GEANT4 Muon Digitization in the ATHENA Framework

D. Rebuzzi [a], K. A. Assamagan [b], A. Di Simone [c], Y. Hasegawa [d], N. Van Eldik [e]

[a] *Dipartimento di Fisica Nucleare e Teorica Università di Pavia and INFN, Sezione di Pavia via A. Bassi 6, I-27100, Pavia, Italy*

[b] *Brookhaven National Laboratory (BNL) Department of Physics, Upton , NY 11793, USA*

[c] *CERN PH-ATC, 1211 Geneva 23, Switzerland*

[d] *Department of Physics, Shinshu University, 3-1-1 Asahi, Matsumoto 390-8621 Nagano, JAPAN*

[e] *National Institute for Nuclear Physics and High Energy Physics (NIKHEF) Kruislaan 409, 1098 SJ Amsterdam, The Netherlands*

Abstract

The aim of this note is to describe the Muon Digitization software packages, completely re-written to operate on GEANT4 hits. The Muon Digitization is independent of the GEANT4 detector simulation, runs within the ATHENA framework.

Key words: muon digitization, DC2 simulation, ATHENA algorithms

1 Introduction

The present note describes the Muon Digitization software, recently re-written to run in the ATHENA framework.

ATHENA is a control framework based on the GAUDI architecture originally developed by the LHCb experiment. This architecture has been extended through collaboration with ATLAS, and an experiment neutral implementation, also called GAUDI, has been created. ATHENA is the result of this kernel framework, together with ATLAS-specific enhancements, such as the event data model and event generator framework. It is intended that almost

all software used in physics, whether for event generation, reconstruction or analysis, will be in the form of specializations of a few specific components, where specialization means adding functionalities to a standard component while keeping its interface the same. Within the application framework this is done by deriving new classes from one of the base classes: `DataObject, Algorithm, Converter`.

Anything which is essentially a procedure, i.e., a set of rules for performing transformations on more data-like objects, or for creating new data-like objects should be designed as a class derived from the `Algorithm` base class. The role of an algorithm is to take input data, manipulate it and produce new output data. In general, an algorithm will be configurable: it will require certain parameters (*properties*), such as cut-offs, upper limits on the number of iterations, convergence criteria, etc., to be initialized before the algorithm may be executed. These parameters may be specified at run time via *job Options* files. In ATHENA, data objects produced by algorithms are saved in common area in the transient memory from where other modules can access them and produce new data objects. The Transient Data Store (TDS) is the mechanism to provide such a common area in the GAUDI architecture: an algorithm creates a data object in the TDS, thus allowing other algorithms to access it. StoreGate (SG) is the ATLAS implementation of the TDS. It manages the data objects in transient form, as well as their transient/persistent conversion. The new Muon Digitization consists of four algorithms, one for each muon technology [1] . The Muon Digitization provides the production of collections of simulated muon digits out of muon hit collections from the GEANT4 simulation, where both the input and the output objects are assumed to be in SG. In the present architecture, muon digits are converted to Raw Data Objects (RDOs) or to byte streams before undergoing the persistent storage [2] . RDOs are the output of the readout electronic, thus are the constituents of the byte stream.

The goal of the muon digitization is to simulate the output signal of the muon detector in ATLAS given the output of GEANT4 detector simulation. The digitization process consists of two steps: in first, the output of the detector simulation, henceworth refered to as GEANT4 hits, is converted to muon digits. The second step, the muon digits are converted to RDO from which the byte stream (the electronic output) is obtained. The muon digits are objects that can be fed directly in the muon reconstruction, thus are known as Reconstruction Input Objects (RIO). The muon off-line reconstruction dataflow starts from the raw data, the byte stream. The byte stream is read into the TDS as transient RDO data objects. In a subsequent steps, the RDO are con-

[1] The code (XXX_Digitization packages, where XXX = MDT, RPC, TGC, CSC) is in the CVS repository [1] and part of the ATHENA software since release 7.4.0.
[2] The persistency of the Muon Spectrometer RDOs is in place within the ATHENA framework since release 7.6.0.

verted into RIO (digits and or clusters), through calibration services, pedestal and noise handling, etc. The step, RDO to RIO, is where the raw data is "prepared" before being fed into the off-line reconstruction. It thus follows that the muon digitization operates in the reverse steps of the off-line reconstruction dataflow, by producing the RIO first, then the RDO (or byte stream subsequently). In this paper, we will describe the detailed implementation of the first step of the muon digitization, where the detector simulation output (GEANT4 hits) are converted into the RIO (muon digits). The second step, which convert the RIO in the RDO (byte stream) is implemented in separate algorithms and documented elsewhere (**we need to cite some paper here for digits to RDO to byte stream**). What follows describes the fundamentals of the Muon Digitization algorithms. In Section 2, the global structure of the Muon Digitization is outlined, with particular emphasis to the infrastructure to handle the muon *pile-up* and to Muon Hits and Muon Digit objects. Sections from 3 to 6 describe the detail of the digitization algorithms (one for each Muon technology) and their validation against the information from the MCTruth.

2 Muon Digitization

The information coming from the interaction of the simulated tracks with the sensitive part of the detector (referred to as the *hit objects*) is transformed by the digitization algorithms into *digit objects* which are the input to the off-line reconstruction programs. The Muon Digitization container houses also a package, MuonDigiExample, which contains two monitoring algorithms (ReadMuonSimHits and ReadMuonDigits, to check the correct hits/digit writing to SG) and four digit validation algorithms (XXXDigitValidation), one for each technology which allow to check the main digit parameters against the MC Truth information. A Muon Digitization jobOptions fragment is included in the global ATLAS Digitization jobOptions (AtlasDigitization.py in [6]) starting from ATHENA release 7.4.0. The ATLAS Digitization reads hit files generated by the GEANT4 complete ATLAS simulation and writes an output file with the digits objects. Muon Digitization has been designed to be independent from the GEANT4 simulation. The Muon Digitization code relies only on MuonGeoModel read-out geometry and the detector-specific MuonSpectrometer OID scheme [11].

2.1 Muon Hits

Hit production in GEANT4 is provided by *Sensitive Detectors*. Muon hits are generated by the XXXSensitiveDetector [2] (where XXX = MDT, RPC, TGC,

CSC) when charged particles cross the sensitive part of the Muon chambers.

A given volume described in the GEANT4 geometry becomes *sensitive* when associated to a properly implemented Sensitive Detector, which is an instance of a class different from the one describing the *real* geometry (*tracking geometry* in the following). The clear separation between Sensitive Detectors and Geometry classes allows to easily decouple the description of the tracking geometry from the one of the readout geometry. Each time a particle trajectory crosses a volume which can generate hits, the kernel is responsible for calling the corresponding Sensitive Detector, which implements the hit generation algorithms. Hence, the Sensitive Detector generates a list of hits, which are stored for further processing. Muon hits are defined as classes in [3].

Hits are labelled by a *Simulation Identifier*, SimID, a 32-bit integer in which the geometry information about the hit position is stored. The `stationName`, `stationEta` and `stationPhi` information is common to all the muon technologies, the remaining identifier fields are specific for any given precision or trigger sensitive element.

The SimIDs are built by means of the XxxHitIdHelper classes (Xxx = Mdt, Rpc, Tgc and Csc) of the MuonSimEvent package [3]. The base class HitIdHelper [4] provides methods for initializing, setting and retrieving the identifier field values. The XxxIdHelper are all derived from the same base class and have specialized methods returning indices relevant for each technology information.

The SimIDs in principle do not coincide with the Offline Identifiers (OIDs) associated to the digit object. The design idea is to keep the event simulation disentangled from the digitization/reconstruction. Within this identifier scheme, only the XxxHitIdHelper classes depend on GEANT4. The hit objects are filled from data provided by the GEANT4 simulation but they do no depend on it. Therefore, the event is completely decoupled from the geometry, the only communication occuring via the identifiers.

Hits have a very ligth content. This is mainly due to the fact that all the geometrical information is encapsuled in the SimIDs. Thanks to the retrieving mechanism mentioned above, the SimIDs are enough to access to all the geometrical information needed later on. For example, in the CSC, the chamber dimensions and the strip pitches are needed during the digitization. They are obtained from the muon geometry model using the chamber offline identifiers, the latter being constructed from the hit identifiers, SimIDs. In addition to the hit identifier, each Muon hits contains the quantity to be digitized. Table 1 list the contents of the SimIDs and of the Muon hit objects for each Muon technology. The GEANT4 hits are collected using AthenaHitsVector containers, one for each Muon technology, where they are inserted in a random way

	MDT	RPC	TGC	CSC
SimID	StationName, PhiSector, ZSector			
	MultiLayer Layer Tube	DoubletZ DoubletR GasGapLayer DoubletPhi MeasuresPhi	GasGap	ChamberLayer WireLayer
Muon Hit	SimID globalTime driftRadius localPos (trackNumber)	SimID globalTime localPos (trackNumber)	SimID globalTime localPos dirCos (trackNumber)	SimID HitStart HitEnd partID (trackNumber)

Table 1
Contents of the Simulation Identifiers (SimIDs) and of the Muon hit objects for the four Muon technologies. The first three fields of the SimID specify the name of the station where the hit is located and its (η, ϕ) position. The remaining fields give information about the specific sensitive volume in which the hit occurred. Muon hits contain the SimID and additional information required by the digitization and the pile-up procedures.

(no sorting is performed at the simulation level). There are therefore four independent hit collections in the muon spectrometer, one for each technology and each hit collection contains, for each event, all the hits in that particular technology. The obtained XxxSimHitCollections are persistified using POOL so that a re-digitization can be done without re-simulating the events.

2.2 Muon Digits

Muon Digits are labelled by an offline identifier (OID) which create the connection to the reconstruction. This contains the digit geometry information packed according to the classes in [7]. OIDs are initialized using the identifier dictionary [8], an xml file which specifies the off-line identifier fields and their allowed ranges. For the muon spectrometer, there are various identifier dictionaries for test beam, or for different detector layouts (P03, Q, etc). The Muon digit classes are located on CVS under [9] The geometrical description of the detector elements are obtained from the methods of the XxxDetectorElement classes of the MuonGeoModel *read-out* geometry ([10]). They allow to get information (like the hit distance from the RO chamber side) needed by the Digitizers to contruct the digit parameters. In the `execute()` method of the Xxx_Digitizer the XxxSimHitCollections vectors are taken and vectors of type XxxDigitCollection are filled with XxxDigit objects.

	MDT	RPC	TGC	CSC
OID	stationName, stationEta, stationPhi, technology			
	multiLayer	doubletR	tgcGasGap	chamberLayer
	tubeLayer	doubletZ	isStrip	wireLayer
	tube	doubletPhi	channel	cscMeasuresPhi
		rpcGasGap		cscStrip
		rpcMeasuresPhi		
		rpcStrip		
Muon Digit	OID	OID	OID	OID
	TDC count	propTime		charge
	(ADC count)	globalTime		
RDO	fired tube	fired channel of CM	raw hit or raw coincidence	4 ADC samplings + strip address

Table 2
Contents of the Offline Identifiers (OIDs), of the Muon digit objects and of the Muon RDOs for the four Muon technologies. The first four fields of the OID specify the name of the station where the hit is located, its (η, ϕ) position and the technology type. The remaining fields give information about the specific sensitive volume in which the digit occurred. Muon digits contain the OID and additional information required by the reconstruction.

2.3 Infrastructure for Event Pile-Up

In addition of being capable of handling hits coming from a single bunch crossing, the digitization, as implemented at the time of this writing, is also be able to handle *piled-up* collisions. Before performing the digitization, hits from several bunch crossings are overlaid taking into account the global time of the hit which is defined as the GEANT4 hit time plus the bunch crossing time with respect to the main crossing. Some tools exist to do this, namely, the TimedHitPointer and the TimeHitPtrCollection classes of [6], which allow one to retrieve the overlaid hits, sorted according detector element offline identifiers. In order to do the overlay and the sorting, the hit classes implement an ordering operator and a hit time open function to return the GEANT4 hit time. Furthermore, some tools also exists to link the overlaid hits the correct Monte Carlo Truth particles: this requires the merging of the Monte Carlo collections of the input events, and subsequently updating the hit-to-particle associations to point the merged Monte Carlo collections.

Simulated GEANT4 hits persistified using POOL can be read in together with previously generated minimum bias events. The hit overlay and sorting according detector elements is then carried out as described above, and the digitization proceeds afterwards. The pile-up model can also handle the case of the cavern background, although this has not been exercised at the time of

the current note.

2.4 Association to Monte Carlo Truth

The muon digitization produces muon digits, i.e., reconstruction input objects needed by the muon reconstruction packages, namely MOORE [**?**] and Muonboy [**?**]. In a subsequent step, the muon digits are convertered to muon RDO, the transient representation of the byte stream. The production of the RDO constitutes the simulation of the electronic output, the raw data!. Any reference to the Monte Carlo information is lost after the digitization, that is, the muon digits or the RDO do not carry any link (pointer, associations) to the original simulated particles. However, such "links" are necessary to establish the Monte Carlo truth tracks and for validation purposes. During the muon digitization, a separate object is recorded and can be persistified, to maintain the link to th original simulated paricles at the digit or RDO level. The recorded object in question is a map of muon off-line identifiers to MuonSimData objects [**?**]. MuonSimData, originally adopted from the Inner Detector implementation, stores the simulation information associated with a simulated raw data object, in two data members. One is an integer, the *simulation data word*, which is a packed information summarizing the digitization. Its interpretation may depend on which of the four sub-detectors is concerned, but will typically contain bit-flags for *noise, lost in readout*, etc. The other consists of a vector of `pair<link to particle, 2 floating-point values>` specifying the link to the original particle and up to two additional pieces of information — encapsulated in the class MuonMCData — depending on the Muon technology: deposited energy, charge, hit time, etc. The objects of type MuonSimData are not applicable to the real production running. However, they can be made persistent in the simulation of non pile-up situations and are useful to carry the associations to the original particles to the tracking stage.

3 MDT_Digitization

Simulated muons interact with materials along their path causing ionization of the tube gas (the MDT Sensitive Detector) in the drift tube of the MDT chambers. During the tracking, collections of MDTSimHits are recorded, each containing the impact parameter and the hit position in the global coordinate reference system, as discussed in Section 2.1. As discussed before, the digitization procedure should convert the hit information from the GEANT4 simulation into an output which should resembles the output signal of the ATLAS detector. The MDT_Digitization offers the infrastructure for the MDT

Digit building out of any *valid*[3] MDTSimHit, each MDT Digit consisting of an OID, a TDC count and, optionally, a ADC count, as in Table 2. The MDT chambers are equipped with TDCs (Time-to-Digit Converters) which measure the signal pulse time for each MDT passing a predefined threshold. The pulse time is measured with respect to the global LHC clock and includes the following contributions:

- time of flight of the particle from its generation vertex to the tube;
- bunch crossing offset if the particle comes from a previous/next event;
- signal propagation delay along the tube;
- additional delays due to cables/electronics;
- the drift time.

In addition, the electronics have an on-board Wilkinson ADC which integrate the input pulse over a predefined time window providing a measure of the pulse height. Starting from the impact parameter, the `driftRadius` of Table 1, associated to the MDTSimHit, the MDT_Digitization performs several tasks:

- conversion of the drift radius into a drift time,
- calculation of the time structure of the event,
- trigger match,
- conversion of total time into TDC counts.

For the $r \rightarrow t$ conversion, two different AlgTools have been implemented and are available in the MDT_Digitization package. Due to a modularity of the architecture, they can be selected via jobOptions setting the property `DigitizationTool` of the MDT_Digitizer algorithm. The user can select a very detailed time-consumpting $r \rightarrow t$ procedure (MDT_Response_DigiTool) or a fast drift distance to time conversion which relies on an external rt relation (RT_Relation_DigiTool). Both provide a routine which converts the impact parameter of the track into a drift time. The first provides also a ADC count correlated with the drift time, to evaluate the slewing corrections, while in RT_Relation_DigiTool, the ADC count is set to fixed number for all digits. The two tools are described in details in Sections 3.1 and 3.2 respectively. The MDT_Digitizer, is a standard ATHENA algorithm with the following functionalities:

- in method `initialize()`: StoreGate and PileUpMerge Service initialization; retrieving of the pointer to MuonGeoModelManager, by which the MdtIdHelpers for the digit offline identifier building are initialized; initialization of the MdtDigitContainer; retrieving of the simulation identifier helper and of the pointer to the digitization tool;
- in method `execute()`: record of the digits and of the SDOs containers in StoreGate; MDTSimHit collection merging using the TimedHitPtrCollec-

[3] In the sense which will be specify in Section 3.3.

tion sorted container; loop over the TimedHitPtrCollection for the given DetectorElement, performing the actions of the `handleMDTSimhit()` method described below; digit creation and storing in the DigitContainer (in `createDigits()` method);
- in method `finalize()`: a SUCCESS StatusCode is returned if the digitization procedure ends succesfully.

In the `handleMDTSimhit()` method, for each MDTSimHit of the time-sorted TimedHitPtrCollection, the SimID is unfolded to get the geometrical information about hit position and the digit identifier (Offline Identifier) is created via the MdtIdHelper. Several checks can be performed on demand: on the hit, to skip it if corrupted, on its associated tracking element, to check its validity, before retieving the distance to the RO chamber side. The drift radius (together with the distance to RO) is passed as input to one or the other MDT_Digitization tools (via the class DigiTool), which returns the correspondent *drift time* and the ADC count.

With the `digitize()` method of DigiTool the tube response is simulated: on demand, the propagation delay of the signal along the wire is calculated (using the distance to RO information from MdtReadoutElement of MuonGeoModel) and added to the drift time together with the time of flight and the bunch time.

Finally the hit is inserted in a vector together with the digit OID, the drift time and the ACD count. The `createDigits()` method loops on the above sorted hit map to check if the tube where the hit occurs has already been fired, if yes an additional check on the tube dead time is performed before registering also the second hit (a single tube can produce more than one hit). If the hit time (to which the tof time has been subtracted) lies within the Matching Window or within the Mask Window, the hit time is converted into a TDC count. A detailed discussion of the time structure of the event is given in Section 3.3.

Using the OID, the TDC count and the ADC count, the MdtDigit object is built and inserted into the DigitCollection. A new DigitCollection is created for each MDT chamber. Table 3 summarizes the main MDT_Digitization properties which can be selected via jobOption to configure the digitization package and their deafult values.

3.1 MDT_Response_DigiTool

The MDT_Response package provides a realistic simulation of the signal formation in MDT tubes. The output for a given impact parameter consists of a drift time and a charge measurement which is correlated with the drift time,

Property	Default Value
OffsetTDC	800 ns
ns to TDC conversion	0.78125
ResolutionTDC	0.5 ns
Signal Speed	299.792458 $\cdot 10^8$cm/s
Use Attenuation	FALSE
Use Tof	TRUE
Use Prop Delay	TRUE
Use Time Window	FALSE
Bunch Count Offset	-300 ns
Matching Window	1000 ns
Mask Window	700 ns
DeadTime	700 ns
Check on MDTSimHits	TRUE
Digitization Tool	MDT_Response_DigiTool

Table 3
Main properties of the MDT_Digitization package and their default values.

thus allowing time slewing corrections to improve the spatial resolution. The simulation performs several subsequent steps:

- generation of clusters along the trajectory of the particle;
- propagation of the clusters to the wire;
- convolution of the raw pulse with the amplifier response function;
- determination of the time at threshold and the integration of the signal pulse.

Figure 1 show a schematic overview of physics in the tube. First, the clusters are generated along the trajectory. During cluster generation cluster size and position fluctuations are taken into account. The clusters are propagated to the wire using a single electron rt relation. A simulation of the diffusion is performed. The single electron rt relation was obtained from GARFIELD and effectively contains the gas properties. The cluster arrival time spectrum is convoluted with the amplifier response function producing a raw signal pulse as shown in Figure 2. The threshold passing time of the pulse is determined, subsequently the pulse is integrated over a fixed integration window. The charge integral is converted into the Wilkinson ADC output. The time at threshold and the ADC output are returned for further use in the MDT_Digitization. The simulation takes into account all mayor contributions to the tube resolution. In addition, two other effects can be simulated (they are not switched on by default, as but can be selected using the jobOptions file):

- The attenuation of the signal pulse while propagating to the readout can be taken into account. In this setting the signal pulse is reduced by a factor

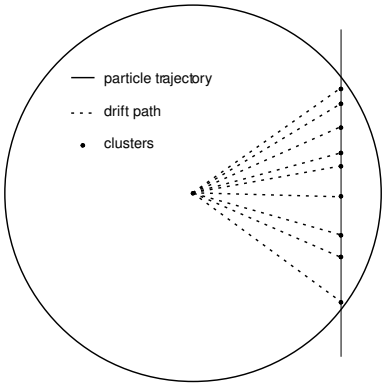


Fig. 1. *Trajectory as used in the simulation. The dots represent the clusters generated along the trajectory, the dotted lines represent the propagation of the clusters to the wire.*

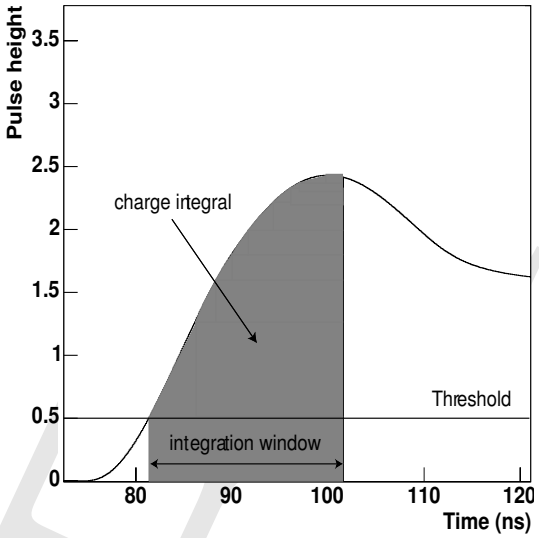


Fig. 2. *Determination of the time at threshold and the successive integrate of the charge.*

$e^{-d_{ro}/l_{att}}$, where d_{ro} is the distance to the readout and l_{att} is the attenuation length of the tube. The attenuation of the signal has a small effect on the tube resolution and is not completely neglectable for chambers with have long tubes.

- The effect of the magnetic field on the drift times can be taken into account. In this case the strength of the magnetic field is retrieved from the DetectorElement. The time shift due to drift in the magnetic field has been parametrized as a function of the drift time and the field strength, it is added to the drift time without magnetic field.

Figure 3 shows the spatial resolution as a function of the impact parameter. It was obtained by fitting a Gaussian distribution to the radial residuals obtained by subtracting the drift radius obtained by converting the measured drift time into a radius from the input impact parameter. The values shown in the figure are the widths of the fit which do not take into account eventual biases. Thus the resolution as show is a lower boundary of the intrinsic resolution of the simulation. In addition a more conservative estimate of the resolution is given by the RMS of the residual distributions. Figure 4 shows the correlation between the ADC counts and the drift time.

3.2 RT_Relation_DigiTool

This digitization tool relies on an external ASCII file (from Garfiled simulation or from the data analized with an autocalibration program) which should

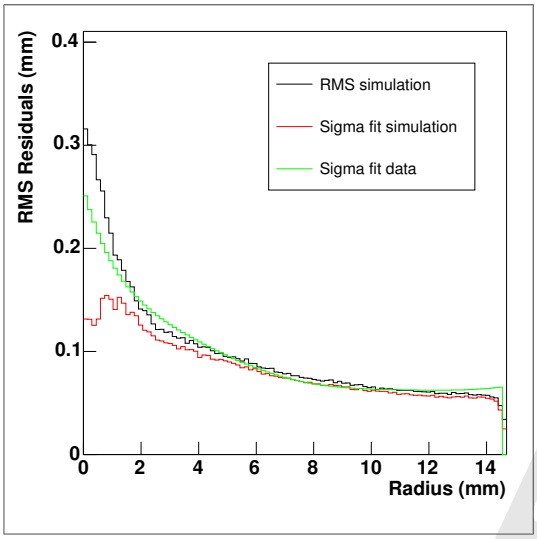


Fig. 3. *Intrinsic simulated spatial resolution.*

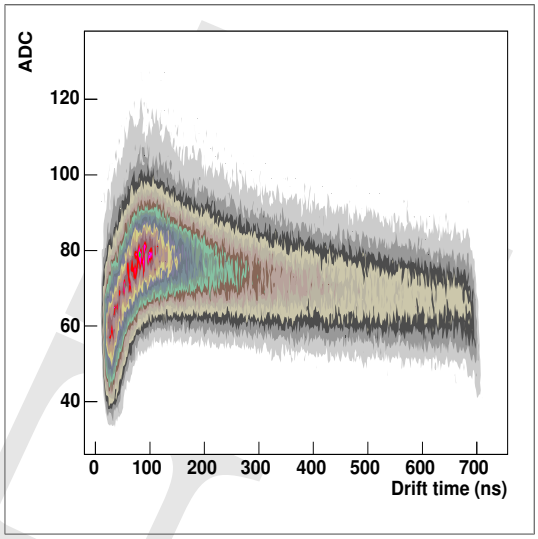


Fig. 4. *ADC vs drift time.*

have a list of r, t and the correspondent resolution on t. The tool operate the drift radius to drift time conversion and applies a gaussian smearing on t according to the given t resolution. Despite the very simple structure, this tool is extremely useful when comparing to data (for the testbeam event digitization, for instance) since the same rt relation as used on the real data can be used to create the simulated digits.

3.3 Simulation of the Time Structure of the Event

According to the different contributions to the pulse time, the drift time should consist of the following components:

$$t_{tot} = t_{tof} + t_{bunch} + t_{prop} + t_{delay} + t_{drift} \quad (1)$$

The time of flight t_{tof} is obtained from the MDTSimHit in form of `globaltime`. The propagation delay t_{prop} is calculated from the position of the hit along the tube (obtained from the MuonGeoModel read-out geometry) and the signal propagation speed. Additionally, in case of pile-up, a bunch crossing offset t_{bunch} is taken into account. The TDCs produce a time stamp every time the input signal passes the threshold. To avoid not physical hit proliferation, the TDCs have a programmable dead time which is set to the maximum drift time of the tube [**??**] as shown in figure 5. The same principle is reproduced in the digitization procedure, where:

- the MDTSimHit with the smallest drift time in the event sets the dead time (for the given event) and is selected;
- any hit falling within the dead time window is discarded;

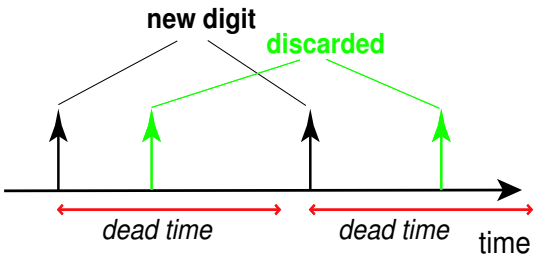


Fig. 5. *Dead time.*

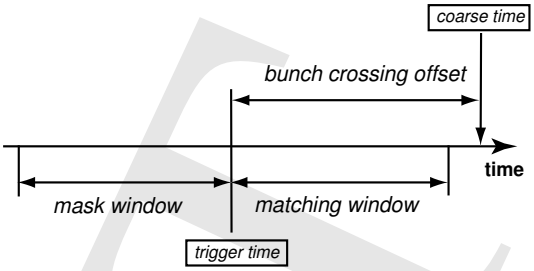


Fig. 6. *Time windows.*

- if a hit with a drift time larger then the dead time is present in the event, it is selected and the dead time is reset.

The dead time is given by three components [**??**]:

$$t_{dead} = t_{fix} + t_{drift} + t_{ADC}$$

Where t_{fix} is a predefined fix dead time which can be set via the jobOptions (default =), t_{drift} is the drift time and t_{ADC} is the output of the Wilkinson ADC. The resulting average deadtime is 1100 ns. A trigger match criterion is applied to all the selected hits following the same procedure (as described in [6]). First, a trigger time t_{trig} is calculated as follows:

$$t_{trig} = t_{delay} + t_{avetof} + t_{bunchoffset}$$

where t_{delay} is the time entering the TDC time calculation 1 and t_{avetof} is the average time a particle at light speed need to reach the center of the chamber. An additional offset $t_{bunchoffset}$ allows the windows to be positioned with respect to the GEANT4 global time. The offset should be negative to ensure that hits from this bunch crossing always have times larger than the trigger time. Then, for every hit the time $t_{TDC} - t_{trig}$ is matched with the time windows:

- if the time of a hit falls within the *matching window*, a MDT_Digit is produced;
- a hit in the *mask window* produces a MDT_Digit which contains no TDC count and is flagged as *masked*;
- any hit outside the windows is discarded.

Depending on the sizes of the two windows it is possible to have *more that one hit per event per tube*, both will be stored. TDC counts are given by a fixed bin size conversion (0.78125 TDC/ns) of the total time 1, smeared by a Gaussian distribution whose resolution can be selected via jobOptions (set by deafult to 0.5 ns). The TDC count is stored is stored into the MDT digit object. The TDC count 1 is required to be positive, any negative time is converted into a zero. A delay t_{delay} can be added to ensure that the total time is always

RPCHit	RPCDigit
SimID of the gas gap	OID of the strip
Position of the hit wrt the gas gap	
Hit time (time of flight of the particle)	Global time (tof + strip propagation)

Table 4
The information included in RPC hits and digits

positive. It is up to the user to set this offset correctly, in order to avoid negative times (the default of 800 ns is recommended).

3.4 MDTDigitValidation

4 RPC_Digitization

RPC hits are generated by the SensitiveDetector (SD) which assigns to them a Simulation Identifier (SimID), uniquely identifying the gas gap each hit is registered in. The position of the hit in the reference system of the gas gap is also stored, together with the time from the beginning of the event, i.e. the time of flight of the particle generating the hit. RPC hits are represented in the simulation code by instances of the class `RPCHit`. The digitization process takes care of adding to the hits the information necessary for further analysis (for example trigger algorithm simulation and track reconstruction). It translates any SimID to a Standard Offline Identifier (OID), which is used by the other ATHENA algorithms to uniquely identify RPC strips in the muon spectrometer. Using the position information provided by the hits, the digitization can properly calculate the propagation time of each electronic signal along its strip, add it to the time of flight of the hit and assign this *global time* to the digit. The information obtained is stored in a new instance of the class `RPCDigit` and posted in StoreGate for further processing. The main differences between RPC hits and digits are summarized in table 4

4.1 Cluster simulation

When a particle generates an avalanche in an RPC, charge signals are induced (and detected) on the readout strips. A set of n adjacent strips with signals is called a *cluster* of size n. In RPC operation, due to possible signal induction on more than one strip, cluster sizes are in general greater than 1, with an average cluster size at working point typically of 1.3. The hit production mechanism provided by the ATLAS Geant4 simulation does not include a tool for proper

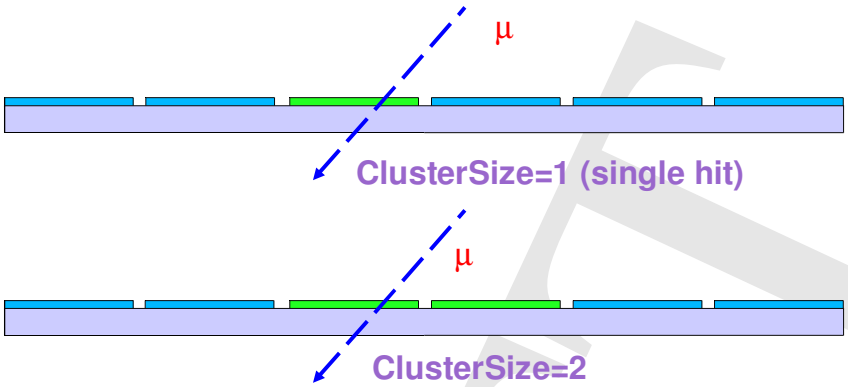


Fig. 7. *RPC clusters*

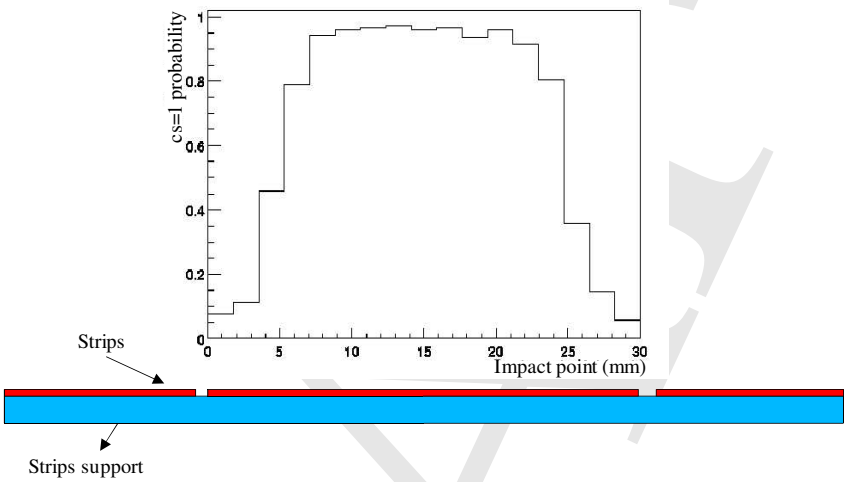


Fig. 8. *Probability to observe cluster size 1 as a function of the impact point along the strip.*

simulation of clusters. Thus a particle generates hits on only one strip, except when secondaries (for example δ) are produced and detected by neighboring strips.

The impact point along a strip is known to influence the size of the cluster the hit will generate. Figure 4.1 shows for example the probability to observe a cluster of size 1 as a function of the impact point of the track along the strip. The strip pattern is also represented. The probability is normalized to the number of clusters with sizes 1 or 2, i.e. for each bin, the complement to unity gives the probability to have a cluster with size 2. From the plot it is clear that, for example, a muon crossing the region between two adjacent strips will most likely generate a cluster of size 2, whereas a track passing in the middle of one strip would induce signals on that strip only. In a small amount of cases, clusters with sizes greater than 2 are also observed. The digitization algorithm reproduces the observed cluster sizes by generating, when necessary, digits on strips adjacent to the one actually crossed by the particle. Cluster simulation is carried on in three steps:

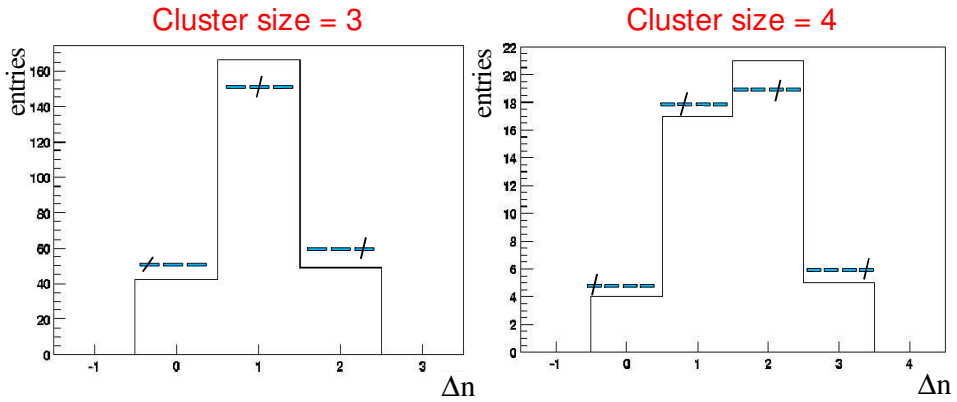


Fig. 9. *Cluster spread distributions for cluster of sizes 3 and 4.*

(1) experimental distributions are used to decide, according to the impact point of the particle along the strip, whether the cluster size will be 1 or 2.
(2) experimental distributions are used to decide what the final size of the simulated cluster will be.
(3) digits are created according to the results of the above steps.

Particular attention needs to be paid to the way the additional digits are created around the one actually crossed by the muon. Figure 4.1 shows the cluster spread distributions. For each cluster of a given size, the plots count which strip was actually crossed by the muon. Upon each bin, the corresponding track/strip configuration is showed. These experimental distributions are hence used to properly create the extra digits.

4.2 Class methods and properties

In `initialize()` method, the PileUpMerge and StoreGate services are initialized, and a pointer to an instance of the class `MuonDetectorManager` is retrieved from the detector store and used to obtain an `rpcIdHelper`. The ASCII file `G4RPC_Digitizer.txt` is read and its contest are used by the algorithm in order to simulate clusters.
Random numbers are obtained in the code from a dedicated stream via the `AtRndmSvc`, which is also initialized in the `initialize()` method.
The `execute()` has responsibility for steering the digitization/cluster simulation process. A loop over the RPCHits is performed, converting each SimID to OID.
The method `PhysicalClusterSize` is hence called, which creates a cluster of size 1 or two according to the impact point of the particle along the strip. The final size of the cluster is decided by the method `TurnOnStrips`.
The last step in the creation of the digitization is the calculation of the propagation time of the electrical signal along the strip length. This is done in the

name	description	default value
`Parameters`	the file with the experimental distributions to be used for cluster simulation	G4RPC_Digitizer.txt
`CTB2004`	true if digitizing data from the 2004 CTB	false
`InputObjectName`	name of the collection to be used as input	RPC_Hits
`OutputObjectName`	name of the output collection	rpc_digits
`WindowLowerOffset`	lower offset of the time window to be used for PileUp	-70 (ns)
`WindowUpperOffset`	upper offset of the time window to be used for PileUp	70 (ns)
`DeadTime`	dead time	50 (ns)

Table 5
Properties of RPC_Digitizer

`PropagationTime` method.
In the hit collections coming from the RPCSensitiveDetector, it sometimes happen that many hits are produced by the same crossing particle, which are very close both in space and time. This is related to ionization and production of secondaries in the gas, and it is thus safe, and also recommended, to eliminate these multiple hits before proceeding to reconstruction. The `execute()` method provides this functionality using a dead time: once a hit is found on a given strip, every other hit coming from the same strip before the dead time is ignored.
Table 5 shows the list of the properties of the RPC_Digitizer algorithm.

4.3 RPCDigitValidation

The algorithm `RpcDigitValidation` has been implemented in order to check the correctness of the hit production and digitization processes. The validation has been performed generating single muon events in the barrel, with no physics processes activated but transportation. For each event, the algorithm execution proceeds as described in the following:

- for each muon, its direction at the generation vertex is retrieved from the MonteCarlo information. With no physics processes activated, this direction

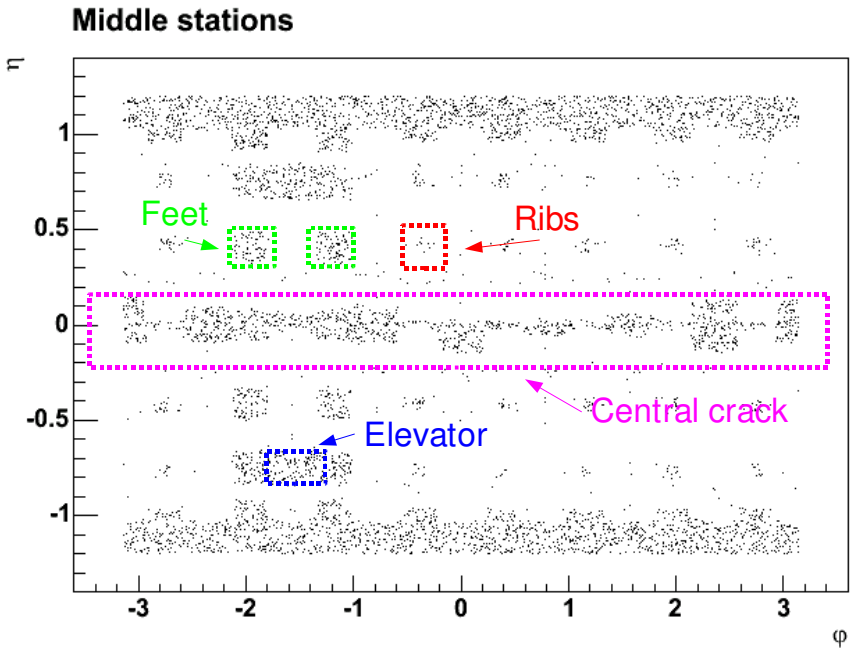


Fig. 10. *RPC digitization validation.*

is a good approximation of the muon trajectory.

- for each RPC digit, its distance from the muon trajectory is calculated
- for each of the three RPC layers (Middle-lowPT, Middle-Pivot, Outer), the closest digit to the muon is selected
- the fields of the OIDs of the selected digits are stored in an ntuple, together with the direction of the muon.

The resulting ntuple can be used to validate both the `RPCSensitiveDetector` and the `RPCDigitization` since it can easily spot any mistake in the generation of the SimID and in its translation to OID. For example the directions of the muons not producing any digit, or which produce digits too far from the track can be analyzed. In figure 4.3 the η and φ directions of muons not producing any RPC digits are plotted. These inefficiencies of the RPC system are concentrated, as expected, in regions not instrumented with RPCs:

- the feet of the ATLAS detector
- the ribs of the barrel magnet system
- the central ($\eta = 0$) crack
- the elevators
- the endcap regions ($|\eta| > 1$)

A similar plot has been produced for the outer stations and didn't show any abnormal inefficient region. With the same procedure, it is possible to look for digits which are created in the wrong place. Figure 4.3 shows for example the results of the validation on an earlier version of the simulation code. The plot shows the inefficiencies *plus* the digits with a wrong position with respect to the muon trajectory, and it clearly spots two regions with problems. Further

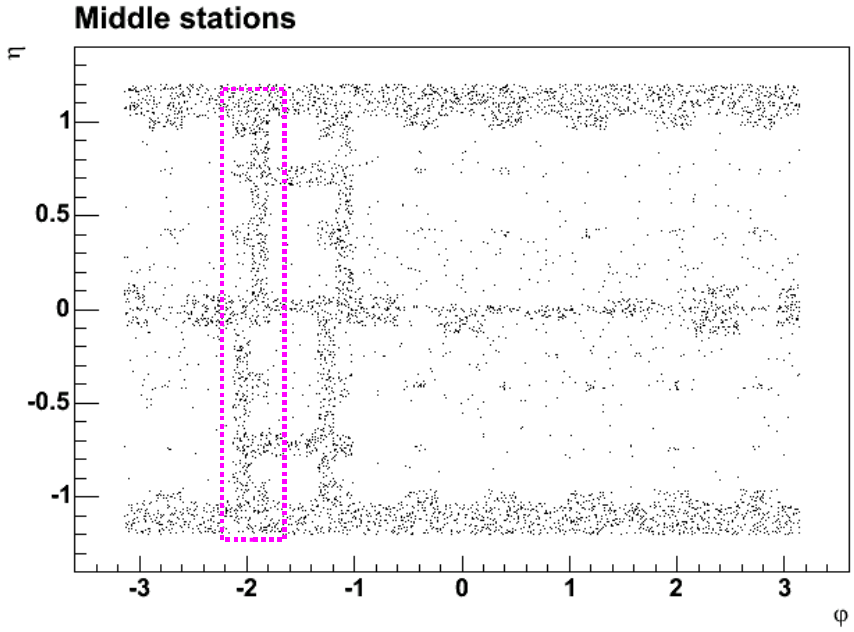


Fig. 11. *Wrong RPC digits in BMF chambers.*

investigation allowed to find an error in the `RPCSensitiveDetector` which assigned wrong SimID to the RPC hits in BMF stations.

5 TGC_Digitization

The TGC system is located at four stations (T1, T2, T3 from smaller $|z|$ to larger, in the middle station, T4 in the inner station) in the forward(F) and end-cap(E) regions in A and C sides of ATLAS detector. The stations are labeled as follows: T1F, T1E, T2F, T2E, T3F, T3E, T4F and T4E. Figure 12 shows longitudinal and R-ϕ views of the TGC layout.

Two or three TGCs in the stations are assembled to make a TGC module. The modules in T1F and T1E has three TGCs, i.e. three sensitive volumes (called as "triplet") and those in the other stations has two TGCs (called as "doublet"). Each TGC can provide two dimensional coordinate measurements in R and ϕ projections. The measurement along R direction is provided by gangs of sensitive wires and the ϕ coordinate is measured by strips. A triplet has thee layers of wire gang readout and two layer of strip readout [4] and a doublet has two layers of wire gang readout and two layer of strip readout.

Signals by hits are read out from wire gangs for the R position determination and from strips for ϕ positions, independently.

[4] There is no strip in the middle TGC of a triplet.

There are 24 modules in ϕ in the forward region and 48 modules in the end-cap region to make 2π coverage.

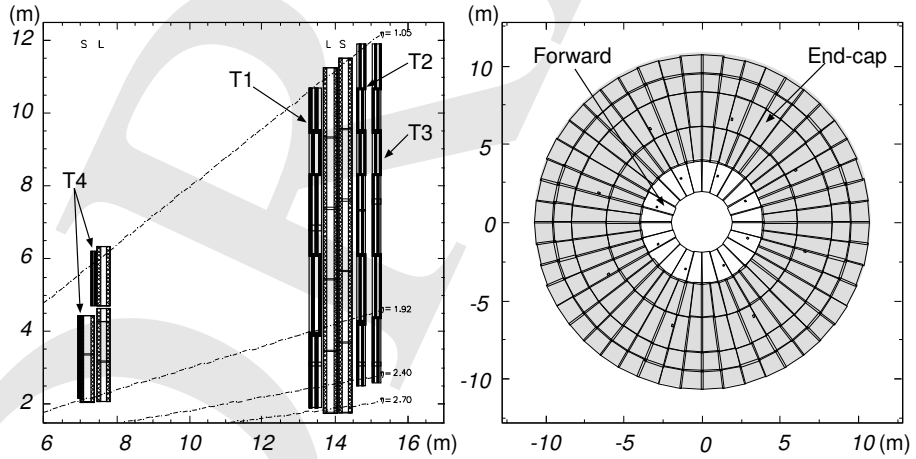
5.1 Digitization in R Direction

Hit positions of R coordinate in TGC can be read by gangs of wires. Figure 13 (a) shows the schematic view of wire gangs in a TGC. Number of wires to be ganged varies depending on TGC type and the position at which TGC is located. This is because all gangs of wires in the three stations(T1, T2 and T3) is aligned so as to point the interaction point. The detailed description of the ganging is described in amdb_simrec data base. TGC digitizer calculates the id number of the wire gang to which a hit belongs based on the database.

5.2 Digitization in ϕ Direction

Hit positions of ϕ coordinate in TGC can be read by strip. Figure 13 (b) shows the schematic view of strips in two readout layers in a TGC module. Every TGC module has two strip layers to measure ϕ coordinate. Each layer of a module has 32 strips. Each strip has a conical shape with respect to the beam axis. Thirty strips have the width of 1/29.5 of the ϕ coverage of a chamber which is $2\pi/24$ and $2\pi/48$[5] n the forward and end-cap regions, respectively, therefore actual width of the strips is $2\pi/24 \times 1/29.5 = 8.875 \times 10^{-3}$ radian

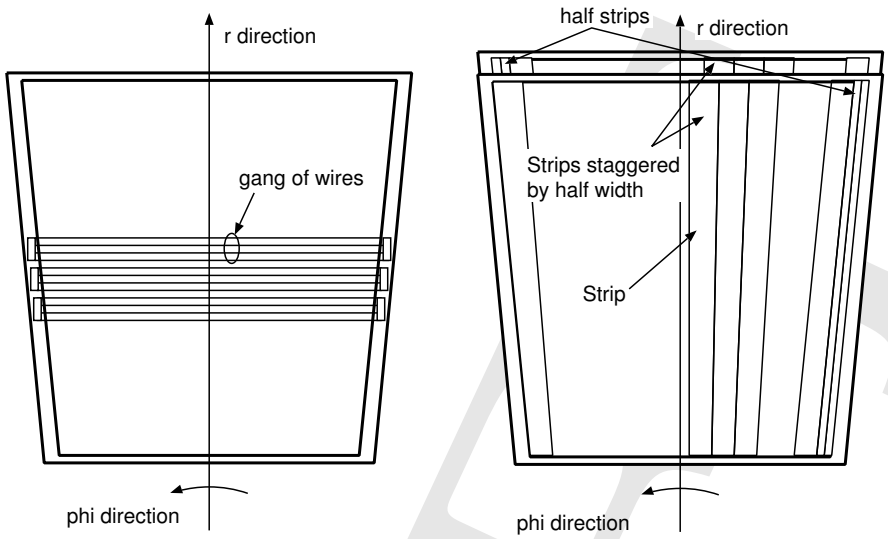
[5] i



(a) Longitudinal view of MDT and TGC, Inner and Middle Station

(b) R-phi view of TGC M1

Fig. 12. Layout of TGCs are shown: (a) longitudinal view of MDT and TGC in the inner and middle stations and (b) R-ϕ view of TGC M1 (T1F and T1E) station.



(a) Wire gang layout (b) Strip layout of TGC modules in A side

Fig. 13. Layout of TGC readout elements is shown: (a) wire gangs in R direction readout and (b) strips in ϕ direction readout.

for the forward chambers and $2\pi/48 \times 1/29.5 = 4.437 \times 10^{-3}$ radian for the end-cap chambers. The other two strips have half width of the other 30 strips.

In the A side, the two half-width strips are located at the edge of the smaller and larger ϕ for TGCs at smaller and larger $|z|$, respectively. On the other hand, in the C side, the two half-width strips are located at the edge of the smaller and larger ϕ for TGCs at larger and smaller $|z|$, respectively. This results in making non-mirror images between chambers in the A and C sides.

In the `amdb_simrec.P.03`, there is no description on strips in ϕ direction. Therefore, the digitization scheme is implemented in the code directly. For the `amdb_simrec.Q` or later, the detailed description on strips is written in amdb and the digitization uses the description. The modification of the digitization code is not needed when the strip configuration is changed.

5.3 Detector response

TGC digitization simulates the following detector responses:

- multi hits due to tracks passing several wire gangs or strips and induced charge spread on cathode plane which may make signals in several strips and
- intrinsic time response due to variation of strength of electric field in a sensitive layer which depends on injection angle of charged tracks and signal propagation along wires and strips,

- detection efficiency (sensitivity) of wire gangs and strips.

5.3.1 Multi hits by a single Charged Track

In case that charged particles pass through sensitive volumes, the particles are assumed to go straight and ionization occurs along their tracks. In case that the tracks spread over two wire gangs, those gangs are made to be fired. the multi hits on the strips is made by the charge spread on the cathode plane which is induced by the avalanche around anode wires. The size of the charge spread is parametrized by the data from test beams and simulations.

Multi hits on wire gangs are generated when a charged track passes the boundary between two wire gangs. The track path belongs to two wire gangs, the track makes two fired wire gangs.

In addition to the track path, multi this on the cathode strips are generated by induced charges spread over the cathode plane extending to two strips. Figure 14 shows the charge spread on the cathode surface. The radius on cathode surface is about a few mm for the surface resistance of $\sim 1\mathrm{M}\Omega/\square$. The radius in which the strip is fired can be set so as to fire according to the threshold value for the cathode signal. However, the default value set in the joboption file is 0cm.

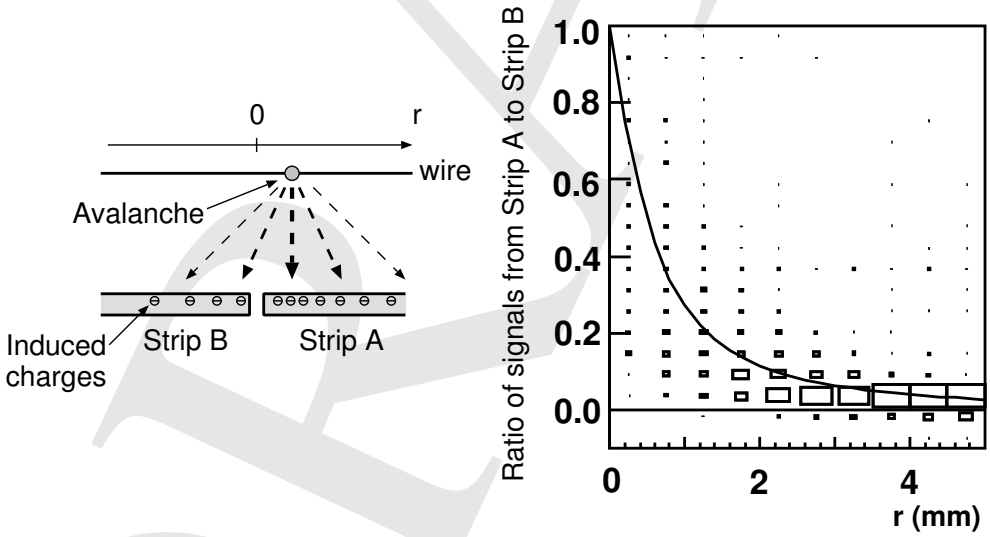


Fig. 14. Scatter plot of ratio of signals from two neigbouring strips as a function of positions in which avalanche is occurs. This result was gotten by a simulation. Surface resistance of cathode plane is set to $1\mathrm{M}\Omega/\square$. "r=0mm" means the boundary of two neighbouring strips. At the edge of the strip(r=0mm), the signals from two strips are same. Solid curve is a theoretical expectation.

In addition, the cross talks between signal channels originating from readout

electronics could be a source of multi hits. At the moment, this is not taken into account.

5.3.2 Time Response

The intrinsic time response is parametrized using a standalone simulation such as `GARFIELD` and the parametrization had been confirmed by test beams. The signal propagation time, the charge distribution and the detection efficiencies are set to the values gotten by test beams.

Figure 15 shows that an example of the distribution of time response of signals from wires. The response time depends on the incident angle of charged particles. Long tail for the angle of 0° is due to tracks passing through week electric fields which locates the middle points between two wires. Larger insistent angle gives shorter response time, this is because the particle can have larger possibility to pass the stronger electric field in which electrons can reach a wire in short time.

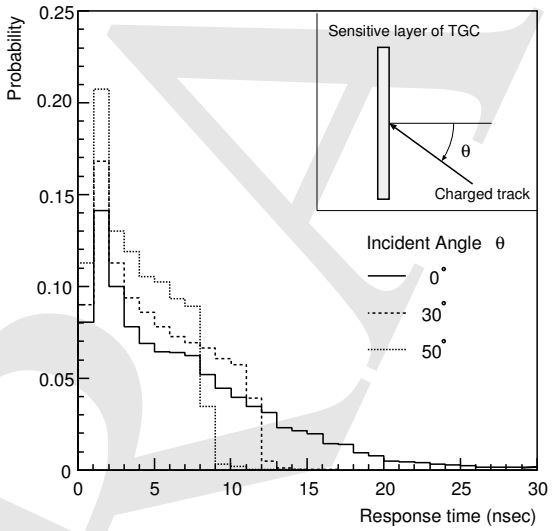


Fig. 15. Simulated time response distribution of TGC as a function of incident angles of charged tracks. Larger angle gives shorter response time.

In addition, the signal propagation time along a wire is set to 3.7ns/m and that along a strip is set to 8.5ns/m. Those values had been measured at test beams.

Time of flight to the chamber, response time and signal propagation time are summed up for the global time.

5.3.3 Detection Efficiency

At the moment, the detection efficiency of chambers are averaged over their sensitive areas. However, the efficiency map of each chamber have been measured by the test bench with cosmic rays, therefore the use of the map will be implemented eventually for realistic simulation.

5.3.4 Conversion of Digits to positions

The conversion of digits to global and local positions is performed by the method
(`TgcReadoutElement`) in the detector description, `MuonGeoModel`. `TgcReadoutElement` returns local position or global position when an id is given. For digits in the R direction, there is no information on ϕ direction, therefore, the ϕ position is represented by the centre position in ϕ of the TGC. On the contrary, for digits in the ϕ direction, no information on R direction exists ans this results in the R position is set to the centre position of TGC along R direction.

5.4 Classes, Methods and Properties

Two main functionalities of TGC digitizer are as follows:

- to create digits from GEANT4 hits
- to simulate detector response (timing, detection efficiency, multi signals by a single hit)

The TGC digitizer consists of two classes: `TGCDigitizer` and `TGC_Digitizer` because of the historical reason that the digitizer had been used in a standalone program and migrated to the Athena framework. `TGCDigitizer` is the class which has the entry point of TGC digitization. All functionality of TGC digitization is implemented to the class of `TGC_Digitizer` which is called by `TGCDigitizer`.

- `TGCDigitizer`
  - `initialize()` initializes the services, such as StoreGateSvc, PileUpMergeSvc and sets the parameters controlling the behavior of the TGC digitizer.
  - `execute()` reads GEANT4 hits from StoreGate in each of detector components corresponding to TGC modules which are triplets or doublets. A triplet has tree sensitive volumes and a double has two. This method calls the `TGC_Digitizer::executeDigi`, which digitizes every hit, for every readout element, i.e., a sensitive volume of a chamber.
  - `finalize()` only returns `StatusCode::SUCCESS`.
- `TGC_Digitizer`

- `initialize()` initializes `TgcHitIdHelper`, `TgcIdHelper` and random number of a stream for the digitization. In this method, `readFileOfTimeJitter()` reads and sets the parameters for intrinsic time response of TGC from `share/timejitter.dat`.
- `executeDigi()` digitizes hits. A single hit can be digitized in the two directions independently: radial and azimuthal directions. The information is restored by MuonGeoModel and accessible from the class of `TGCReadOutElement`. As described the preceding section, R direction is digitized based on the wire ganging information in the database. Digits in ϕ direction are calculated based on the formula in which the structure of strips in TGC is well expressed for `amdb_simrec.P.03`. For `amdb_simrec.Q` or later, ϕ direction is digitized based on the parameter in the database as the same manner of those in R direction.
  The method determines the response time for digits which is commonly used for signals from wire gangs and strips, and signal propagation time along wires and strips. In case that response time is outside of the time window to be set, that hit is removed. This method also removes some hits based on the detection efficiency to be set.
  TGC digits contains Muon ID only. The bunch crossing ID will be included in future, too.
- `readFileOfTimeJitter` reads the parameters of the intrinsic time response from the file `timejitter.dat` and stores them in `vector`.
- `timeJitter` calculates response time of a signal hit according to incident angle of a charged track based on the time response parameters.
- `efficiencyCheck` determines if a hit is detected or not.

TGC digitizer is controlled by the following parameters in a jobOptions file:

5.5 TGCDigitValidation

`TGCDigitValidation` has the functionalities as follows:

- to convert hit position from digits
- to fill ntuple with the converted positions and the position in MCtruth

The ntuple contains, Muon ID, local and global positions and timing of digits, etc., which are summarized in Table 7.

`TGCDigitValidation` is controlled by the parameters, shown in Table 8, in a jobOptions file:

Property	description	default value
SigmaChargeSpreadRadius	Radius of a signal spread in cathode plane in mm.	0.
Multihits	to allow multi hits in wire gangs or not in case that a track has large incident angle to TGC plane	FALSE
EfficiencyOfWireGangs	Detection efficiency of wire gangs	0.999
EfficiencyOfStrips	Detection efficiency of strips	0.999
InputObjectName	Name of input collection	TGC_Hits
OutputObjectName	Name of output collection	tgc_digits
WindowLowerOffset	Lower bound of time window in nsec	-75
WindowUpperOffset	Upper bound of time window in nsec	25

Table 6
Properties of TGCDigitizer and their default values

parameter name	description
nPar	Number of muons
nHits	Number of digits
stName	station name of a digit
stEta	station eta of a digit
stPhi	station phi of a digit
isStrip	type of a digit, 0 for a wire gang, 1 for a strip
gasGap	layer number of gas gap in which a digit is found
channel	channel number of a digit
tof	sum of Tof, response time and propagation time for a digit
gx, gy, gz	global position of a digit
lx, ly, lz	local position of a digit

Table 7
Contents of the ntuple

6 CSC_Digitization

A detailed description of the CSC detector in the ATLAS Muon Spectrometer can be found elsewhere [**?**]. The digitization in the CSC is the simulation of

parameter name	description
`DoTGCTest`	to do TGC validation or not
`DumpTrackRecord`	to dump track record or not
`NtupleLocID`	to set location and ID number of ntuple

Table 8
Parameters for control of TGCDigitValidation

the charge distribution on the CSC cathode strips given a hit in the sensitive gas. The process also identifies the the strips numbers and their orientations. Thus after processing all the hits in the event, the CSC digitization outputs the list of digits into the transient event store from where they could be picked up by other algorithms. The object referred to as a CSC digit is nothing more than the compact identifier of a strip together with the charge on that strip. In the digit, the charge is given in the number of equivalent electrons.

An incident particle, traversing the CSC gas, may produce primary and secondary ionisation electrons, leading the formation of an avalanche on the anode wire as illustrated in Figure 16. The induced charge distribution on the

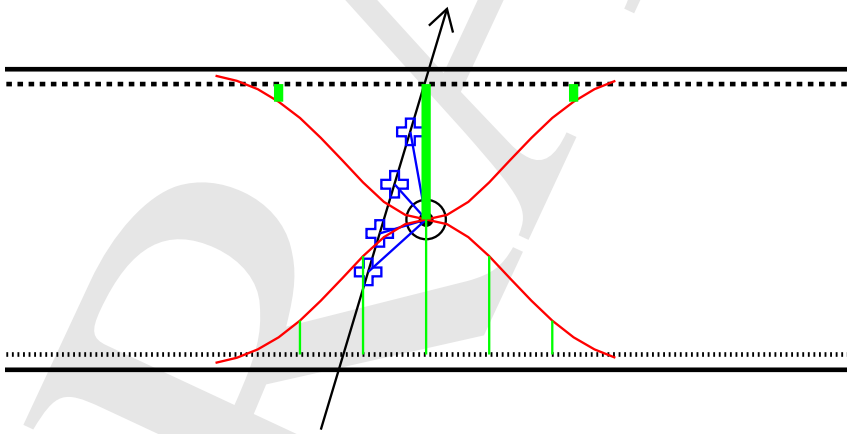


Fig. 16. *Avalanche production in the CSC gas: the incident particle produces primary and secondary electrons along its path, leading to a charge multiplication and collection at the anode wire. This induces a distribution on the cathodes, shared equally between the two segmented cathod strips. The process of CSC digitization consists of finding the charge distribution of the cathode strips given of the collection of hit objects in the gas.*

segmented cathode can be written down as follows:

$$\Gamma(\lambda) = K1\frac{1 - \arctan(K_2\lambda)}{1 + K_3 \arctan(K_2\lambda)} \tag{2}$$

where the constanst K_2 and K_3 are related empirically as

$$K_2 = \frac{\pi}{2}\left(1\frac{1}{2}K_3^{1/2}\right) \tag{3}$$

and $\lambda = x/d$ with x being the precision coordinate and d the anode-cathode spacing, which, in the CSC, is equal to the anode wire pitch. By requiring that the induced charge is distributed equally on each of the two cathodes, and using the relation 3, the charge distribution of Equation 2 reduces to a one-parameter expression [**?**,**?**,**?**,**?**].

For a given simulated hit in the sensitive gas, the number of interactions along the incident particle path is obtained according the Poisson distribution, with a variable mean value determined from test beam studies. For each interaction, the number of primary electrons is obtained according to a probability density function also determined form test beam: this distribution closely follows the Landau energy loss distribution in a thin absorber, with a small probability of large tail as shown in Figure **??**. Since in the CSC, the precision coordinate is determined by a relative measurement of the induced charge on the adjacent strips, variations of the order 20% or less in the gas gain do not affect the spatial resolution. This means that the performance of the CSC is hardly affected by temperature and pressure variations. Furthermore, CSC performance is not affected by the drift time properties of the gas since no timing measurement in the determination of the precision coordinate. The spatial resolution is affected by the noise in the amplifier. The resolution on the centroid of the charge distribution depends linearly on the signal to noise ratio. Other factors, such the electronic gain calibration, the geometrical cathode deformation, contribute to limit the achievable precision of the position determination.

6.1 CSCDigitValidation

7 Conclusions

Four different packages have been developed for the digitization of MDTS, RPCs, TGCs and CSCs. These packages provide the construction of the Muon simulated digit collection starting from the hit collections which are the output of the simulation.

8 Acknowledgements

The authors would like to thank Davide Costanzo for his wise and constant supervision and for providing the pile-up infrastructure (together with Paolo Calafiura). A special thank goes to Alessandro De Salvo for providing the first structure of the MDT_Digitization algorithm which uses the external rt relation. The authors are indebted to him and to Ludovico Pontecorvo for providing realistic rt relations, from Garfield simulation and from data analysis.

References

[1] MuonDigitization, in the ATLAS CVS repository (offline/MuonSpectrometer/MuonDigitization).

[2] XXXSensitiveDetector classes, in the ATLAS CVS repository (offline/MuonSpectrometer/MuonG4/MuonG4SD).

[3] Muon Hit and SimID Helpers classes, in the ATLAS CVS repository (offline/MuonSpectrometer/MuonSimEvent).

[4] The HitHelper base class, in the ATLAS CVS repository (offline/Simulation/HitManagement).

[5] The MuonSimData class for MCTruth deposit, in the ATLAS CVS repository (offline/MuonSpectrometer/MuonSimData).

[6] Global ATLAS Diitization, in the ATLAS CVS repository (offline/Simulation/Digitization).

[7] Muon Offline Identifier Helpers, in the ATLAS CVS repository (offline/MuonSpectrometer/MuonIdHelpers).

[8] Muon Offline Identifier Helpers, in the ATLAS CVS repository (offline/DetectorDescription/IdDictParser).

[9] Muon Digit classes, in the ATLAS CVS repository (offline/MuonSpectrometer/MuonDigitContainer).

[10] MuonGeoModel raw and read-out geometry, in the ATLAS CVS repository (offline/MuonSpectrometer/MuonGeoModel).

[11] A. DiCiaccio *et al.*, "*Hierarchical Software Identifier Scheme for the ATLAS Muon Spectrometer*", ATLAS Internal Note, ATL-MUON-2000-020 (2000).