Preprint typeset in JHEP style - HYPER VERSION

GEANT4 Muon Digitization in the ATHENA Framework

D. Rebuzzi*
INFN, Sezione di Pavia and Dipartimento di Fisica Nucleare e Teorica
Università di Pavia, Via A.Bassi 6, I-27100 Pavia, Italy
Email: daniela.rebuzzi@pv.infn.it

K. A. Assamagan
Brookhaven National Laboratory (BNL)
Department of Physics, Upton , NY 11793, USA
Email: ketevi@bnl.gov

A. Di Simone
CERN PH-ATC, 1211 Geneva 23, Switzerland, and INFN-CNAF, Italy
Email: andrea.di.simone@cern.ch

Y. Hasegawa
Department of Physics, Shinshu University,
3-1-1 Asahi, Matsumoto 390-8621 Nagano, Japan
Email: hasegawa@azusa.shinshu-u.ac.jp

N. Van Eldik
National Institute for Nuclear Physics and High Energy Physics (NIKHEF)
Kruislaan 409, 1098 SJ Amsterdam, The Netherlands
Email: nveldik@nikhef.nl

ATL-SOFT-PUB-2007-001
05 March 2007

ABSTRACT: The aim of this note is to describe the Muon Digitization software packages, completely re-written to run in the ATHENA framework and to interface with the GEANT4 Muon Spectrometer simulation.
The Muon Digitization is the simulation of the Raw Data Objects (RDO), or the electronic output, of the Muon Spectrometer. It consists of two steps: in the first step, the output of the detector simulation, the Muon Hits, is converted to Muon Digits, namely intermediate objects that can be fed into the reconstruction. In the second step, the Muon Digits are converted into RDO, the transient representation of raw data byte stream.
In this paper we will describe the detailed implementation of the first step of the Muon Digitization, where the detector simulation output is "digitized" into Muon Digits. We will describe the fundamentals of the Muon Digitization algorithms, outlining their global structure, with some emphasis on the simulation of piled-up events. We will also describe the details of the digitization validation procedures against the Monte Carlo information.

*Corresponding author.

Contents

1. Introduction

The present note describes the Muon Digitization software, recently re-written to run in the ATHENA framework and to operate on the GEANT4 Hits.

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, simulation, 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 three base classes: `DataObject, Algorithm, Converter, AlgTool, Services`.

Everything which is essentially a procedure, i.e., a set of rules for performing transformations on 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 *jobOptions* files. In ATHENA, data objects produced by algorithms are saved in a common transient memory area from where they can be accessed by other modules, to 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 in the ATHENA sense described above, one for each muon technology[1]. The Muon Digitization produces collections of simulated Muon Digits out of Muon Hit collections from the GEANT4 detector simulation. The input hit collections are read in from the TDS and the output digit collections are written back in to the TDS.

In the present architecture, Muon Digits are converted to Raw Data Objects (RDOs) or to byte streams before undergoing the persistent storage. RDOs are the output of the readout electronic, thus are the constituents of (and the transient representation of) the byte stream.

The goal of the Muon Digitization is to simulate the output signal of the muon detector in ATLAS starting from the output of GEANT4 detector simulation. The digitization process consists of two steps: in first, the output of the detector simulation, henceforth referred to as GEANT4 Hits, is converted to Muon Digits. In the second step, the Muon

[1]The Muon Spectrometer consists of four different technologies, two for the Precision Chambers (Monitored Drift Tube Chambers, MDTs, and Cathode Strip Chambers, CSCs) and two for the trigger system (Resistive Plate Chambers, RPCs, and Thin Gap Chambers, TGCs). For details see [**?**]

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 for debugging and validation purposes.

The muon off-line reconstruction data flow starts from the raw data, the byte stream or the persistified RDOs in POOL [1]. The byte stream (or the persistified RDO) is read into the TDS as transient RDO. In a subsequent steps, the RDOs are converted into Reconstruction Input Objects (RIOs), also known as PrepRawData or clusters of PrepRawData, through calibration services, pedestal and noise handling, etc. The step from RDO to RIO, is where the RIO is "prepared" before being fed into the off-line reconstruction.

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 Muon Digits. The second step, which converts the digits in the RDO (byte stream) is implemented in separate algorithms and documented in [3]. 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 four digitization algorithms (one for each Muon technology) and their validation against the information from the Monte Carlo Truth.

2. Muon Digitization

As already outlined, 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. This procedure occurs separately for each Muon technology.

The Muon Digitization code is part of the ATHENA software since release 7.4.0. The code is in the ATLAS CVS repository under [4]. Besides the four digitization algorithms, the Muon Digitization container houses a package, MuonDigiExample, which contains monitoring algorithms (ReadMuonSimHits and ReadMuonDigits, to check the correct hits/digit writing to SG), ntuple makers, and the four digit validation algorithms (XXXDigitValidation, where XXX = MDT, RPC, TGC, CSC) which allow to check the main digit parameters against the Monte Carlo Truth information. A Muon Digitization jobOptions fragment is included in the global ATLAS Digitization jobOptions (AtlasDigitization.py in [5]) 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.

The Muon Digitization has been designed to be independent from the GEANT4 simulation. It relies only on the read-out geometry (provided by MuonGeoModel [6]) and the detector-specific Muon Spectrometer Offline Identifier (OID) scheme [7].

2.1 Muon Hits

Hit production in GEANT4 is provided by *Sensitive Detectors*. Muon hits are generated by the XXXSensitiveDetector classes [8] (where XXX = MDT, RPC, TGC, CSC) when

charged particles cross the sensitive parts 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 [9].

Each hit is labeled 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 [9]. The base class HitIdHelper [10] 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 indexes relevant for each technology information.

The SimIDs in principle do not coincide with the OIDs associated to the digit objects. 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 not depend on it. Therefore, *the event is completely decoupled from the geometry*, the only communication occurring via the identifiers.

Hits have a very light content. This is mainly due to the fact that all the geometrical information is encapsulated in the SimIDs. Thanks to the retrieving mechanism mentioned above, the SimIDs are enough to access 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 hit contains the quantity to be digitized. In Table 1, the contents of the SimIDs and of the Muon hit objects are listed 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 (no sorting is performed at the simulation level). There are therefore four independent hit collections in the Muon Spectrometer per event, one for each 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 the output of the first step of the digitization procedure. They resemble the detector output and are basically defined by the reconstruction group. Muon Digits are labeled by an offline identifier (OID) which create the connection to the reconstruction.

	MDT	RPC	TGC	CSC
SimID	StationName, PhiSector, ZSector			
	MultiLayer	DoubletZ	GasGap	ChamberLayer
	Layer	DoubletR		WireLayer
	Tube	GasGapLayer		
		DoubletPhi		
		MeasuresPhi		
Muon Hit	SimID	SimID	SimID	SimID
	globalTime	globalTime	globalTime	HitStart
	driftRadius	localPos	localPos	HitEnd
	localPos	(trackNumber)	dirCos	partID
	(trackNumber)		(trackNumber)	(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. Besides the SimID, the Muon Hits contain additional information, listed here, which is required by the digitization and the pile-up procedures.*

This contains the digit geometry information packed according to the classes in [11]. OIDs are initialized using the identifier dictionary [12], 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 geometrical description of the detector elements is obtained from the methods of the XxxDetectorElement classes of MuonGeoModel *read-out* geometry [13]. They allow to get information (like the hit distance from the RO chamber side) needed by the Digitizers to construct 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.

The Muon Digit classes are located on cvs under [14]. In Table 2, the contents of the OIDs, of the Muon Digits and of the RDOs are shown for the four Muon technologies.

2.3 Infrastructure for Event Pile-Up

In addition of handling hits coming from a single bunch crossing, the digitization, as implemented at the time of this writing, is also 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 [5], 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

	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	BC Tag	charge
	(ADC count)	globalTime		hit time
RDO	fired tube	fired channel	raw hit or	N ADC samplings +
		of CM	raw coincidence	strip address

Table 2: *Contents of the Offline Identifiers (OIDs), of the Muon digit objects and of the Muon RDOs for each Muon technology. The first four fields of the OID specify the name of the station where the Digit 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 plus additional information, listed here, required by the reconstruction. For the CSC, the number N of ADC samplings may be variable; however the default is $N = 4$.*

function to return the GEANT4 hit time. Furthermore, some tools also exists to link the overlaid hits to 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 association to point the merged Monte Carlo collections.

Simulated GEANT4 hits persistified using POOL can be read in together with previously generated minimum bias and cavern background events. The hit overlay and sorting according to the given detector elements is then carried out as described above, and the digitization proceeds afterwards.

2.4 Association to Monte Carlo Truth

In the second step of the digitization procedure, Muon Digits are converted to muon RDOs, the transient representation of the byte stream. The production of the RDOs 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 RDOs 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 the original simulated particles at the digit or RDO level. The recorded object in question is a map of muon off-line identifiers to MuonSimData objects [15]. 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 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 shown in Table 1. As stressed before, the digitization procedure should convert the hit information from the GEANT4 simulation into an output which should resemble the output signal of the ATLAS detector. The MDT_Digitization offers the infrastructure for the MDT Digit building out of any *valid*[2] 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. 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. 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.

Starting from the impact parameter, the `driftRadius` of Table 1, associated to the MDTSimHit, the MDT_Digitization performs several tasks:

1. conversion of the drift radius into a drift time
2. calculation of the time structure of the event

[2]In the sense specified in Section 3.4.

3. trigger match

4. 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-consuming $r \rightarrow t$ 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 a fixed number for all digits. The two tools are described in Sections 3.2 and 3.3 respectively.

3.1 Class methods and properties

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 TimedHitPtrCollection 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 `create Digits()` 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 (OID) 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 retrieving the distance to the chamber read out (RO) 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

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.*

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 (see later), the hit time is converted into a TDC count. A detailed discussion of the time structure of the event is given in Section 3.4. 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 default values.

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

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

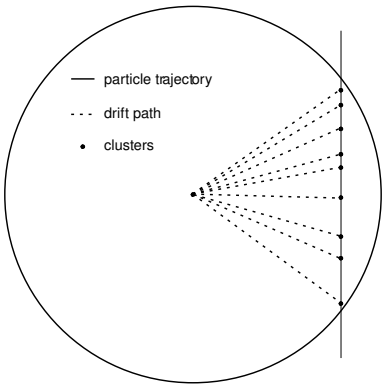

Figure 1: *Trajectory as used in the simulation. The dots represent the clusters generated along the trajectory, the dotted lines represent the propagation of the clusters to the wire.*

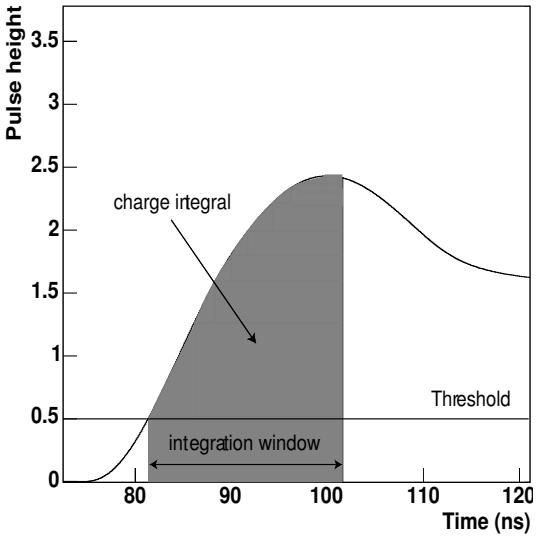

Figure 2: *Determination of the time at threshold and the successive integration of the charge.*

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 [20] 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 major contributions to the tube resolution. In addition, two other effects can be simulated (they are not switched on by default, 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 $e^{-d_{ro}/l_{att}}$, 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 negligible for chambers with 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, this resolution 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.

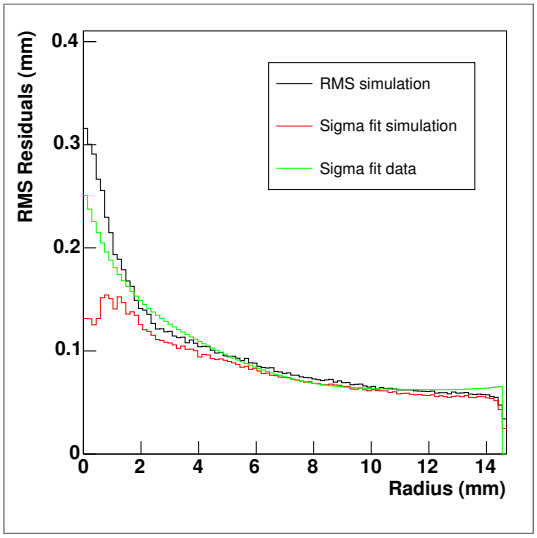

Figure 3: *Intrinsic simulated spatial resolution.*

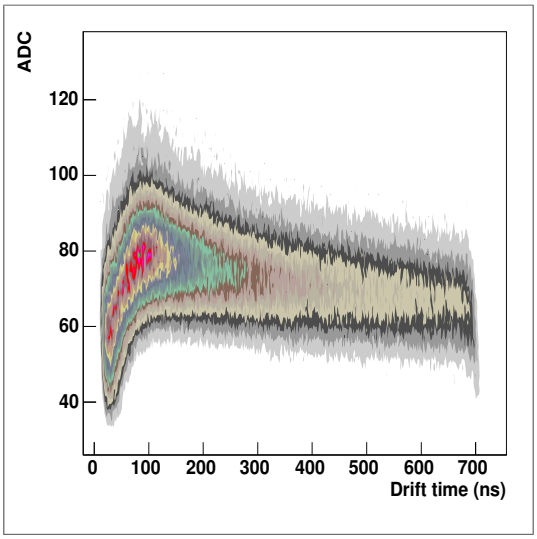

Figure 4: *ADC vs drift time.*

3.3 RT_Relation_DigiTool

This digitization tool relies on an external ASCII file (from GARFIELD [20] simulation or from the data analized with an autocalibration program) which should have a list of r, t and the corresponding 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.

3.4 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:

$$t_{tot} = t_{tof} + t_{bunch} + t_{prop} + t_{delay} + t_{drift} \tag{3.1}$$

The time of flight t_{tof} is obtained from the MDTSimHit in form of `global time`. 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

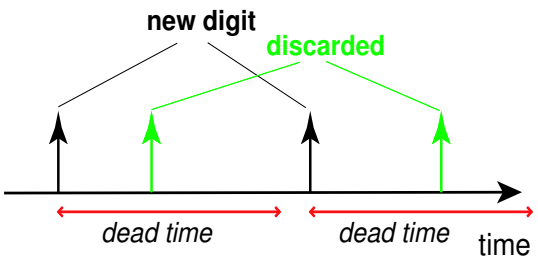

Figure 5: *Dead time.*

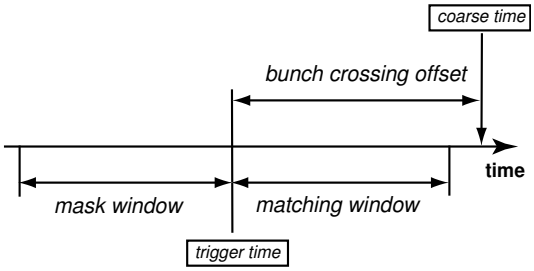

Figure 6: *Time windows.*

a time stamp every time the input signal passes the threshold. To avoid non physical hit proliferation, the TDCs have a programmable dead time which is set to the maximum drift time of the tube [16] 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;
- 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 [16]:

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

Where t_{fix} is a predefined fix dead time which can be set via the jobOptions (default = 700 ns), 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 [17]). 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 (3.1) 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 size of the two windows it is possible to have *more than 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 (3.1), smeared by a Gaussian distribution whose resolution can be selected via jobOptions (set by default to 0.5 ns). The TDC count is stored into the MDT digit object. The TDC count (3.1) 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 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).

3.5 MDTDigitValidation

The XXXDigitValidation algorithms have been implemented in order to check the correctness of the digit production and of the digitization processes. The general method consists of comparing known ("true") track position with associated digit position (from MuonGeoModel) and to study the residual distributions. The Truth information is coming from the MuonSimData (as discussed in Paragraph 2.4), a separate object which can be stored together with the RawData, which maintains the link to the hits at the Digit/RDO level. The validation is performed generating single muon events in the barrel, with no physics processes activated but transport. The output of each validation algorithm is a Ntuple which contains the main Digit/Truth parameters. The XXXDigitValidation algorithms are part of the RunTimeTest (RTT) [18] ATHENA nightly control.

The MDTDigitValidation performs a calculation of the residual $(r_{calc} - r_{true})$, where r_{true} is coming from the MuonSimData hit information and r_{calc} from the conversion of the drift time of the digit into a drift radius using the rt-relation. In the MDTDigitValidation ntuple, many parameters from the truth information at the entrance of the Muon Spectrometer (from MSEL, Muon Spectrometer Entry Layer) and from the HepMC::GenVertex of the muon tracks are also stored, for any kind of additional validation. Figure 7 shows one of the validation plots for the MDT, included in the RTT.

4. RPC_Digitization

RPC hits are generated by the RPCSensitiveDetector (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 into an 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

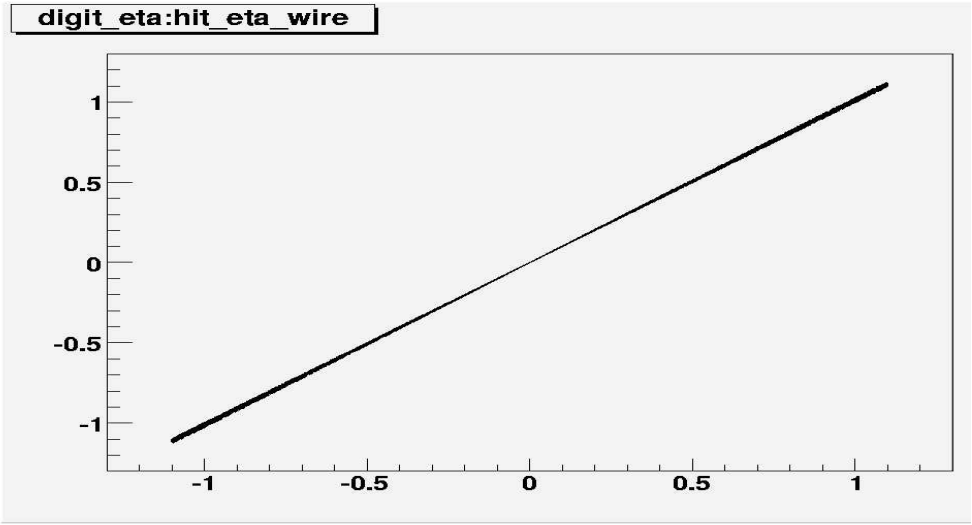

Figure 7: *Eta from the digits as a function of eta of the hits.*

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.*

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 one, 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 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 9 shows, for example, the probability to observe a cluster of size one 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 one or two, i.e. for each bin, the complement to unity gives the probability to have a cluster with size

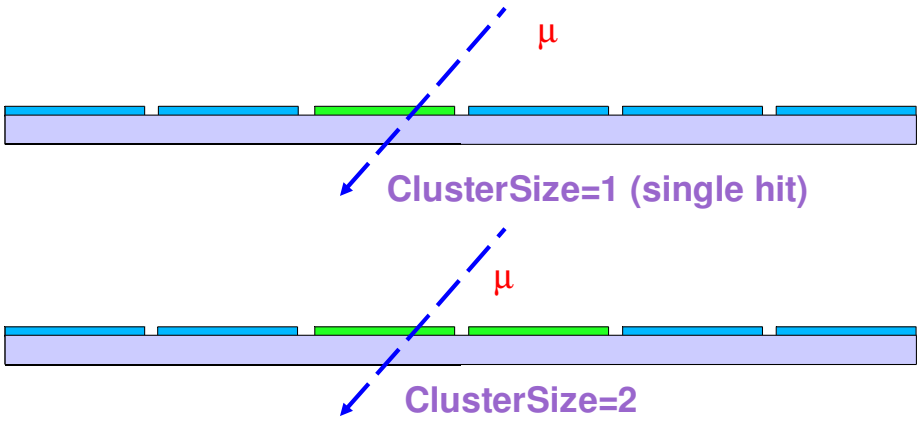

Figure 8: *RPC cluster formations and consequent size.*

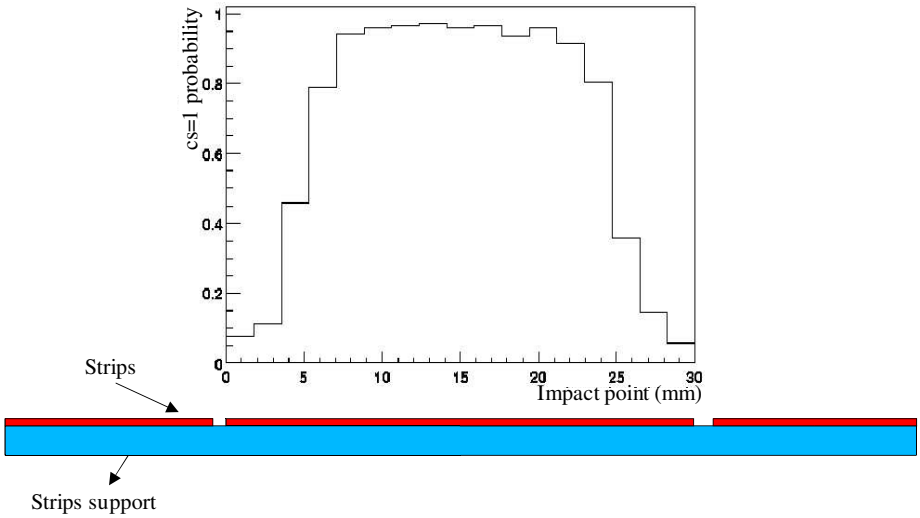

Figure 9: *Probability to observe cluster size one as a function of the impact point along the strip.*

two. 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 two, whereas a track passing in the middle of one strip would induce signals on that strip only (see for instance Figure 8). In a small amount of cases, clusters with sizes greater than two 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:

1. experimental distributions are used to decide, according to the impact point of the particle along the strip, whether the cluster size will be one or two

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.

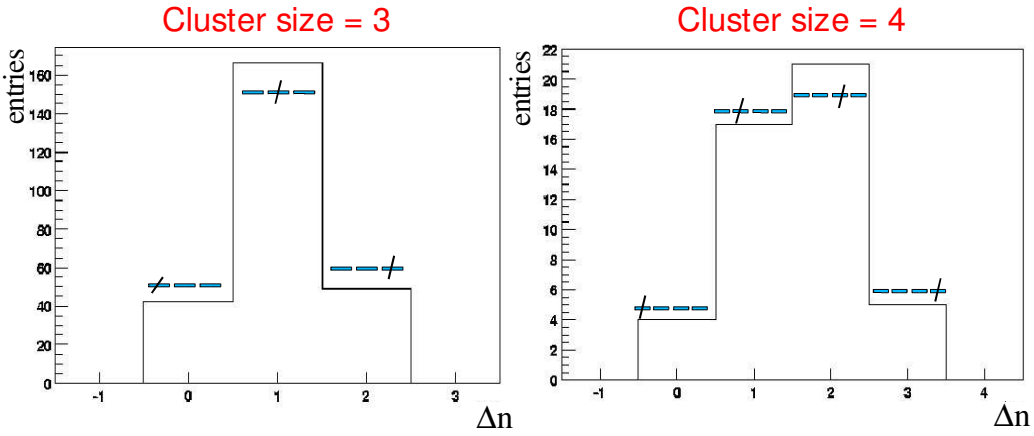

Figure 10: *Experimental cluster spread distributions for cluster of sizes three and four.*

Particular attention needs to be paid to the way the additional digits are created around the one actually crossed by the muon. Figure 10 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 shown. 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 content 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 one 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 `PropagationTime` method.
In the hit collections coming from the RPCSensitiveDetector, it sometimes happen that many hits, very close both in space and time, are produced by the same crossing particle. 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, this sets the dead time. Every hit from the same strip falling within the dead time is ignored.
Table 5 shows the list of the properties of the RPC_Digitizer algorithm and their default values.

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 the RPC_Digitizer and their default values.*

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 transport. 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 Monte-Carlo information. With no physics processes activated, this direction 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 11 the η and φ directions of muons not producing any RPC Digit are plotted. These inefficiencies of the RPC system are concentrated, as expected, in regions not instrumented with RPCs:

- the feet of the ATLAS detector

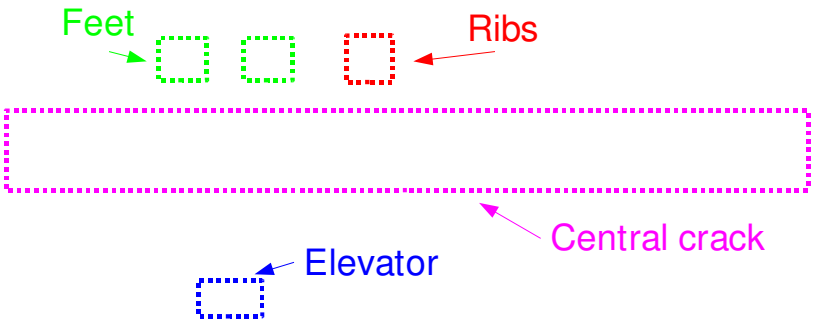

Figure 11: *RPC digitization validation. Scatter plot of η and ϕ directions of muons not producing any RPC Digit. The inefficiency regions correspond to Muon Spectrometer areas not instrumented with RPCs.*

- 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 did not show any abnormal inefficient region. With the same procedure, it is possible to look for digits which are created in the wrong place. Figure 12 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 investigation allowed to find an error in the `RPCSensitiveDetector` which assigned wrong SimID to the RPC hits in BMF stations.

5. TGC_Digitization

Longitudinal and r-ϕ views of the layout of the Thin Gap Chamber(TGC) system are shown in Figure 13(a) and (b), respectively. The system consists of one layer of triplet modules and two layers of doublet modules in the middle station ($|z| \sim$ 13 - 15m) and one layer of doublet modules in the inner station($|z| \sim$7 - 8m). The triplet and doublet

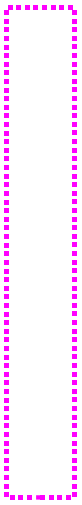

Figure 12: *Wrong RPC digits in BMF chambers.*

modules are assembled with three and two TGCs, respectively. Each TGC measures the two coordinates in the r and ϕ projections.

The system covers a η range from 1.05 to 2.40 which is divided into two subregions; end-cap ($1.05 \leq |\eta| \leq 1.92$) and forward ($1.92 \leq |\eta| \leq 2.40$) regions where the modules are

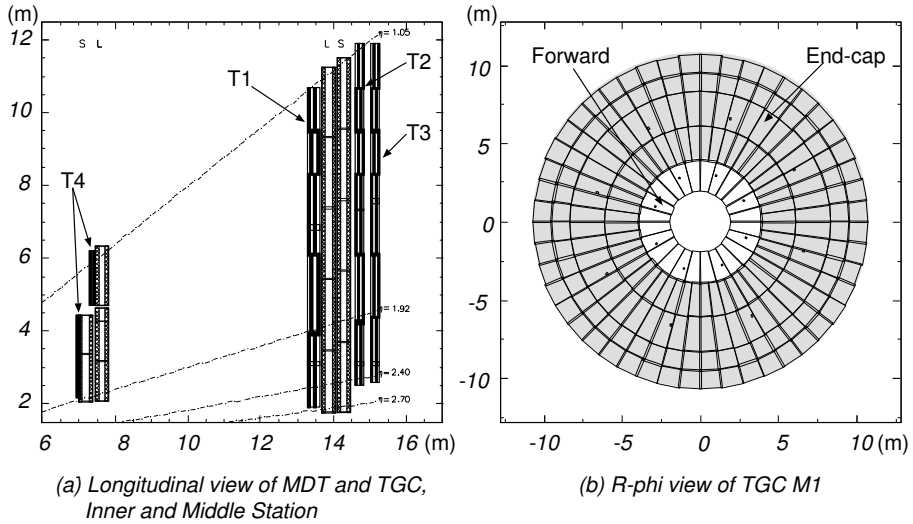

Figure 13: *Layout of the TGC system are shown: (a) longitudinal view of MDT and TGC in the inner and middle stations and (b) r-ϕ view of the layout of TGC T1 triplet modules, where each module is indicated in trapezoidal shape.*

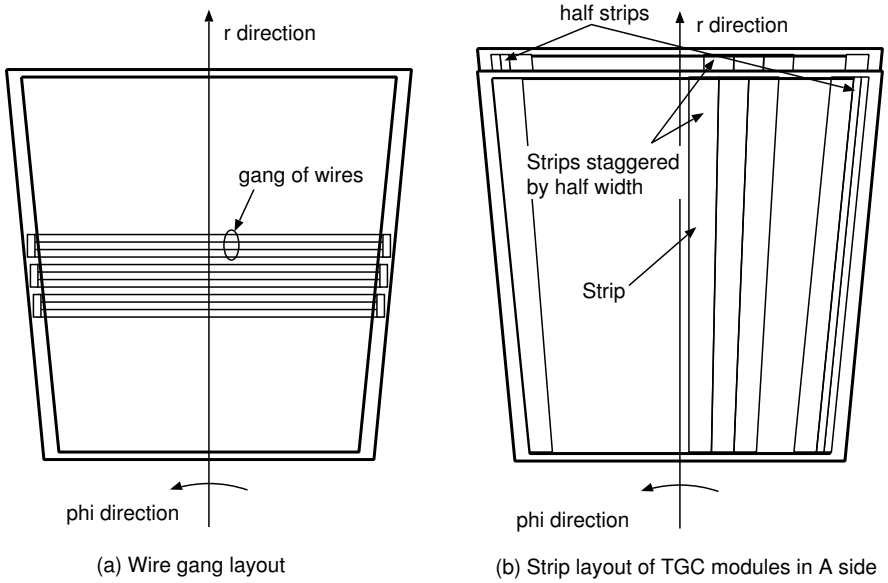

Figure 14: *Layouts of (a) ganged wires and (b) strips are shown. Ganged wires and strips are used to measure r and ϕ position of hits, respectively.*

mounted concentrically. In order to identify the location of a TGC module, each module is labeled as follows: T1F, T1E for triplets and T2F, T2E, T3F, T3E for doublets in the middle station, and T4F, T4E for doublets in the inner station, where the last character in the labels denotes a subregion.

The main functionalities of TGC digitizer are the following:

- to create digits from GEANT4 hits,
- to simulate detector response (timing, efficiency and multi-signals by a single hit), and
- to add a bunch-crossing (BC) tag to a digit.

5.1 Digitization of Hits

Each TGC has independent binary readouts in r and ϕ projections. Hit positions in r direction are measured by gangs of wires. Figure 14 (a) shows a schematic view of ganged wires in a TGC. The Level-1 muon trigger logic for TGCs requires that the number of wires to be ganged varies in a range of 4 to 26 according to the location of a TGC module. A detailed description of the ganging is described in amdb_simrec database [19].

Hit positions in ϕ direction are measured by strips. Figure 14 (b) shows a schematic view of a strip layout of two TGC layers of a module. Every triplet and doublet module has two strip layers to measure the ϕ position of hits. Each layer has 30 full-width strips and 2 half-width strips. The trigger logic requires that strips of the two layers are arranged in staggered layout.

Since ganged wires and strips have binary readouts, the TGC digitizer determines the ID numbers of ganged wires and strips from hit positions simulated by GEANT4.

5.2 Detector Response

TGC digitizer simulates the following detector response:

- ionization along a path of a charged track,
- induced charges on a cathode plane by an avalanche around an anode wire,
- time response and signal propagation time along wires and strips, and
- detection efficiency of wires and strips.

5.2.1 Ionization Along the Path of Charged Tracks

GEANT4 does not simulate detailed ionization process, therefore the TGC digitizer takes into account ionization along the path of a charged track. As the result, in the case that a particle passes through a region where electrons due to ionization are collected by more than two gangs of wires, those gangs output hit signals.

5.2.2 Induced Charges on Cathode Plane

The TGC digitizer simulates charges induced by an avalanche around an anode wire, which spreads on a cathode plane. Figure 15 illustrates the induced charges picked up by strips. The radius on the cathode plane is about a few mm for the surface resistance of $\sim 1\mathrm{M}\Omega$/square. In the case that strips fall in the region where the induced charges exceed the threshold value, those strips output signals.

5.2.3 Time Response

The intrinsic response time of a TGC is parametrized using GARFIELD as a function of the incident angle of a charged track. The parametrization has been confirmed by test beams. Figure 16 shows an example of distributions of time response of signals from wires. The response time depends on the incident angle of a charged particle. The long tail seen in case of angle 0° is due to tracks passing through the region of weak electric field located at the middle between adjacent wires. A larger incident angle gives shorter response time, since a particle with large incident angle traverses a region of strong electric field where electrons can reach to an anode wire in shorter time.

Signal propagation times along a wire and a strip has been measured at test beams, and set to 3.7ns/m and 8.5ns/m, respectively, in the TGC digitizer.

5.2.4 Detection Efficiency

In order to take into account the detection efficiency, the TGC digitizer removes digits so as to get suitable efficiencies for wires and strips. We assume that the efficiencies for wires and strips are independent of positions in a TGC, i.e., constant everywhere. We set the values for the wires and strips to those measured at testbeams.

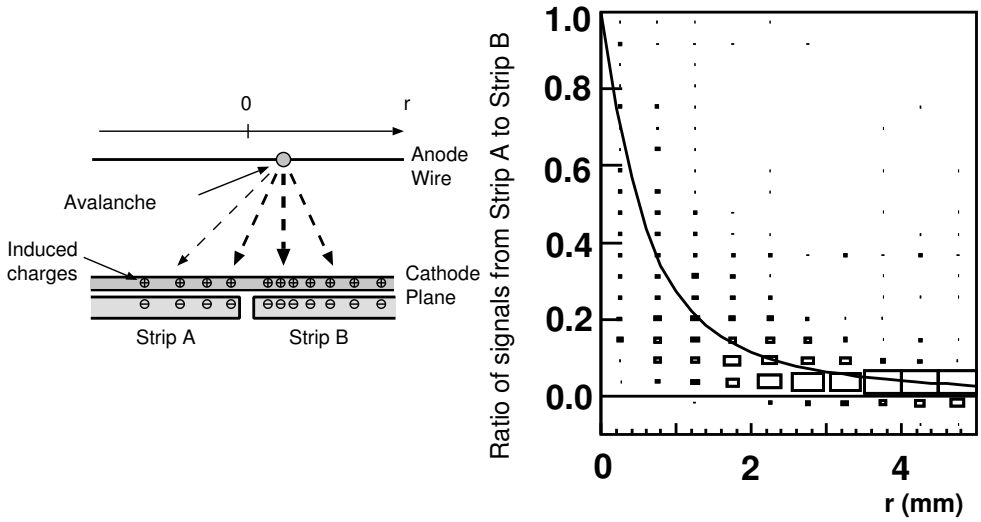

Figure 15: *Scatter plot of ratio of signals from two neighboring strips as a function of positions in which the avalanche occurs. This result was obtained by a simulation. The surface resistance of cathode plane is set to 1MΩ/square. "r = 0 mm" means the boundary of two neighboring strips. At the edge of the strip(r = 0 mm), the signals from two strips are same. The solid curve represents the theoretical expectation.*

5.3 Bunch Crossing Tag

A BC tag is assigned to each digit according to the time of flight (ToF) of hits simulated by GEANT4 and the response time simulated by the TGC digitizer. The BC tagging is performed based on a sum of the ToF and the response time. In order to reduce the

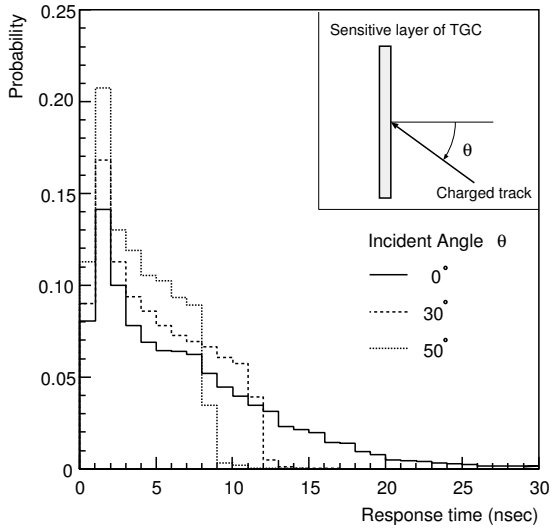

Figure 16: *Simulated time response distribution of TGC as a function of incident angles of charged tracks. Larger angle gives shorter response time.*

fluctuation of the ToF due to the length of a track path from the interaction point, a module-by-module correction is done by subtracting the ToF of a particle with infinite momentum passing through the inner most position of the module. After the correction, the fluctuation of the corrected ToF is less than 3 ns.

There are three time windows of previous, current and next bunches. In the case that the corrected time of a hit is outside of the three time windows, no digit is made out of the hit. In order to avoid inefficiency of triggering, the trigger logic requires that adjacent trigger windows are overlapped and thus a single hit may produce two digits with different BC tags.

5.4 Class methods and properties

The TGC digitizer consists of two classes: `TGCDigitizer` and `TGC_Digitizer`. `TGC_Digitizer` is called by `TGCDigitizer` and includes all methods concerning the TGC digitization.

- `TGCDigitizer`
  - `initialize()` initializes the services, such as StoreGateSvc, PileUpMergeSvc and sets the parameters controlling the behavior of the TGC digitizer.
  - `execute()` reads GEANT4 hits (positions, ToF and incident angles) from TDS (StoreGate) and writes digits from TGC digitizer (IDs of readout gangs of wires and strips and a bunch crossing tag) for each hit to TDS. This method calls the `TGC_Digitizer:: executeDigi` performs the digitization of hits of TGCs.
  - `finalize()` only returns `StatusCode::SUCCESS`.
- `TGC_Digitizer`
  - `initialize()` initializes `TgcHitIdHelper`, `TgcIdHelper`, random number of a stream and calls `readFileOfTimeJitter()`.
  - `executeDigi()` digitizes hits. A single hit is digitized in r and ϕ directions independently. A digit has a Muon ID and a BC tag.
  - `readFileOfTimeJitter` reads the parameters of the intrinsic time response from the file `timejitter.dat` and stores them in vectors.
  - `timeJitter` calculates response time based on the time response parameters taking into account incident angle of a charged track.
  - `efficiencyCheck` removes hits in order to get the nominal detection efficiencies for the wires and the strips.
  - `bcTagging` assigns a bunch-crossing tag to each digit. There are three kinds of BC tags; previous, current and next bunch. The assignment is based on the properties of `WindowOffsetWire`, `WindowOffsetStrip`, `WindowWire`, `WindowStrip`.

The `TGC_Digitizer` is controlled by the properties listed in Table 5.4 and modifiable via jobOptions file.

Property	description	default value
SigmaChargeSpreadRadius	Radius, in mm, of a region where charges are induced and strips output signals in mm.	`0.`
Multihits	to simulate ionization along a track path or not for debugging	FALSE
EfficiencyOfWireGangs	Detection efficiency of ganged wires	0.999
EfficiencyOfStrips	Detection efficiency of strips	0.999
InputObjectName	Name of input collection	TGC_Hits
OutputObjectName	Name of output collection	tgc_digits
WindowOffsetWire	Offset of time window for wire hits in nsec	0
WindowOffsetStrip	Offset of time window for strip hits in nsec	0
WindowWire	Width of time window for wire hits in nsec	40
WindowStrip	Width of time window for strip hits in nsec	50

Table 6: *Properties of TGCDigitizer and their default values.*

5.5 TGCDigitValidation

The `TGCDigitValidation` algorithm performs a validation of `TGC_Digitization` by means of the following functionalities:

- convert hit position from digits
- fill the 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 changeable via a jobOptions file.

Figure 17 shows channel number as a function of r or ϕ positions for a forward TGC in T1 station (T1F).

6. CSC_Digitization

The digitization in the CSC is the simulation of the charge distribution on the CSC cathode strips given a hit in the sensitive gas. The process also identifies the strips numbers and

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 TGCDigitiValidation ntuple.*

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: *Control parameters for the TGCDigitValidation.*

their orientations. Thus after processing all the hits in the event, the CSC digitization outputs the list of digits into the TDS from where they could be picked up by other algorithms, for example for the simulation of the Raw Data objects (RDO) or for clusterization. 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 and the hit time. In the digit, the charge is given in number of equivalent electrons.

A detailed description of the CSC detector in the ATLAS Muon Spectrometer can be found elsewhere [2]. Here only the digitization process is reported. An incident charged particle, traversing the CSC gas, may produce primary and secondary ionization electrons, leading the formation of an avalanche on the anode wire as illustrated in Figure 18. The induced charge distribution on the segmented cathode can be written as follows:

$$\Gamma(\lambda) = K_1 \frac{1 - \arctan(K_2\lambda)}{1 + K_3 \arctan(K_2\lambda)} \tag{6.1}$$

where the constants K_2 and K_3 are related empirically as

$$K_2 = \frac{\pi}{2}\left(\frac{1}{2}K_3^{1/2}\right) \tag{6.2}$$

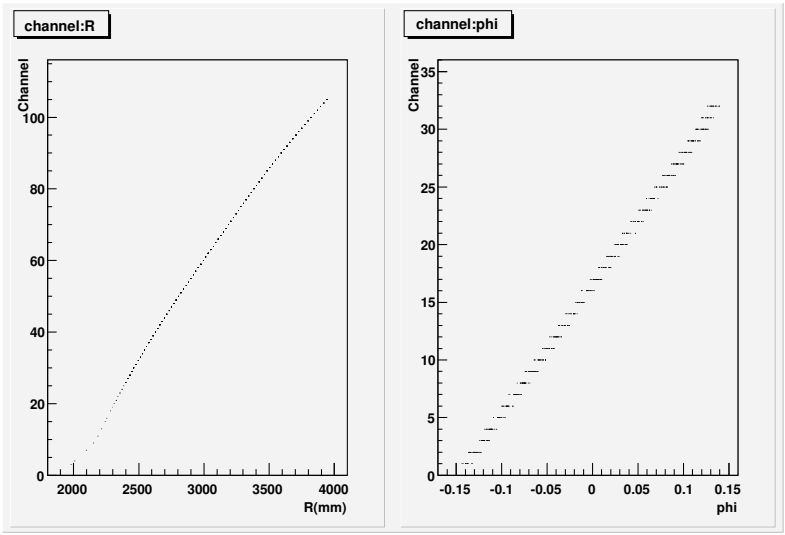

Figure 17: *Channel number as a function of r (left) and ϕ positions(right). The relation between channel number and r position is not linear, because of the number of wires to be ganged depends on the r position. On the other hand, the channel position for the ϕ direction changes linearly according to ϕ position.*

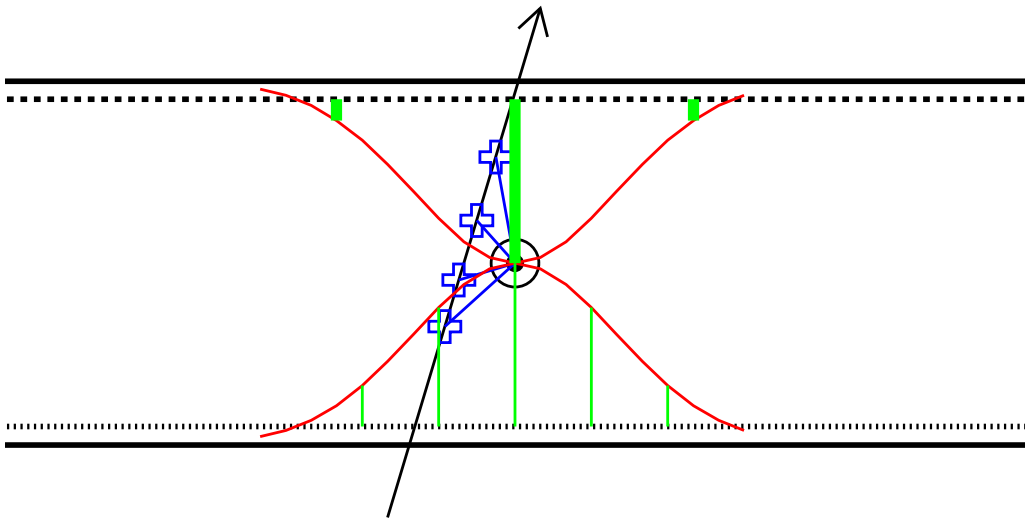

Figure 18: *Avalanche production in the CSC gas: the incident particle produces primary and secondary electrons along its path, leading to a charge multiplication and collection at the anode wire. This induces a distribution on the cathodes, shared equally between the two segmented cathode strips. The process of CSC digitization consists of finding the charge distribution on the cathode strips given of the collection of hit objects in the gas. Note that this is just a sketch: the charge drift to the anode should follow the field lines.*

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 6.2, the charge distribution of Equation 6.1 reduces to a one-parameter expression [2].

For a given simulated hit in the sensitive gas, the number of interactions along the 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 from test beam: this distribution closely follows the Landau energy loss distribution in a thin absorber, with a small probability of large tail. The charge collection at the anode (taking into account the gas gain and charge attenuation effects) and the charge distribution on the cathode are then simulated according to Equation 6.1. Since in the CSC the precision coordinate is determined by a relative measurement of the charge induced 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, the CSC performance is not affected by the drift time properties of the gas since there is 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. The amplifier noise is folded into the charge distribution as follows:

$$charge[i] = charge[i] + noise * Gaussian \tag{6.3}$$

by smearing the charge on the strip i with a Gaussian whose standard deviation is the noise in number of equivalent electrons.

The second stage in the digitization is the simulation of the raw data, i.e., the output of the CSC electronics. This output consists of a number of ADC values determined by the sampling rate of the amplifier signal. The bi-polar shape of the signal, shown in Figure 19 is modeled according to the following equation:

$$ADCValue = \left(1.0 - \frac{z}{n+1}\right) z^n \exp\left(-z\right) \tag{6.4}$$

where n is the number of amplifier integrations, and z is written as follows:

$$z = (samplingTime - startTime)/signalWidth \tag{6.5}$$

and $signalWidth$ is the width of the positive lobe of the bipolar signal. The charge on the strip is taken as difference between the maximum of the positive lobe and the baseline (pedestal). In the simulation, the charge on the strip is "known", as obtained from the hit digitization: the problem therefore consists of going from one charge value (in number of equivalent electrons or femto-Coulombs) to number N of ADC samplings (in ADC counts) at an assumed sampling rate — the CSC data may be taken at a variable sampling rates. The signal shape is normalized to the charge (in ADC counts) and sampled to generate the

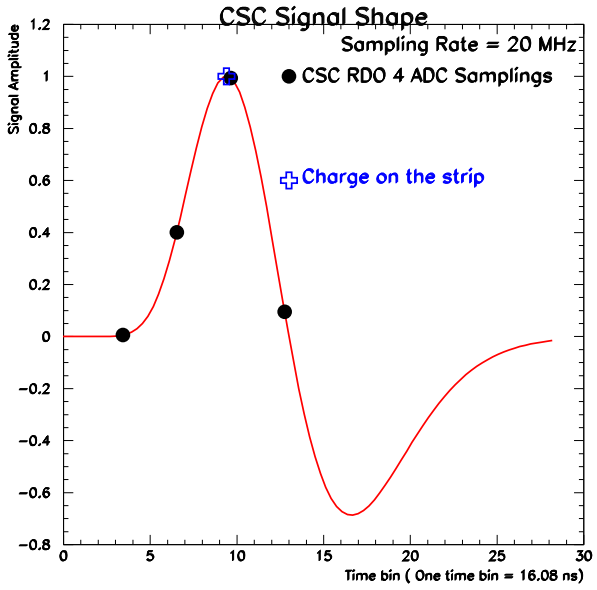

Figure 19: *CSC amplifier signal shape. The signal can be sampled at an adjustable rate and a voltage comparator gives the ADC value at each sampling. The charge on the strip is taken as the difference between the maximum of positive lobe and the baseline. Finding the maximum and the baseline requires a fit to the ADC samplings during the reconstruction. Then the calibration curve is also needed to convert the maximum to femto-Coulomb. The time of the peak, may be used either for a two-dimensional clusterization or to eliminate out-of-time hits that would otherwise spoil the determination of the precision coordinate.*

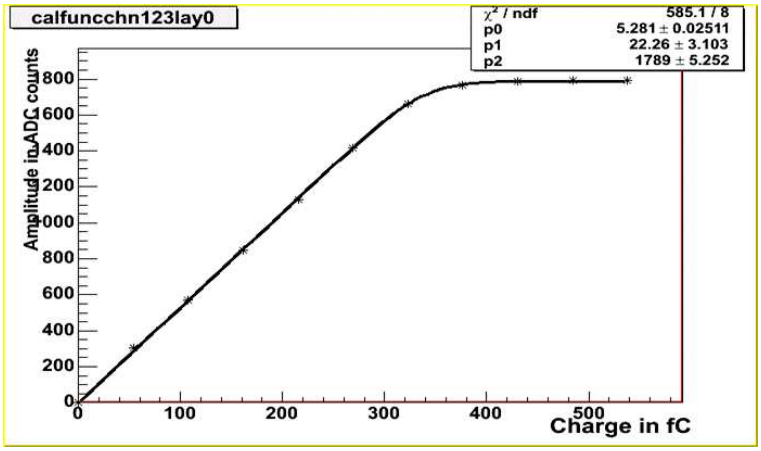

Figure 20: *CSC calibatrion curve for one strip. The calibration constants are written to and obtained from the conditions database.*

simulated raw data, i.e., the ADC samplings for each fired strip. The detailed simulation of what actually happens in the ROD (Read Out Driver) has not yet been carried out. Once the ADC samplings are obtained, the byte stream is encoded, using the expected format

of the ROD, to simulate the production of the CSC raw data.

The conversion of the simulated charge on the strip into the ADC counts requires the calibration constants for the strip and the calibration curve. An example of a calibration curve for one strip in shown in Figure 20.

6.1 Class Methods

The simulation software of the charge segmentation on the CSC cathode strip is in the ATLAS CVS repository under [4]. It consists of two algorithms: one called CSC_Digitizer which, given a single hit in the sensitive gas, produces the charge distribution on the cathode strips as described above. The other one, CscDigitBuilder manages the whole digitization process:

- gets the collections of GEANT4 hits to be digitized
- delegates the digitization of a single hit to the CSC_Digitizer algorithm
- creates the collections of digits, organized according the CSC chamber identifiers and save the digitized data in the TDS where they could be picked up either by a reconstruction algorithm or the algorithms that handle the simulation of the electronic output.

There are also a number of runtime parameters (or properties) to control the digitization process without having to modify and recompile the code. The parameters are the noise level of 1000 electrons and the digitization time window set by default to ± 50 ns with respect to the main bunch crossing.

6.2 CSC_Digitization Validation

To validate the CSC digitization, one considers the GEANT4 tracking of very high energy single muons, originating at a fixed vertex (no spreading, no magnetic field), without the production and tracking of secondary particles. These are therefore stiff tracks whose direction at the plane of the CSC strip is given by their direction at the vertex. In each strip plane, one can then plot the distance (residual) from the strip with highest charge to the track: this residual should be a flat distribution contained between ± 0.5 of the strip width. This residual distributions are shown in Figure 21 for the precision and the non-precision strips. Further validation is done during the reconstruction where the simulated CSC raw data is read, either as byte stream or persistified RDO in POOL. From the byte stream, the ROD format is used to decode the byte stream into the transient RDO. As explained before, the CSC raw data is series of ADC samplings in ADC counts.

During the reconstruction, a parabolic fit is done around the maximum shown in Figure 19 to extract the maximum relative to the baseline and the peaking time. The calibration curve — of Figure 20 for example — is then used to convert the relative maximum ADC counts from the parabolic to the strip charge in number of electrons. A clusterization is done on the charge/peaking time distribution. The plot in Figure 22 is the residual of the cluster positions in the planes of the anode wire: what is plotted is the local distance

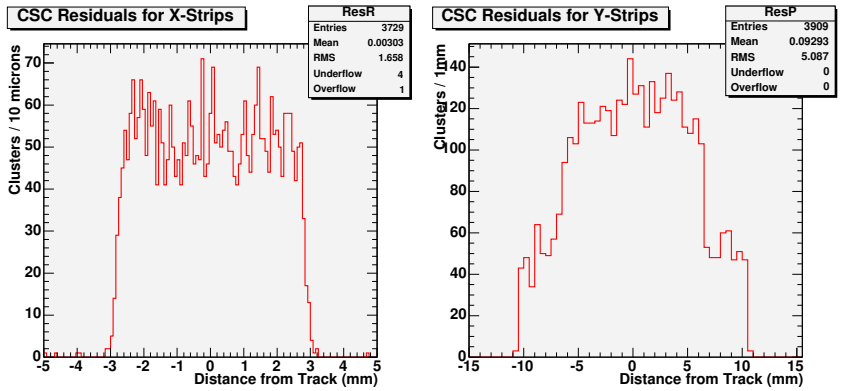

Figure 21: *Validation of the CSC digitization: the distance from the strip with the highest charge in a strip plane to the track. This is expected to be a uniform distribution between ± half the strip width. The width of the strips can be seen, especially in the right plot, for the non precision strips, where the strips width changes significantly from the small to the large chamber. This exercise validates the geometry of the CSC (local, global positions) and the charge segmentation at the cathodes.*

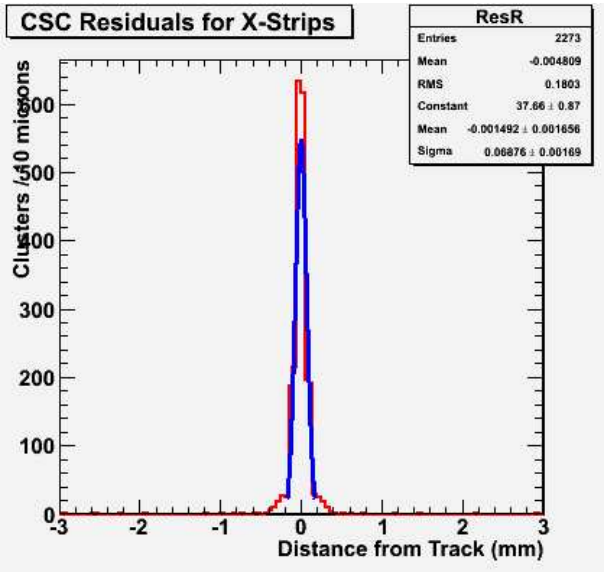

Figure 22: *Validation of the CSC digitization including the simulation of the electronic ouptut. A clusterization is done and the residual is plotted. The residual is determined as the local distance from the cluster centroid to the track in the plane of the wires. Muon tracks, originating from a fixed vertex (no vertex spreading), normally incident at the center of the chamber (no magnetic field) are considered. The residuals from all the chambers are integrated in this plot. It is therefore an estimate of the intrinsic single hit resolution of the precision strips.*

from the track position obtained from the clusterization to the true track position at wire plane. The muon tracks are normally incident on the chamber plane at the center of the chamber in order to avoid the geometrical effect due to the finite size of the chambers.

The single hit resolution of the precision strips is of the order 68 microns as can be seen in Figure 22.

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 GEANT4 simulation.

8. Acknowledgments

The authors would like to thank Davide Costanzo for his 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] LCG Applications Area/POOL Webpage, `http://lcgapp/cern.ch/project/persist`.

[2] ATLAS Collaboration, "*Muon Spectrometer Technical Design Report*", CERN/LHCC/97-22 (1997).

[3] K. A. Assamagan *et al.*, "*Definition of "Raw Data Objects" for the MDT Chambers of the Muon Spectrometer*", ATL-DAQ-2003-022 and K. A. Assamagan *et al.*, "*Raw Data Objects Definition for the RPC chambers of the ATLAS Muon Spectrometer*", ATL-DAQ-2003-018.

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

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

[6] S. Spagnolo *et al.*, "*The Description of the Atlas Detector*", Proceedings of CHEP04, Interlaken, Switzerland, 27 Sep. - 1 Oct. 2004.

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

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

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

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

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

[12] Muon Dictionaries, in the ATLAS CVS repository (offline/ DetectorDescription/ IdDictParser).

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

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

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

[16] C. Posch, E. Hazen and J. Oliver, "*CMOS front-end for ATLAS MDT*", ATL-MUON-2002-003, June 2002.

[17] Y. Arai, "*AMT-3 (ATLAS Muon TDC version 3) User's manual*"`atlas.kek.jp/tdc/amt3/AMT3manual_032.pdf`.

[18] RunTimeTest Web Page, `http://www.hep.ucl.ac.uk/atlas/AtlasTesting/RTTUserGuide/RTTUserGuide.html`.

[19] AMDB Web Page, `http://atlas.web.cern.ch/Atlas/GROUPS/MUON/AMDB_SIMREC/amdb_simrec.html`

[20] R. Veenhof, GARFIELD, version 7.04, CERN Program Library W5050.