        float ClusterRate = 0.;
        float ClusterSDev = 0.;

        for (UInt p = 0; p < GIFInfra->GetNPartitions(tr,sl); p++){
            float MeanPartRate = GetTH1Mean(StripNoiseProfile_H.rpc[T][S][p]);
            float cSizePart = NoiseCSize_H.rpc[T][S][p]->GetMean();
            float cSizePartErr = (NoiseCSize_H.rpc[T][S][p]->GetEntries()==0)
                ? 0.
                : 2*NoiseCSize_H.rpc[T][S][p]->GetStdDev() /
                    sqrt(NoiseCSize_H.rpc[T][S][p]->GetEntries());
            float cMultPart = NoiseCMult_H.rpc[T][S][p]->GetMean();
            float cMultPartErr = (NoiseCMult_H.rpc[T][S][p]->GetEntries()==0)
                ? 0.
                : 2*NoiseCMult_H.rpc[T][S][p]->GetStdDev() /
                    sqrt(NoiseCMult_H.rpc[T][S][p]->GetEntries());
            float ClustPartRate = (cSizePart==0) ? 0.
                : MeanPartRate/cSizePart;
            float ClustPartRateErr = (cSizePart==0) ? 0.
                : ClustPartRate * cSizePartErr/cSizePart;

            outputRateCSV << MeanPartRate << '\t'
                << cSizePart << '\t' << cSizePartErr << '\t'
                << cMultPart << '\t' << cMultPartErr << '\t'
                << ClustPartRate << '\t' << ClustPartRateErr << '\t';

            RPCArea += stripArea * nStripsPart;
            MeanNoiseRate += MeanPartRate * stripArea * nStripsPart;
            ClusterRate += ClustPartRate * stripArea * nStripsPart;
            ClusterSDev += (cSizePart==0)
                ? 0.
                : ClusterRate*cSizePartErr/cSizePart;
        }

        MeanNoiseRate /= RPCArea;
        ClusterRate /= RPCArea;
        ClusterSDev /= RPCArea;

        outputRateCSV << MeanNoiseRate << '\t'
            << ClusterRate << '\t' << ClusterSDev << '\t';
    }
}

```

¹⁷⁶³ *Source Code B.19: Description of rate result calculation and writing into the CSV output Offline-Rate.csv. Are saved into the file for each detector, the mean partition rate, cluster size and cluster mutiplicity, along with their errors, for each partition and as well as a detector average.*

```

for (UInt tr = 0; tr < GIFIInfra->GetNTrolleys(); tr++) {
    UInt T = GIFIInfra->GetTrolleyID(tr);
    for (UInt sl = 0; sl < GIFIInfra->GetNSlots(tr); sl++) {
        UInt S = GIFIInfra->GetSlotID(tr,sl) - 1;
        for (UInt p = 0; p < GIFIInfra->GetNPartitions(tr,sl); p++) {
            float noiseWindow =
                BMTDCWINDOW - TIMEREJECT - 2*PeakWidth.rpc[T][S][p];
            float peakWindow = 2*PeakWidth.rpc[T][S][p];
            float windowRatio = peakWindow/noiseWindow;

            float PeakCM = MuonCMult_H.rpc[T][S][p]->GetMean();
            float PeakCS = MuonCSize_H.rpc[T][S][p]->GetMean();
            float NoiseCM = NoiseCMult_H.rpc[T][S][p]->GetMean() *windowRatio;
            float NoiseCS = NoiseCSize_H.rpc[T][S][p]->GetMean();
            float MuonCM = (PeakCM<NoiseCM) ? 0. : PeakCM-NoiseCM;
            float MuonCS = (MuonCM==0 || PeakCM*PeakCS<NoiseCM*NoiseCS)
                ? 0.
                : (PeakCM*PeakCS-NoiseCM*NoiseCS)/MuonCM;
            float PeakCM_err = (MuonCMult_H.rpc[T][S][p]->GetEntries()==0.)
                ? 0.
                : 2*MuonCMult_H.rpc[T][S][p]->GetStdDev() /
                    sqrt(MuonCMult_H.rpc[T][S][p]->GetEntries());
            float PeakCS_err = (MuonCSize_H.rpc[T][S][p]->GetEntries()==0.)
                ? 0.
                : 2*MuonCSize_H.rpc[T][S][p]->GetStdDev() /
                    sqrt(MuonCSize_H.rpc[T][S][p]->GetEntries());
            float NoiseCM_err = (NoiseCMult_H.rpc[T][S][p]->GetEntries()==0.)
                ? 0.
                : windowRatio*2*NoiseCMult_H.rpc[T][S][p]->GetStdDev() /
                    sqrt(NoiseCMult_H.rpc[T][S][p]->GetEntries());
            float NoiseCS_err = (NoiseCSize_H.rpc[T][S][p]->GetEntries()==0.)
                ? 0.
                : 2*NoiseCSize_H.rpc[T][S][p]->GetStdDev() /
                    sqrt(NoiseCSize_H.rpc[T][S][p]->GetEntries());
            float MuonCM_err = (MuonCM==0) ? 0. : PeakCM_err+PeakCS_err;
            float MuonCS_err = (MuonCS==0 || MuonCM==0) ? 0.
                : (PeakCS*PeakCM_err + PeakCM*PeakCS_err +
                    NoiseCS*NoiseCM_err + NoiseCM*NoiseCS_err +
                    MuonCS*MuonCM_err)/MuonCM;

            float DataRatio = MuonCM/PeakCM;
            float DataRatio_err = (MuonCM==0) ? 0.
                : DataRatio*(MuonCM_err/MuonCM + PeakCM_err/PeakCM);
            float eff = DataRatio*Efficiency0_H.rpc[T][S][p]->GetMean();
            float eff_err = DataRatio*2*Efficiency0_H.rpc[T][S][p]->GetStdDev() /
                sqrt(Efficiency0_H.rpc[T][S][p]->GetEntries()) +
                Efficiency0_H.rpc[T][S][p]->GetMean()*DataRatio_err;

            outputEffCSV << eff << '\t' << eff_err << '\t'
                << MuonCS << '\t' << MuonCS_err << '\t'
                << MuonCM << '\t' << MuonCM_err << '\t';
        }
    }
}

```

Source Code B.20: Description of efficiency result calculation and writing into the CSV output Offline-L0-EffCl.csv. Are saved into the file for each detector, the efficiency, corrected taking into account the background in the peak window of the time profile, muon cluster size and muon cluster multiplicity, along with their errors, for each partition and as well as a detector average.

C

1766

1767

Structure of the hybrid simulation software

1768

C.1 Introduction

1769

insert text here...

